RASPBERRY PI для ДЕТЕЙ

Книга предназначена для первоначального изучения Raspberry PI – одноплатного компьютера размером с банковскую карту. Благодаря простым пошаговым инструкциям юный читатель сможет превратить свою «малинку» и в настоящий медиацентр, и в автосимулятор (с помощью языка Scratch), и в сердце умного дома (программируя на Python и подключая разные датчики). А в конце будет создан настоящий сайт и протестирован на смартфоне.

Издание рекомендуется школьникам средних и старших классов, желающим изучить возможности Raspberry PI и научиться в нем программировать.

Интернет-магазин:
www.dmkpress.com

Оптовая продажа:
КТК «Галактика»
e-mail: books@alians-kniga.ru

ДМК ПРЕСС ИЗДАТЕЛЬСТВО
www.дмк.рф

ISBN 978-5-97060-667-4

RASPBERRY PI для ДЕТЕЙ

RASPBERRY PI для ДЕТЕЙ

Михаэль Вайгенд

mitp
ДМК ПРЕСС ИЗДАТЕЛЬСТВО

RASPBERRY PI для ДЕТЕЙ

Михаэль Вайгенд

Raspberry Pi для детей

Michael Weigend

Raspberry Pi für Kids

mitp

Михаэль Вайгенд

Raspberry Pi для детей

ДМК Пресс
издательство
Москва, 2019

УДК 004.738, 004.62, 519.6
ББК 32.973
В12

Вайгенд М.
В12 Raspberry Pi для детей / пер. с нем. Ю. Ю. Энглерт. – М.: ДМК Пресс, 2019. – 564 с.: ил.

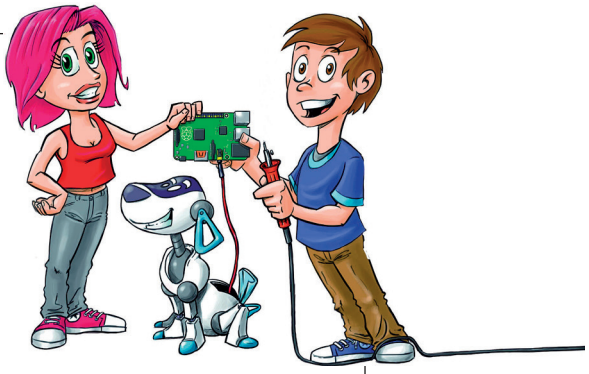
ISBN 978-5-97060-667-4

Книга предназначена для первоначального изучения Raspberry PI – одноплатного компьютера размером с банковскую карту. Благодаря простым пошаговым инструкциям юный читатель сможет превратить свою «малинку» и в настоящий медиацентр, и в автосимулятор (с помощью языка Scratch), и в сердце умного дома (программируя на Python и подключая разные датчики). А в конце будет создан настоящий сайт и протестирован на смартфоне.

Издание рекомендуется школьникам средних и старших классов, желающим изучить возможности Raspberry PI и научиться на нём программировать.

УДК 004.738, 004.62, 519.6
ББК 32.973

ISBN 978-3-95845-768-3 (нем.)
ISBN 978-5-97060-667-4 (рус.)



Содержание

Введение 13

Raspberry Pi 18
- От материнской платы к готовому компьютеру 18
- Установка программного обеспечения 22
- Первый запуск Raspberry Pi 30
- Рабочий стол 35
- Как подключить Raspberry Pi к интернету 38
- А теперь посчитаем 39
- Работаем с файловым менеджером 41
- Ввод Unix-команд в приложении LXTerminal 46
- Вопросы 51
- Задание: установка фона для рабочего стола 52
- Ответы на вопросы 53
- Решение задачи: установка фона рабочего стола 53

Создаём медиацентр и киоск (интерактивный терминал) 55
- Как слушать музыку с помощью MOC 56
- «Безголовый» Raspberry 59
- Проект 1. Музыкальный центр с дистанционным управлением 65

1

2

Проект 2. Создание интерактивного терминала 67
Проект 3. Raspberry Pi как медиацентр 71
Вопросы 80
Ответы 80

3
Автогонки и метеоры: как их программировать в Scratch? 82
Запуск Scratch 83
Проект 4. «Ухуху-у-у!» – первый Scratch-проект 85
Проект 5. Гоночная трасса Формулы 1 95
Проект 6. «На помощь! Метеориты!» 106
Студии программы Scratch 120
Задания 123
Решение задач 125
Ответы на вопросы 127

4
Мультяшные истории 128
Проект 7. Шуточный мультфильм 128
Проект 8. Интерактивная анимация – синхронизация через сообщения 140
Проект 9. Викторина 151
Тестируем проект 161
Вопросы 161
Задания 162
Ответы на вопросы 173
Ответы на задания 173

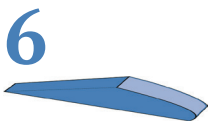
5
Создание проектов с помощью Picoboard 178
Плата Picoboard 178
Проект 10. Магические слова – распознавание речи 182
Проект 11. Создаём измеритель уровня звука 186
Проект 12. Игра «Пинг-понг» 191
Проект 13. Устройство для приготовления лимонада 197

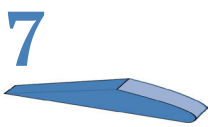
Вопросы 205
Задания. Экспозиметр 205
Ответы на вопросы 206
Ответы на задания 207

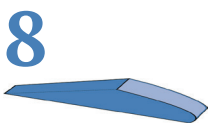
Интерактивные игры и симуляторы 208
Проект 14. «Помоги утке!» 209
Проект 15. «Поймай комара» 213
Проект 16. Формула 1 223
Вопросы 230
Задания. Садимся на Луну 230
Ответы на вопросы 232
Решение заданий 234

Знакомство с Python 237
Что такое Python? 237
Оболочка Python 238
Первый скрипт для Python 245
Интерактивные программы 253
Ввод и вывод данных 256
Проект 17. Тормозной путь 257
Имена и переменные 263
Вопросы 266
Задания 266
Ответы на вопросы 269
Решение задач 269

А что это там мигает? Управляем светодиодами с помощью Raspberry Pi 272
Сигналы SOS. Как подавать их с помощью команд Python и светодиодов? 272
Проект 18. Программируем сигнал SOS 284
Вопросы 287
Задания. Создай два мигающих светодиода 288







Ответы на вопросы 289
Ответы на задания 289

9
Компьютер принимает решения 291
Оператор ветвления 291
Проект 19. А что это за пластик? 295
Условия 298
Условный повтор или инструкция while 300
Проект 10. Угадай число 301
Световые сигналы 303
Проект 21. Простая мигалка 304
Проект 22. Шаблон мигалки 305
Вопросы 305
Задания. Идти на улицу или не идти? 305
Ответы на вопросы 306
Решение задачи 306

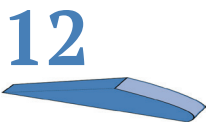
10
Управление с помощью переключателя 308
Переключатель 308
Проект 23. Счётчик 311
Проект 24. Дверной звонок – проигрываем звуковые файлы 314
Проект 25. Сигнализация 320
Проект 26. Единички и нолики. Перфокарта в качестве цифрового ключа 323
Вопросы 333
Задания 333
Ответы на вопросы 334
Решение задач 335

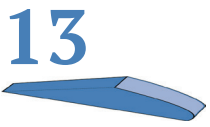
11
Светодиодные дисплеи 338
Проект 27. Светодиодная матрица 339
Проект 28. Перемещающиеся светящиеся линии 346

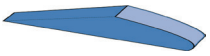
Блок со светодиодным матричным индикатором 349
Проект 29. Управление отдельными светодиодами 350
Вечно одно и то же!.....354
Вопросы 354
Задания.....354
Ответы на вопросы 356
Решение заданий 356

Сбор данных и их обработка.....359
Коллекции 359
Обработка секвенций 361
Проект 30. Планеты.....365
Проект 31. Вытяни карту 372
Проект 32. Учим лексику 377
Проект 33. Световой сигнал 379
Проект 34. Светодиодные буквы.....383
Вопросы 385
Задание. Гороскоп.....385
Ответы на вопросы 386
Решение задачи.....387

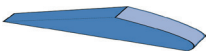
Работа с ЖК-индикатором.....388
Работа с ЖК-дисплеем.....388
Как отобразить текст на ЖК-индикаторе?.....393
Кусочек целого. Срез.....395
Проект 35. Цифровые часы с ЖК-индикатором 396
Проект 36. Таймер.....397
Вопросы 401
Задание. Блуждающие звёзды 401
Ответы на вопросы 402
Решение задачи.....402



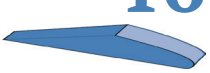




Проекты с использованием ультразвукового датчика 404
Какие бывают функции? 404
Проект 37. Каков размер окна в доме? 412
Проекты с использованием ультразвукового датчика 415
Проект 38. Измерение расстояния 419
Проект 39. Ориентация в пространстве с помощью ультразвука 425
Вопросы 432
Задания 433
Ответы на вопросы 434
Ответы на задания 435



Измерение температуры и система «Умный дом» 438
Измерение температуры 438
Проект 40. Делаем замеры температурных изменений 442
Проект 41. Сохранение данных в формате CSV 445
Как управлять беспроводной розеткой? 450
Проект 42. Отправляем секретные команды ночью 457
Другие проекты 460
Вопросы 461
Задания 461
Ответы на вопросы 463
Решение задач 464



Графический пользовательский интерфейс 468
Как создать пользовательский интерфейс? 468
Проект 43. «Сегодня ты выглядишь великолепно!» 469
Картинки в виджетах 474
Цвета 475
Проект 44. «Смешивание цветов» 476
Проект 45. Сигнальная азбука 477

Проект 46. Делаем гимнастику с ультразвуком 482
Делаем выбор с помощью экранных переключателей и чекбоксов 486
Проект 47. Выбери цвет 486
Проект 48. Меню-консультант 489
Вопросы 491
Задания. Таймер.....491
Ответы на вопросы 492
Решение задач.....492

Работа с камерой 494

17

Модуль видеокамеры.....494
Тестируем модуль камеры.....496
Программное обеспечение камеры.....497
Модуль PIL 499
Проект 49. Распознаём движение 500
Проект 50. Покадровая замедленная съёмка 504
Проект 51. «Цветной ключ» 506
Вопросы 511
Задания.....511
Ответы на вопросы 514
Решение задач.....514

Raspberry Pi в качестве веб-сервера – всегда к вашим услугам.....518

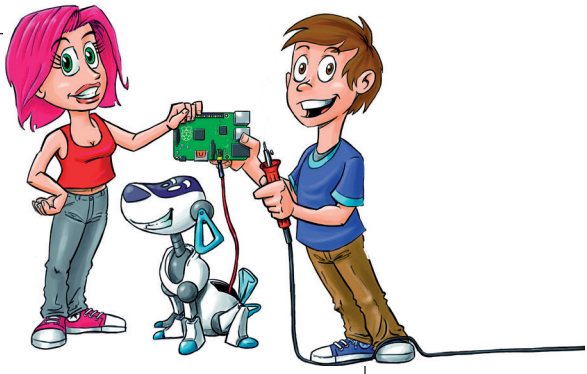
18

Как настроить Raspberry Pi в виде сервера?.....518
Проект 52. Который час? Создаём динамические веб-страницы 525
Проект 53. Шпион в саду 529
Лёгким движением руки мобильник превращается… в модем 533
Проект 54. Совершенно секретно! Создаём сайт с защитой доступа 534
Проект 55. Управление светодиодом через сайт 539
Проект 56. Управление домашними устройствами через беспроводную сеть 542

Вопросы 546
Задание. Измерение температуры через сеть 547
Ответы на вопросы 547
Решение задачи....548

Примечания для родителей и преподавателей....551
Что нам нужно для работы? 552
Список покупок....553
Почта со всех концов света, или Как заказать онлайн 556
Как работать с книгой?....557

Указатель....559



Введение

Raspberry Pi – это хоть и маленький, но настоящий компьютер, который состоит всего из одной небольшой платы. Но на этом маленьком компьютере ты можешь создавать собственные проекты. Название этого компьютера – Raspberry Pi – состоит из двух слов: Raspberry – «малина» и Pi – число «пи». А логотипом этого компьютера стало изображение малины. Стоит такой компьютер совсем недорого, а все программные средства, которые понадобятся для работы с ним, и вовсе бесплатные. С «малинкой» так и хочется поэкспериментировать и поиграть, но всё же это не игрушка. Raspberry Pi – надёжная машинка, и используют ее не только для обучения, но и для серьёзных проектов в области техники и экономики.

Этот компьютер, как, впрочем, и его старшие собратья, состоит из аппаратной части и программного обеспечения. Аппаратная часть – это все то, что можно увидеть и потрогать руками (корпус, плата со смонтированными на ней деталями компьютера, клавиатура, монитор). А вот программное обеспечение – это всевозможные программы и данные, которые находятся в самом компьютере и которые контролируют его работу. Все программы создаются программистами. А пишутся программы на языке, который компьютер понимает. Этот язык и называется языком программирования. Языков программирования в мире много. Так как главная тема этой книги – программирование, мы будем использовать такие языки программирования, как Scratch и Python. Эти языки программирования облегчат изучение темы. С помощью программирования ты закрепишь свои познания в компьютерной технике. Можешь по-

дойти к этому творчески. Без знаний программирования изучать «малинку» будет сложнее.

- Scratch – это язык программирования для начинающих. Ты создаешь программу прямо на экране при помощи мышки. Много ошибок не наделаешь, зато быстро освоишь базовые идеи программирования. Язык программирования Scratch поможет тебе прямо на экране создавать мультфильмы и игры.
- Python – это язык программирования для профессионалов. Но и его тоже легко освоить. Python используется в промышленности для работы с крупными и специализированными проектами. С помощью Python ты сможешь создавать программы, шифровать данные, управлять роботами и многое другое.

Отличительная особенность «малинки» от обычного компьютера (планшета, ноутбука, стационарного компьютера) состоит в том, что ты по своему усмотрению можешь изменять его аппаратную конфигурацию. То есть для каждого проекта ты можешь из набора деталей собрать компьютер, по своим техническим характеристикам наиболее отвечающий поставленным задачам. Прежде всего (кроме, конечно, самого Raspberry Pi) тебе понадобятся:

- такие электронные устройства, как монитор, клавиатура, мышка и колонки, которые, скорее всего, у тебя уже есть, только ты используешь их для других целей;
- электронные схемы, которые ты соберёшь из маленьких деталей, таких как сопротивления, конденсаторы, сенсоры и светодиоды;
- другие механические элементы, которые ты сможешь самостоятельно в считанные минуты смастерить из бумаги, фольги, проволоки, липкой ленты и прочих подручных материалов;
- несколько специальных устройств, таких, например, как RPi-модуль и чертёжная доска Scratchboard.

Как читать эту книгу

Эта книга состоит не только из пояснительного текста, но и из инструкций. Шаг за шагом следуя инструкции, ты научишься создавать программы. Такие пояснительные тексты порой понять бывает довольно сложно. Но, скажу по собственному опыту, соблюдение некоторых правил поможет тебе. Итак, вот эти три основных правила.

1. Шаг первый – делай постепенно! Сначала читай инструкцию и вникай в прочитанное. Сконцентрируйся на прочитанном и не спеши читать дальше.
2. Шаг второй – не бойся делать ошибки! Даже если сомневаешься, всё ли ты понял правильно, начни воплощать в жизнь этот шаг. Здесь тебе понадобbтся немного смелости и отваги (даже для выполнения первого шага).
3. Шаг третий – понял, значит, экспериментируй! Если первый шаг выполнен, то ты увидишь результат. Соответствует ли он тому, что написано в инструкции? Если нет, перечитай этот пункт еще раз. Может быть, ты понял написанное как-то иначе. Попробуй снова!

Немного о содержании книги

Книга состоит из трех частей.

В первой части (главы 1–2) речь идет о подготовке к работе с «малинкой». Прежде всего расскажем об установке основной программы, обеспечивающей работу компьютера: операционной системы Raspbian. Дальше ты узнаешь, как установить программные средства и игры и как можно использовать Raspberry Pi в качестве медиацентра – слушать музыку и смотреть фильмы. Никаких особых знаний для этого тебе не потребуется. Все необходимое ты узнаешь, читая книгу и выполняя инструкции.

Во второй части книги (главы 3–6) ты начнёшь программировать с помощью Scratch. Для этого нужно будет собрать модули программного текста. Делается это быстро и легко. Таким образом ты сможешь программировать видео, игры, автосимуляторы и многое другое. Для работы с несколькими проектами тебе понадобится Picoboard (сенсорная доска). Она состоит из курсора и звуковых и световых сенсоров. С помощью этого устройства ты сможешь собрать систему наведения и педаль газа для автосимулятора, управлять курсором на экране или с помощью светового сенсора проверить количество сахара в лимонаде. Фантазия не знает границ.

В третьей части (главы 7–18) мы начнем работать с программой Python. С помощью этого языка программирования ты начнешь писать программы для управления светодиодами, для управления домашней техникой с помощью радиосигнала, научишься измерять температуру или с помощью ультрафиолетового сенсора контролировать спортивные упражнения. С помощью камеры (или инфракрас-

ной камеры) твой Raspberry Pi будет наблюдать за садом. С помощью видеокамеры и программы, написанной на Python, твой компьютер сможет обнаружить и моментально отреагировать на движение или распознать образец краски на кодовой карте. В последней главе речь идет о том, как можно использовать Raspberry Pi в качестве веб-сервера. Ты создашь веб-сайты, которые затем сможешь протестировать на своем смартфоне. Всё это звучит очень страшно и сложно. Но программные тексты небольшие, меньше страницы. Так что ты сможешь всё быстро записать, проверить и усовершенствовать.

В приложении есть глава для родителей и преподавателей. Там находится список всех устройств и электронных деталей, которые понадобятся для проектов. К счастью, почти всё сейчас можно заказать в интернете. Но для пользования онлайн-магазинами тебе понадобится помощь родителей.

Структура главы

Еще пару слов о структуре некоторых глав. Ближе к концу каждой главы ты найдёшь несколько несложных вопросов, касающихся ее содержания. Они помогут тебе ещё раз обдумать основные моменты этой главы.

В каждой главе будет рассмотрена новая тема, для которой следует написать отдельную небольшую программу. Написание таких программ и будет заданием по программированию. Но эти задания будут несложными, т. к. в каждой главе ты найдешь большое количество указаний и подсказок.

Ну и в самом конце главы ты найдешь развёрнутые пояснения к заданиям и ответы на вопросы.

Указания и дополнительная информация
В книге ты будешь встречать значок с изображением собачки. Этот значок показывает, что выделенный голубым цветом текст – это дополнительная информация, которая может оказаться очень важной.

Практические советы
Советы и подсказки в каждом разделе обозначены значком с восклицательным знаком, а сам тест подсказки или совета выделяется фиолетовым цветом.

Загрузка информации с веб-сайта

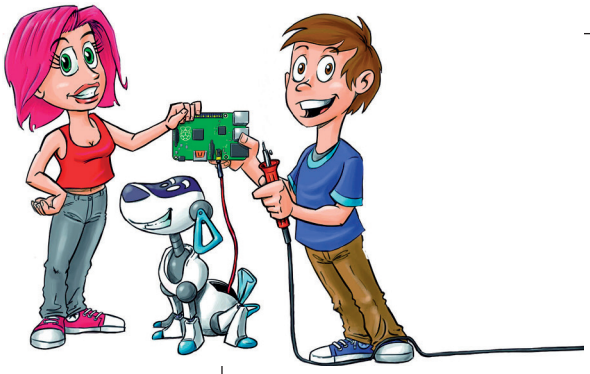
Все программные тексты ты можешь скачать с официального веб-сайта, расположенного по адресу www.mitp.de/767. Здесь образцы программ находятся в одном zip-архиве. Чтобы его скачать, перейди по ссылке **Programmbeispiele** (Программные игры) и кликни на строке **Downloads** (Загрузка). Сохрани этот файл в удобной для тебя папке. Далее распакуй скачанный zip-архив в папке, где ты его сохранил. После разархивации в описании проекта ты для каждой главы увидишь папку, в которой будет сохранена программа для игры. Точную пошаговую инструкцию, как скачать программные тексты на «малинку», ты найдешь в седьмой главе.

Кроме того, на сайте находятся дополнительные главы, для которых не нашлось места в бумажном формате книги. В первой речь идёт о создании с помощью Raspberry Pi научных проектов. В этих главах объясняется, как собрать устройства, которые можно использовать для химических и физических экспериментов. Например, как собрать фотометр, с помощью которого можно определить содержание красителя. Или описание простого спектрофотометра, предназначенного для проведения спектрального анализа и определения с его помощью видов красителей. Здесь ты также найдешь примеры симуляторов и моделей, иллюстрирующих естественные процессы.

Основная тема второй дополнительной главы – это обработка картинок. Ты узнаешь, как автоматически создать модель и сохранить её в виде векторной графики. Кроме того, будут использованы модули библиотеки (PIL – Python Image Library) объектно-ориентированного программирования (Definition von Klassen).

Третья дополнительная глава описывает эксперименты с инфракрасной камерой.

Если ты хочешь рассказать об успешном проекте или указать на найденные в книге ошибки, то можешь обратиться к автору по адресу: mw@crative-informatics.de.



1
Raspberry Pi

В первой главе речь идёт о том, как собрать функциональный компьютер и установить на него программное обеспечение.

Ты уже знаком с операционной системой Linux? Ну, или хотя бы слышал о ней? Дистрибутив *Raspbian*, который мы используем, как раз и создан на базе операционной системы Linux Debian и хорошо приспособлен для нашей «малинки». Ты копируешь файлы, создаёшь нового пользователя, подключаешь компьютер к сети и устанавливаешь игры и полезный софт.

От материнской платы к готовому компьютеру

Изначально Raspberry Pi не является полноценным компьютером. Это материнская плата с процессором и многочисленными разъёмами для клавиатуры, монитора и прочих устройств. В этом разделе речь пойдёт об аппаратной части, т. е. о частях, которые можно потрогать руками. В этой главе ты узнаешь, как из твоего Raspberry Pi собрать полноценный компьютер. Самая новая версия материнской платы называется Raspberry Pi 3 модель b и появилась рынке 29 февраля 2016 года. Если твой Raspberry Pi более ранней

версии, ничего страшного. Все проекты, описанные в этой книге, работают и с другими моделями.

Сначала о главном

Возможно, ты сейчас читаешь этот раздел вместе со своими родителями. В приложении приводится список комплектующих, которые понадобятся для создания проектов. Поэтому спроси родителей, смогут ли они обеспечить тебя этими деталями.

Помимо самого компьютера Raspberry Pi, потребуются следующие устройства:

- карта памяти (SD-card). На ней будут храниться все программы и данные. Для работы с Raspberry Pi тебе понадобится маленькая карта микро-SD. Она должна быть объёмом, как минимум, 4 ГБ и со скоростью обмена данными не менее 10 МБ/с. В 1 ГБ содержится примерно 1 млрд бит;
- клавиатура, подключаемая через USB-разъём. Это самая обычная клавиатура с кабелем. Но если проводной клавиатуры нет, подойдет и беспроводная;
- электропитание. На материнской плате Raspberry Pi имеется маленький USB-вход, который можно использовать для подключения к электросети. Для питания компьютера нужно использовать блок питания с напряжением 5 В и силой тока не менее в 1000 миллиампер (1000 мА). Лучше всего для этого подходит зарядное устройство для мобильного телефона (стоимость от 700 руб.);
- монитор с разъёмом HDMI. Первые буквы аббревиатуры HD означают высокое разрешение. MI – это мультимедийный интерфейс. Всё это нужно для одновременной передачи картинки и звука. Лучше всего взять монитор с разрешением Raspberry Pi со встроенными динамиками. Конечно, для подключения монитора понадобится кабель HDMI. Для мониторов со входом VGA или DVI будет нужен адаптер (его стоимость от 800 руб.). С помощью VGA (от англ. *Video Graphics Array* – видеографическая матрица) и DVI (от англ. *Digital Visual Interface* – цифровой видеоинтерфейс) тоже можно получить разрешение высокого качества. Но Raspberry Pi способен работать и с мониторами более старых моделей, чьё минимальное разрешение составляет 640×480;

- мышь USB;
- для Raspberry Pi второго поколения понадобится Wi-Fi-адаптер. Конечно, компьютер можно подключить и без Wi-Fi-адаптера, напрямую, используя интернет-кабель. Но, применяя беспроводное соединение Wi-Fi, ты освободишься от части проводов и сможешь подключить Raspberry Pi к смартфону на базе операционной системы Android или к iPhone. Обрати внимание: Raspberry 3 уже имеет встроенный модуль для подключения к Wi-Fi. Поэтому USB-адаптер беспроводной сети (его стоимость примерно от 800 руб.) потребуется только для Raspberry Pi более ранних версий.

Многие из этих вещей, возможно, у тебя уже есть. И ты можешь попробовать, будут ли они работать с твоим компьютером. Как правило, проблем с этим не возникает.

Но если ты купишь какую-то новую деталь, проверить ее на совместимость (подходит ли она к твоему компьютеру?) можно, пройдя по ссылке: http://elinux.org/RPi_VerifiedPeripherals.

Всё остальное оборудование, которое может тебе понадобиться для специализированных проектов, описано в приложении. Возможно, некоторые детали придётся заказать в интернете. Попроси родителей, чтобы они тебе в этом помогли.

Структура Raspberry Pi

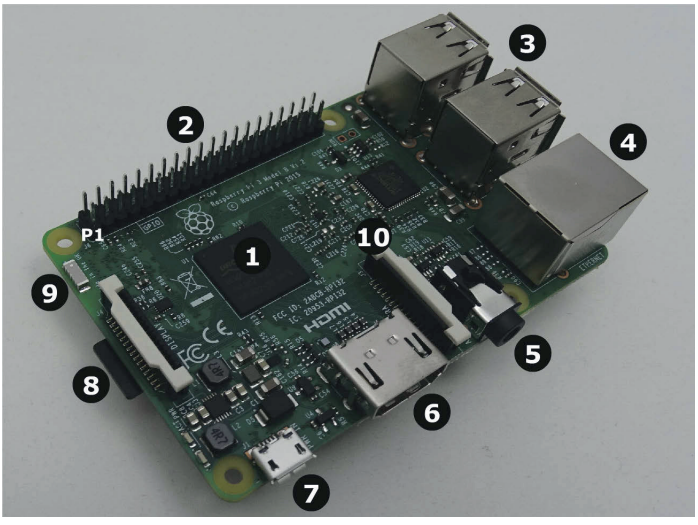


Рис. 1.1. *Raspberry Pi модель 3B*

На рис. 1.1. показана структура Raspberry Pi 3.

1. Это процессор. Важнейшая часть, или сердце, любого компьютера. Процессор обрабатывает все данные. В основном он состоит из многочисленных электронных ключей (переключателей), которые, открываясь и закрываясь, пропускают электрический сигнал в нужном направлении. То есть эти электронные ключи управляют электрическими сигналами, направляя их по определённой траектории. В твоём Raspberry Pi установлен мощный четырёхъядерный процессор с тактовой частотой 1,2 ГГц. Значение 1,2 ГГц обозначает, что процессор выполняет, и это означает, что в секунду он выполняет 1200 млн вычислений. Аббревиатура ARM означает *Advanced RISC Machines* (усовершенствованный риск-процессор). Но к слову «риск» это не имеет никакого отношения. Это сокращённая форма от *Reduced Instruction Set Computer*, т. е. компьютер с упрощенным набором команд. Особенность такого процессора – это малое потребление электричества. Поэтому его используют и в мобильных телефонах. Ещё одна деталь: как уже упоминалось ранее, это четырёхъядерный процессор. То есть один четырёхъядерный процессор состоит из четырёх одноядерных, и все эти четыре ядра работают одновременно и совместно.
2. Интерфейс GPIO. Эта аббревиатура от английского *General Purpose Input Output Device*, что означает «устройства ввода-вывода общего назначения». Этот интерфейс имеет 40 разъёмов, выполненных в виде контактных штырьков, припаянных к соответствующим электрическим дорожкам материнской платы. С помощью проводов-перемычек ты можешь подключать к компьютеру различные электронные схемы. Важно знать, где находится штырёк № 1, от которого ведется нумерация остальных разъёмов. На картинке штырёк № 1 обозначен как P1. Больше информации об устройстве ввода-вывода смотри в пятой главе.
3. У Raspberry Pi третьей модели есть четыре USB-разъёма. К ним ты можешь подключить клавиатуру и мышь.
4. Разъём RJ45. К этому разъёму подключается интернет-кабель для локальной сети. Этот кабель для проводного интернета подключается напрямую к компьютеру. Такое соединение намного практичнее, чем Wi-Fi.
5. Аналоговый аудиовыход с дополнительным композитным видеовыходом. Аудиовыход служит для подключения наушников или колонок. Колонки – это громко-

говоритель со встроенным усилителем и собственной системой энергопитания. К видеовыходу подключается телевизор.

6. Разъём HDMI. Через этот разъём с помощью кабеля HDMI к компьютеру подключается монитор. Эта аббревиатура расшифровывается как *High Definition Multimedia Interface*, а говоря русским языком, «мультимедийный интерфейс с высоким разрешением». То есть монитор, подключенный к этому разъему, обеспечивает очень хорошее качество картинки с разрешением 1920×1080 пикселей. А «мультимедийный» означает, что через этот разъём передается не только видеокартинка, но и звук.
7. Микроразъём USB служит для подключения блока питания.
8. Держатель для карт внешней памяти микро-SD. Будь осторожен! Когда компьютер включён, ни в коем случае не трогай эту карту! Разъём устроен так, что при нажатии карта выскакивает. Если карта памяти выскочит, твой Raspberry Pi в одно мгновение потеряет все свои данные.
9. Это антенна Bluetooth и Wi-Fi. Служит для подключения твоего компьютера к телефону или домашней интернет-сети через беспроводную Wi-Fi-сеть.
10. Вход для подключения CSI-камеры (*Camera Serial Interface* – Последовательный интерфейс камеры) (см. главу 12).

Больше ничего не забыли? Ну конечно! Твой Raspberry Pi не имеет кнопки **ON/OFF**. Как только ты подключишь его к источнику питания, он сразу же начнёт работу. Но прежде чем ты это сделаешь, позаботься о том, чтобы программное обеспечение было установлено на карту микро-SD. Как это сделать, сейчас объясню.

Установка программного обеспечения

В предыдущем разделе речь шла об аппаратном обеспечении компьютера, т. е. о деталях, которые можно потрогать руками. Но, для того чтобы твой Raspberry Pi действительно работал, ему потребуется *программное обеспечение*. Это программы и данные, которые руками пощупать нельзя. Важнейшей частью программного обеспечения компьюте-

ра является операционная система. Она отвечает за работу всего компьютера, хранение данных, работу клавиатуры, мышки и других подключённых устройств, а также за пользовательский интерфейс. Короче говоря, операционная система несёт ответственность за основные функции компьютера. Без операционной системы не будет работать ни один компьютер. В этом разделе речь пойдёт о том, как установить операционную систему на твой Raspberry Pi. Для этого тебе понадобится другой компьютер (например, Apple Mac или компьютер с операционной системой Windows). Если у тебя пока нет своего компьютера, попроси кого-то, кто тебе поможет.

Для Raspberry Pi существует множество операционных систем. В этой книге мы будем использовать операционную систему *Raspbian* – версию операционной системы Linux. Точнее говоря, это модификация операционной системы Debian Jessie, которая подходит для Raspberry Pi как нельзя лучше. Эта операционная система устанавливается на SD-карту. Легче всего это сделать при помощи установщика NOOBS (*New Out Oft he Box Software*), в котором *Raspbian* уже есть. Говоря проще, *New Out Oft he Box Software* означает примерно следующее: «Новая система, с помощью которой можно быстро и легко установить программное обеспечение». Можно купить SD-карту с предустановленной NOOBS. Если у тебя такая карта памяти уже есть, то следующий раздел можешь пропустить. Если же у тебя такой карты нет, NOOBS можно бесплатно скачать и установить его на свою SD-карту. Как это сделать, я объясню ниже.

Как скачать NOOBS

Перед установкой на твой ПК, Mac или ноутбук SD-карту нужно подготовить. Если у тебя нет опыта обращения с компьютером, то здесь тебе пригодится помощь экспертов.

На официальном сайте Raspberry Pi http://www.raspberry-pi.org/downloads доступны для скачивания установщики NOOBS и другие операционные системы. Из двух предложенных вариантов выбери NOOBS (обрати внимание – NOOBS LITE не подходит!), кликни **Download ZIP**. Начнётся скачивание ZIP-архива, который нужно сохранить на жёстком диске. Чтобы не занимать слишком много места, все данные в архиве запакованы в один файл. Это называется архивацией файла. Архивацию можно назвать сжатием данных. Но тем не менее всё равно объем этого ZIP-файла остается большим, и для его загрузки потребуется некоторое время.

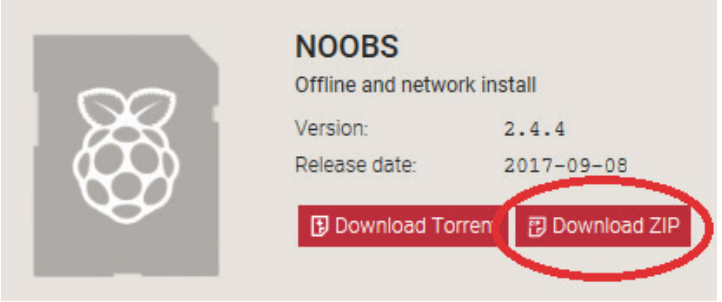


Рис. 1.2. *Загрузка NOOBS*

Следующим шагом станет распаковка архива, чем мы сейчас и займемся.

Распаковка архива на Mac OS X

Если у тебя компьютер марки Apple, то распаковать архив очень просто. Mac OS (начиная с версии 10.8 зовётся просто OS X) умеет это делать сам и не требует установки сторонних программ. Итак, чтобы распаковать скачанный тобою архив, нужно выполнить три шага. Сначала создай папку для данных NOOBS (команда **Создать** ⇨ **Новая папка**). Далее перемести ранее скачанный файл во вновь созданную папку. После дважды щелкни мышью на этом файле. Начнется распаковка. По окончании распаковки в папке появятся файлы и папки программы – загрузчика NOOBS.

Распаковка с помощью архиватора WinRAR (для Windows)

Операционная система Windows требует для распаковки архива установки дополнительного программного обеспечения. Если на твоем компьютере установлена операционная система Windows, то программа-архиватор, возможно, уже была установлена ранее. Самый распространенный архиватор – это *WinRAR*. Если на твоем компьютере подобной программы не оказалось, её придётся скачать и установить. Пробную бесплатную версию программы можно скачать, перейдя по ссылке http://www.win-rar.com.

Обычно по умолчанию скачиваемый файл сохраняется в папке **Загрузки** (Download). Чтобы найти на жестком диске скачанный ZIP-архив, воспользуйся поисковой системой. Каждая операционная система имеет функцию поиска. В Windows 10 ты найдёшь строку для ввода поисковых запросов в левом нижнем углу экрана, правее кнопки

Windows (см. рис. 1.3). Для поиска скачанного файла введи в поисковую строку слово «noobs» и осуществи поиск. Если архиватор WinRAR на твоем компьютере установлен, то по окончании поиска в списке найденных файлов ты увидишь значок скачанного архива в виде трех связанных книжек. Правее этого значка находится название файла (см. рис. 1.4).

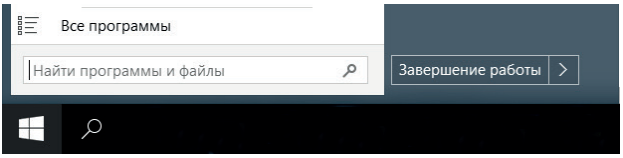


Рис. 1.3. *Поисковая строка Windows 10*

Рис. 1.4. *Значок архива с названием*

Чтобы распаковать архив, дважды щелкни мышью по его значку. Архиватор будет запущен, и на экране ты увидишь его окно со списком файлов и папок, которые были сохранены в архиве (см. рис. 1.5).

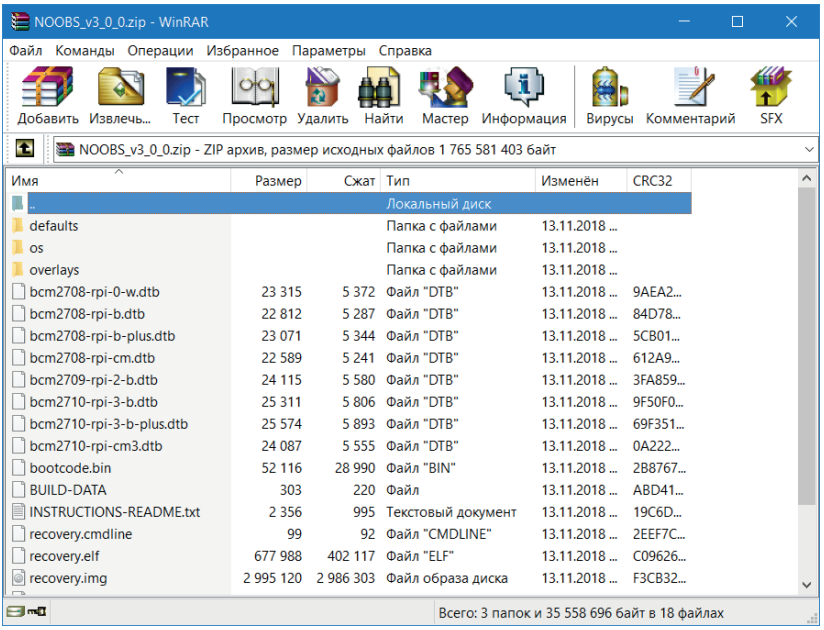


Рис. 1.5. *Окно архиватора с содержимым папки NOOBS*

Как видишь, архив NOOBS состоит из множества файлов и папок. Щелкни мышью на значке **Извлечь...**. На экране появится окно **Путь и параметры извлечения** с открытой вкладкой **Общие**. Здесь ты можешь выбрать папку, в которой будут сохранены распакованные файлы. В правой части вкладки **Общие** щелкни мышью на значке **Рабочий стол**. В поле ввода пути сохранения файла **Путь для извлечения (если не существует, то будет создан)** появится строка, указывающая путь к рабочему столу. Щелкни мышью в конце этой строки и после слова **\Desktop** введи с клавиатуры слово **\NOOBS** (см. рис. 1.6 сверху). Путь сохранения файла выбран. Теперь для запуска распаковки файлов нажми в нижней части вкладки **Общие** кнопку **OK**. Когда архив будет распакован, на рабочем столе появится новая папка NOOBS, в которой будут храниться извлеченные файлы.

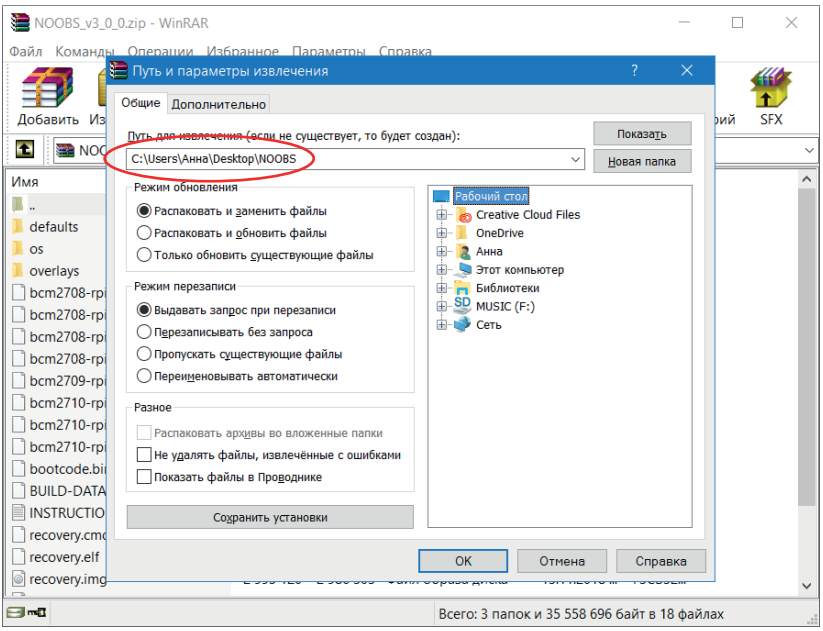


Рис. 1.6. *Выбираем папку для хранения файлов NOOBS*

Форматирование SD-карты

Прежде чем копировать файлы NOOBS на SD-карту, её необходимо отформатировать. Во время этой процедуры вся информация на карте удаляется и появляется новая структура хранения файлов. Форматирование – опасная функция, при которой все файлы, которые хранились на этой SD-карте, уничтожаются безвозвратно. Поэтому лучше,

если рядом с тобой будет сидеть кто-то, кто хорошо разбирается в компьютерах.

Mac

Если у тебя компьютер Mac, воспользуйся **дисковой утилитой** этой операционной системы. Вставь SD-карту в специальный слот (есть не на всех моделях компьютеров). Чтобы открыть поисковую систему **Spotlight**, одновременно нажми клавиши ⌘ и **Пробел**. В строке поиска вводи первые буквы словосочетания «дисковая утилита», пока вся фраза не появится целиком. Дважды щелкни мышью на названии утилиты. На экране появится окно этой программы. В левой части этого окна щелкни мышью на строке с именем твоей SD-карты. В правой верхней части программы щелкни мышью на ярлыке **Стереть**. Выбранная вкладка будет открыта. Выбери из открывающегося списка **Формат** строку **MS-DOS (FAT)**. В нижней части окна нажми кнопку **Удалить** и в появившемся окне подтверди свои действия. SD-карта будет отформатирована и приготовлена к работе. Теперь подготовленную SD-карту нужно из компьютера извлечь. Для этого щелкни правой кнопкой мыши на значке SD-карты и выбери из появившегося контекстного меню команду **Извлечь**. Всё. Извлеки SD-карту из разъёма.

Windows

Если на твоем компьютере установлена операционная система Windows, тебе понадобится специальная программа для форматирования. Зайди на сайт организации SD Association (https://www.sdcard.org/downloads/) и скачай программу SD Memory card Formatter for Windows Download (см. рис. 1.7).

Скачанная программа тоже представляет собой ZIP-архив, который тебе нужно распаковать, а потом установить. Если возникнут сложности, спроси кого-то, кто уже устанавливал эту программу.

Порядок форматирования карты SD следующий.

- Осторожно вставь SD-карту в щель кардридера компьютера. Кардридером называется разъём для чтения-записи SD-карт. Обрати внимание: объём памяти SD-карты должен быть не менее 8 ГБ. Если у тебя микро-SD, понадобится адаптер. Обрати внимание на то, какое название присвоит операционная система вставленной SD-карте. Например, **диск D**.

Рис. 1.7. *Загрузка программы форматирования*

- После запуска программы форматирования **SD Card Formatter** выбери из открывающегося списка **Select card** (Выберите карту) свою SD-карту.

  Если кроме твоей SD-карты в компьютере больше SD-карт нет, имя твоей карты в открывающемся списке **Select card** (Выберите карту) появится автоматически. Если же к компьютеру подключено больше карт, открой список карт **Select card** (Выберите карту) и выбери свою.
- Если захочешь, введи в поле ввода **Volume label** имя, которым ты хочешь эту карту назвать.
- Чтобы запустить процесс форматирования, нажми кнопку **Format**, расположенную в правом нижнем углу окна.

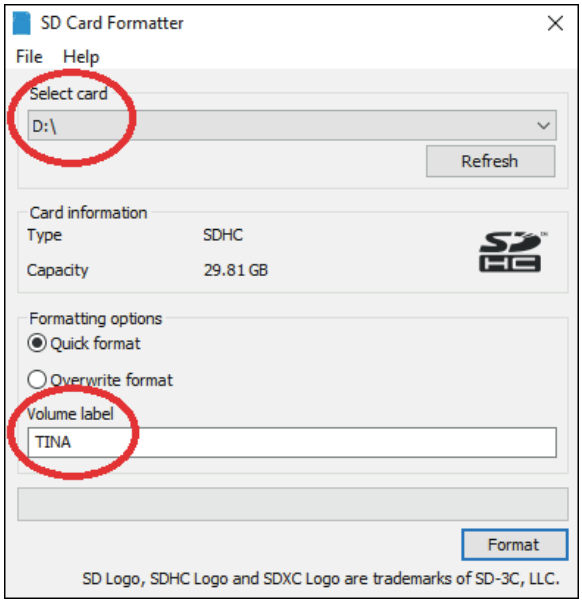


Рис. 1.8. *При форматировании информация с SD-карты сначала удаляется, после чего производится её разметка для установки операционной системы*

Копируем NOOBS

После того как SD-карта подготовлена, скопируй на неё файлы NOOBS. В Windows это выглядит так:

- открой папку **NOOBS** и выдели все файлы, которые в ней находятся. Для этого нажми комбинацию клавиш **Ctrl**+**A** (в данном случае буква A означает *all*, то есть в переводе с английского «все»);
- нажми комбинацию клавиш **Ctrl**+**C**. Все выделенные файлы будут скопированы в оперативную память компьютера (в данном случае буква C означает *copy*, то есть в переводе с английского «копировать»);
- выбери в файловом менеджере свою SD-карту (например, **диск D**) и нажми комбинацию клавиш **Ctrl**+**V**. Всё, что было скопировано в оперативную память компьютера, будет перенесено на SD-карту;
- готово.

В MAC OS принцип копирования точно такой же. Только в MAC OS вместо клавиши **Ctrl** ты будешь пользоваться командными клавишами.

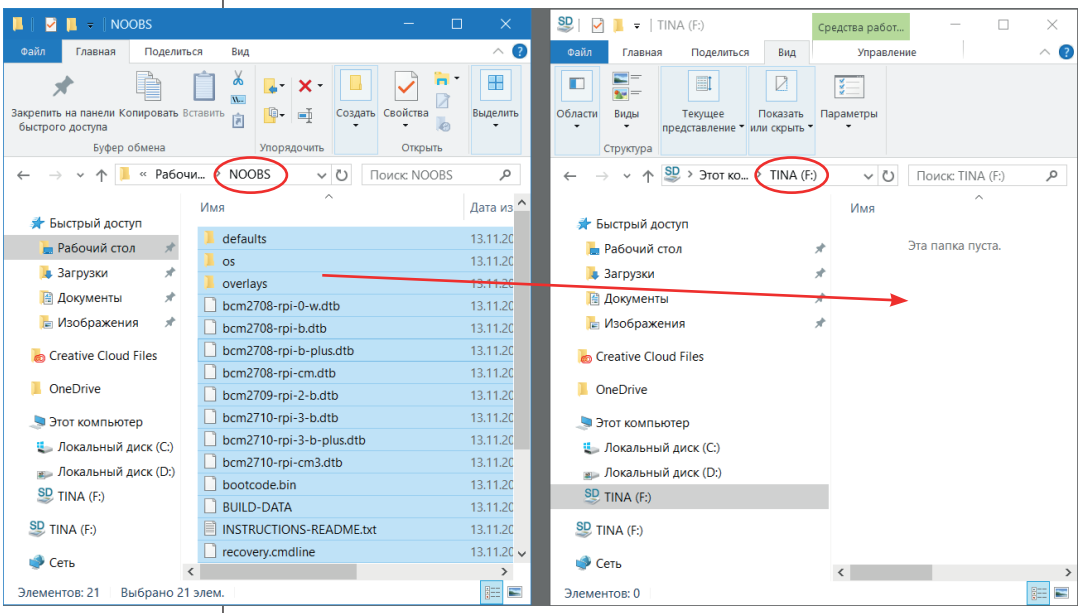


Рис. 1.9. *Содержимое папки NOOBS (обрати внимание: не сама папка!) копируется на SD-карту*

Первый запуск Raspberry Pi

Перед первым запуском нужно к компьютеру подключить все периферийные устройства (монитор, клавиатуру, мышь) и вставить SD-карту в кардридер. Карта в кардридер Raspberry Pi (разъём для SD-карты) вставляется контактами вверх (см. рис. 1.10). Для этого вставь карту в щель кардридера и легко толкай ее пальцем, пока не услышишь щелчок фиксатора. Карта должна выступать всего лишь на несколько миллиметров над краем платы.

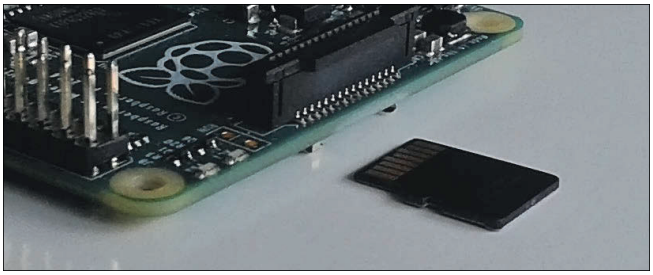


Рис. 1.10. *Вставляем карту микро-SDHC*

Подключи остальные устройства. Клавиатура и мышь присоединяются к компьютеру через USB-разъём, а монитор – через кабель HDMI. В заключение ты подсоединишь в гнездо микро-USB провод от блока питания (см. рис. 1.11). Так как кнопки включения у компьютера нет, Raspberry Pi начнёт свою работу сразу, как только ты включишь блок питания в розетку. Кстати, запуск компьютера также называют *загрузкой*.

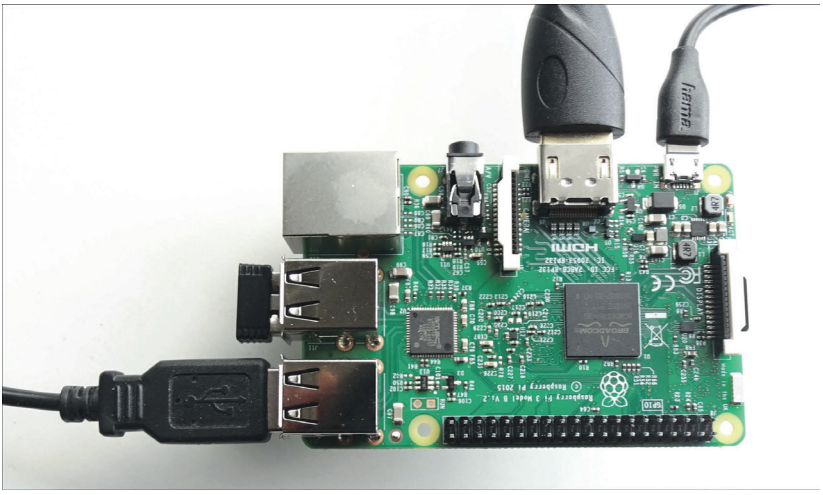


Рис. 1.11. *Подключённые устройства слева направо: кабель для мышки, радиопередатчик для беспроводной клавиатуры, кабель HDMI и кабель микро-USB для электропитания*

Во время первой загрузки появится окно запуска **NOOBS**. С его помощью выбирается, какая операционная система будет установлена. Операционная система Raspbian уже сохранена на SD-карте. В левом верхнем углу щелкни мышью на квадратике в левой части строки **Raspbian (Recommended)**. В выбранном квадратике появится маленький крестик (см. рис. 1.12).

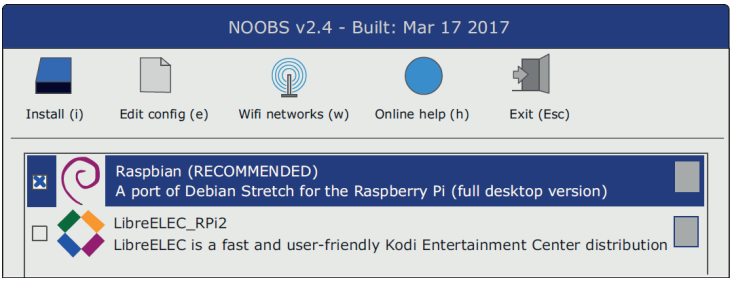


Рис. 1.12. *Окно запуска меню NOOBS*

В нижней части экрана ты можешь выбрать настройки языка. Очень важно, чтобы ты сразу установил русскую раскладку клавиатуры. В нижней части экрана в открывающемся списке **Language** (буква L) по умолчанию ты увидишь строку **English (UK).** Щелкни мышью на маленьком чёрном треугольнике рядом с этой строкой, чтобы появился список языков. Выбери строку **Русский**. Рядом с клавиатурой автоматически появится функция **RU**, с помощью которой устанавливается QWERTY-раскладка.

Что такое QWERTY-раскладка?
В немецкой клавиатуре первые верхние буквы начинаются с Q, W, E, R, T и Z. А в английской и русской раскладках начальными буквами идут Q, W, E, R, T, Y. Отличаются они шестой буквой. В английской раскладке это буква Y.

Для запуска загрузки Raspbian нажми кнопку **Install**. Процесс установки операционной системы займёт некоторое время, поэтому, пока ожидаешь, изучи появляющиеся на экране пояснения к операционной системе.

По окончании загрузки система оповестит об *успешной установке*. Произойдёт перезапуск, и вскоре ты увидишь рабочий стол, примерно такой, как на рис. 1.15. Почему я пишу «примерно»? Дистрибутив Raspberry Pi, который находится в свободном доступе в интернете, обновляется каждые несколько месяцев, поэтому вид рабочего стола может быть другим. Но чаще всего обновления дистрибутива незначительны, и внешний вид рабочего стола вряд ли изменится. Да и основные функции пользовательского интерфейса в последние годы едва ли изменились.

Конфигурация Raspberry Pi

Каждый новый компьютер должен быть настроен по собственному вкусу. Это называется конфигурация. Нажми на кнопку в верхнем левом углу. Появится основное меню операционной системы. Выбери в появившемся меню строку **Настройки** и в появившемся подменю щелкни мышью на строке **Raspberry-Pi-конфигурация** (рис. 1.13). В английском меню **Настройки** переводится как **Preferences**. В следующих разделах я объясню, как выполнить несколько важных настроек, которые ты сможешь сделать самостоятельно.

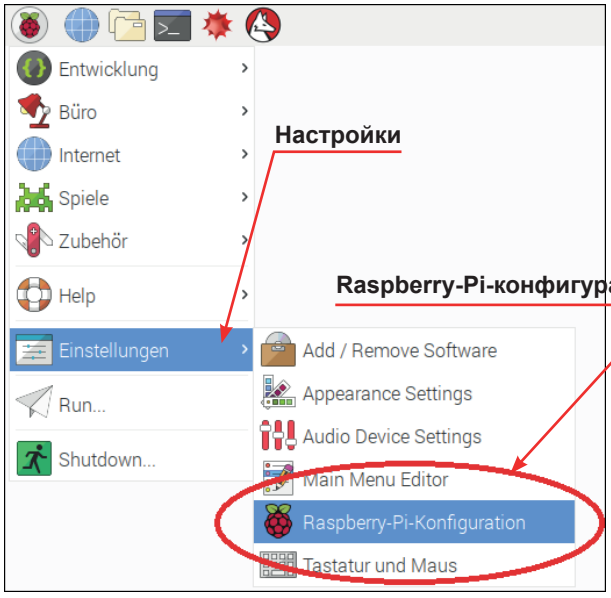


Рис. 1.13. *Запуск конфигурации*

Что делать, если клавиатура работает не так, как надо?
Raspberry Pi – очень надёжный компьютер. Крайне редко случается так, что мышка или клавиатура перестают работать. В таких случаях достаточно выдернуть USB-шнур из разъёма, а потом вновь его вставить. Это безопасное действие ты можешь выполнить при включенном компьютере.
Если же это не поможет, то, возможно, клавиатура или мышь требует слишком много электроэнергии. Тогда тебе придётся подключить активный USB-разветвитель с собственным блоком питания. Разветвитель, или, по-другому, хаб (от англ. *hub* – разветвление, развилка, узел), имеет один USB-вход и множество USB-выходов (портов). USB-вход с помощью кабеля подключается к USB-разъёму Raspberry Pi. Клавиатура, которая потребляет большое количество энергии, подключается к порту (USB-выходу) хаба.

Изменение пароля

Raspberry Pi с операционной системой Linux может работать со многими пользователями. Это значит, что твоим компьютером могут воспользоваться другие люди. У каждого пользователя есть своё имя и пароль. Чтобы войти в систему, сначала нужно зарегистрироваться и ввести свои данные. Возможно, твой Raspberry Pi будешь использовать

только ты. Но и в этом случае ты сам тоже можешь работать под несколькими именами. Почти как актёр, исполняющий разные роли.

На твоём компьютере, помимо предустановленной операционной системы, уже настроена функция системного администратора под именем root, и один «обычный» пользователь с именем pi и паролем paspberry.

Кто такой системный администратор?
Системный администратор (root) имеет особые права и несёт особую ответственность. Он может создавать новых пользователей и иметь доступ к файлам, способным изменить работу операционной системы. Определённые действия (например, установку нового программного обеспечения) может внести только администратор. Представь себя системным администратором, этаким хозяином большого многоквартирного дома.

Большую часть времени ты будешь работать под именем pi. Одним из профессиональных навыков пользования компьютером является изменение пароля.

Выбери в программе **Raspberry-Pi-конфигурация** вкладку **Система** (см. рис. 1.14).

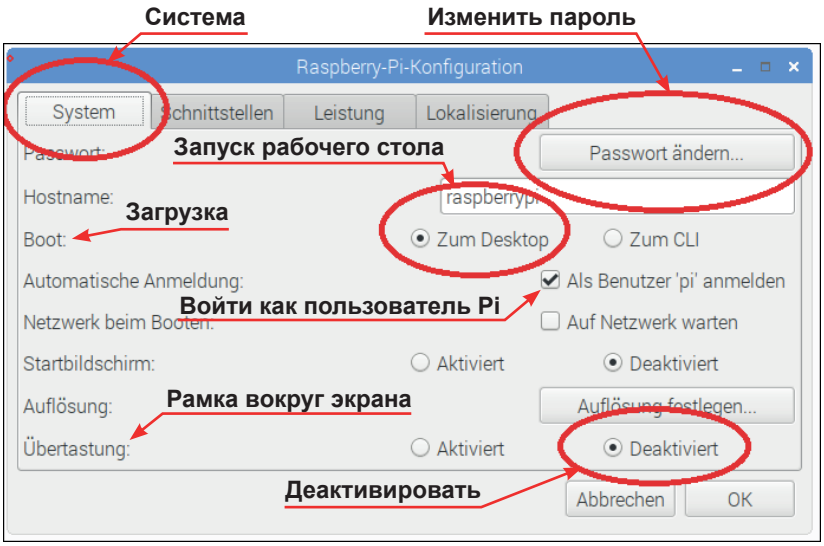


Рис. 1.14. *Изменение пароля и отключение рамки вокруг экрана*

В правом верхнем углу окна программы нажми кнопку **Изменить пароль**. В появившемся диалоговом окне в поле ввода **Ввести новый пароль** введи своё кодовое слово (па-

роль) и дважды щёлкни на строке **Подтвердить новый пароль**. Затем нажми кнопку **OK**.

Сделать рабочий стол доступным

Для большинства проектов, представленных в этой книге, ты будешь использовать Raspberry Pi в качестве стационарного компьютера с монитором, клавиатурой и мышкой. Но гораздо практичнее, чтобы после запуска системы ты авторизовался не сразу, а после запуска графического интерфейса Raspberry Pi.

Для этого сделай так, как показано на рис. 1.14. Установи флажок левее строки **Войти как пользователь Pi**. В строке **Boot** (Загрузка) установи переключатель **Zum Desktop** (Запуск рабочего стола).

Как убрать чёрные рамки

Если у тебя HD-монитор, то вокруг рабочего стола можно заметить чёрную рамку. Это значит, что площадь монитора используется не полностью. Если тебе это мешает, тогда придётся отключить отображение рамки вокруг экрана. В строке **Рамка вокруг экрана** установи переключатель **Деактивировать** (см. рис. 1.14). После перезапуска компьютера чёрная рамка исчезнет.

Рабочий стол

В основном ты будешь работать с Raspberry Pi на своем рабочем столе (Desktop) (см. рис. 1.15). В переводе с английского *desktop* означает «поверхность стола». На самом деле он и выглядит как рабочий стол, на котором тут и там лежат всякие нужные вещи: книги, бумаги, письменные принадлежности, ластик. Если во время конфигурации выбрать опцию **Boot** (Загрузка) **Zum Desctop** (Запуск рабочего стола), после загрузки компьютера на экране сразу же появляется графический интерфейс. Иначе после каждого запуска компьютера рабочий стол будет запускаться из командной строки с помощью команды

```
$ sudo startx
```

Более подробную информацию о том, что такое командная строка, ты узнаешь из раздела «Работаем с файловым менеджером» на стр. 41.

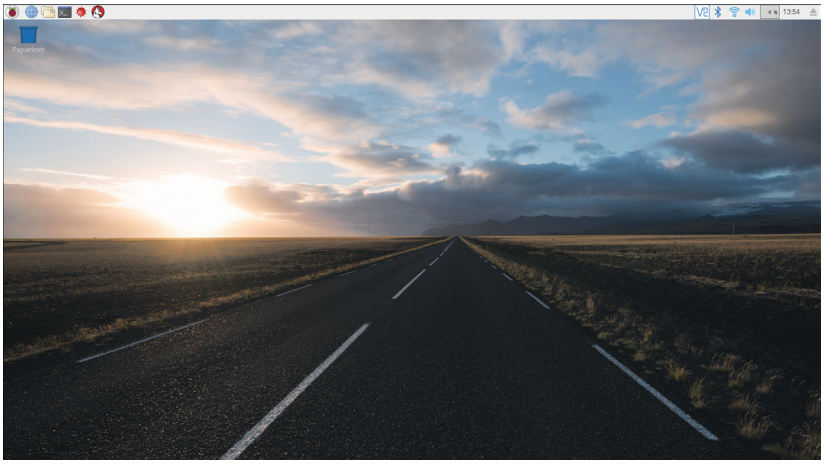


Рис. 1.15. *Рабочий стол операционной системы Raspbian (2017)*

В верхнем левом углу рабочего стола находится панель быстрого запуска (**Application Launch Bar**). Функции этой панели быстрого запуска очень важны.

Рис. 1.16. *Верхний левый угол рабочего стола*

В верхнем левом углу ты найдёшь следующие символы.

1. Меню **Пуск**, позволяющее быстро получить доступ к установленным на этом компьютере программам.
2. Интернет-браузер **Chromium** (Хромиум) для работы в интернете.
3. Файловый менеджер **PCManFM**. С его помощью можно получить доступ к файлам в различных папках (так же, как Explorer в операционной системе Windows).
4. **LXTerminal**. Это приложение служит для ввода команд в командной строке операционной системы Unix.
5. **MATHEMATICA** (Математика) компании Wolfram Research. Это приложение позволит тебе выполнять математические действия и рисовать математические графики. А теперь информация для специалистов. Кнопка с изображением овчарки, расположенная правее кнопки **MATHEMATICA** (Математика), относится к той же математической программе. Более подробную информацию об этом ты найдешь в разделе «А теперь посчитаем» на стр. 39.

Выключаем компьютер

Как уже говорилось, Raspberry Pi не имеет кнопки включения. Ты можешь его отключить, вытащив из гнезда USB-шнур. Затем отсоединить его от питания. Хотя такое выключение может повредить содержимое SD-карты. Но такое случается очень редко. В случае повреждения SD-карты все данные будут утеряны, и тебе придётся заново форматировать SD-карту, копировать файлы NOOBS и вновь запускать операционную систему. Чтобы не повредить данные, сохраненные на SD-карте, завершай работу компьютера по-другому. Щелкни мышью на кнопке и выбери команду **Shutdown**. В появившемся диалоговом окне выбери команду **Завершить** и нажми кнопку **OK**.

Как изменить фон рабочего стола

Возможно, картинка на рабочем столе тебе не нравится. Чтобы её изменить и залить рабочий стол каким-то одним цветом, действуй следующим образом:

- щёлкни правой кнопкой мыши в любом месте рабочего стола. На экране появится контекстное меню. Выбери в появившемся контекстном меню команду **Appearance Settings** (Настройки рабочего стола). Появится окно как на рис. 1.17;

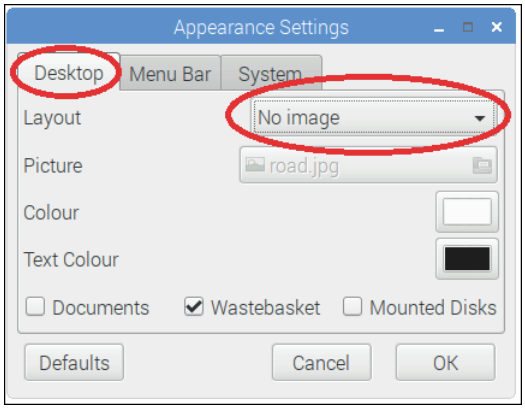


Рис. 1.17. *Меняем фон рабочего стола*

- выбери вкладку **Desktop**;
- открой открывающийся список **Layout**. Для этого щелкни мышью на маленьком треугольнике в правой части этого списка;
- выбери строку **No image** (Нет изображения);

- нажми кнопку **Colour** и выбери цвет, который тебе понравится. Затем нажми кнопку **OK**.

Как подключить Raspberry Pi к интернету

Установка программы с помощью NOOBS происходит без подключения твоего Raspberry Pi к интернету, т. е. находится в режиме «офлайн». Но для многих операций интернет-соединение необходимо. Raspberry Pi имеет разъём RJ45, к которому подключается сетевой кабель. При этом к твоему компьютеру будет подходить еще один кабель. Чтобы уменьшить количество подключаемых кабелей, следует использовать Wi-Fi. Новая модель Raspberry3 уже имеет встроенный Wi-Fi-модуль, который и обеспечит беспроводную интернет-связь. Для старых моделей придётся приобрести адаптер (стоимость его примерно 750 руб.), который ты вставишь в свободный USB-разъём. Для подключения к интернету через Wi-Fi выполни следующие действия.

- В правом верхнем углу рабочего стола нажми кнопку и запусти программу конфигурации компьютера. Появится список доступных интернет-соединений, к которым можно подключиться прямо сейчас.
- Найди в этом списке название вашей домашней сети и щёлкни по ней мышкой, чтобы появилось окно, как показано на рис. 1.18.

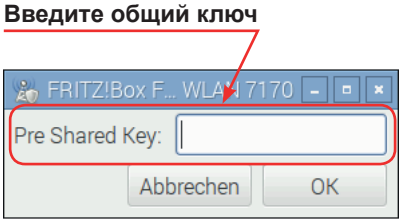


Рис. 1.18. *Подключение Rpi к интернету*

- Возможно, твоя сеть использует стандарт безопасности WPA2, что в переводе с английского расшифровывается как *Wi-Fi Protected Access 2*, а по-русски – «защищённый доступ Wi-Fi». Это означает, что все данные, которые проходят через эту сеть, надёжно защищены. Однако если кто-то чужой получит доступ к паролю ключа безопасности (Pre Shared Key), то все данные могут быть

вскрыты и прочитаны посторонними лицами. Если пароль ты не знаешь, поинтересуйся у родителей. Затем введи этот пароль в поле ввода **Введите общий ключ** и нажми кнопку **OK**.

Если компьютер подключился к сети интернет, иконка подключения к сети изменится на вот такой значок  .

А теперь посчитаем

Увлекаешься ли ты математикой и математическим моделированием? Нет? Ну тогда дальше не стоит и читать, можешь просто пропустить этот раздел. На твоём RPi установлена программа Mathematica, разработчиком которой является Стивен Вольфрам (он же основатель компании «Wolfram Research»). Mathematica поможет тебе решать уравнения (компьютерная алгебра) и строить графики функций. Интересно? Тогда вот тебе для начала маленький пример.

Чтобы запустить программу «Математика», щелкни мышью на кнопке  , расположенной в верхней левой части рабочего стола. Откроется окно блокнота (текстовый редактор), в котором ты можешь вписать тест и математические функции. Функция определяется порядком расчёта, например: $f(x) = x^2$. Графики функций – это график в системе координат.

Щелкни мышью на кнопке  и выбери команду **Insert**, в появившемся подменю выбери команду **Free-Form Input**. В окне документа появится маленькое поле ввода со знаком равенства. В это поле ввода нужно вписать инструкцию на английском. Например:

```
plot x**2 with dashed grid lines
```

По-русски это значит следующее: «нарисуй *x* больше 2 с пунктирными линиями». Выбери в меню **Evaluation** команду **Evaluate Notebook**. После того как введенная инструкция будет выполнена, в системе координат появится график функций с пунктирными линями. На рис. 1.19 этот пример хорошо проиллюстрирован.

После того как ты введешь инструкцию, ниже появится похожая на математическую формулу новая версия инструкции со скобками.

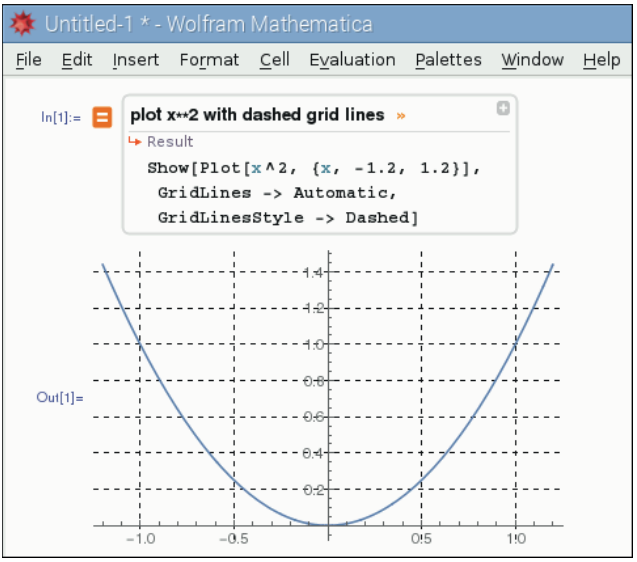


Рис. 1.19. *График функции f(x) = x^2*

На диаграмме с горизонтальными и вертикальными пунктирными линиями увидишь график функций, где значение *x* изменяется от –1,2 до +1,2. Взгляни на диаграмму и найди пересечение графика с горизонтальной линией, на которой указаны оба числа.

Если щёлкнуть мышью на формуле, то введенная ранее в верхней строчке инструкция исчезнет. Теперь ты формулу можешь редактировать. Например, изменить диапазон значения *x* (рис. 1.20).

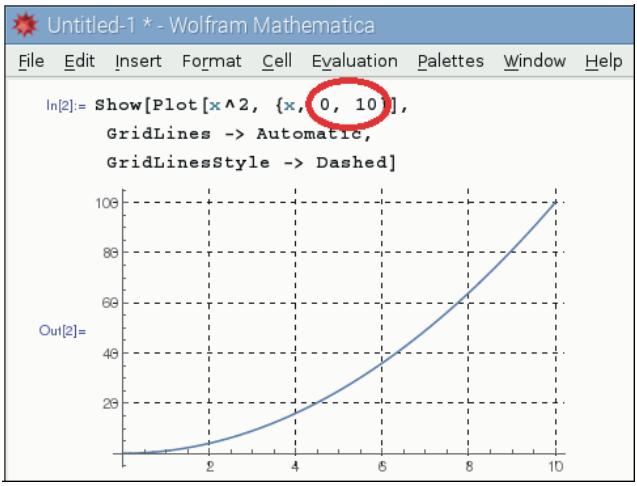


Рис. 1.20. *График диапазона значений x от 0 до 10*

Работаем с файловым менеджером

Все данные хранятся в папках. Вместо слова «папка» часто используется слово «каталог» или «директория» (от анг. *directory*). С помощью файлового менеджера ты сможешь просматривать содержимое папок, открывать их, переносить данные из одной папки в другую, данные копировать или удалять. Щёлкни мышью в правом верхнем углу на кнопке файлового менеджера . Файловый менеджер будет запущен (рис. 1.21).

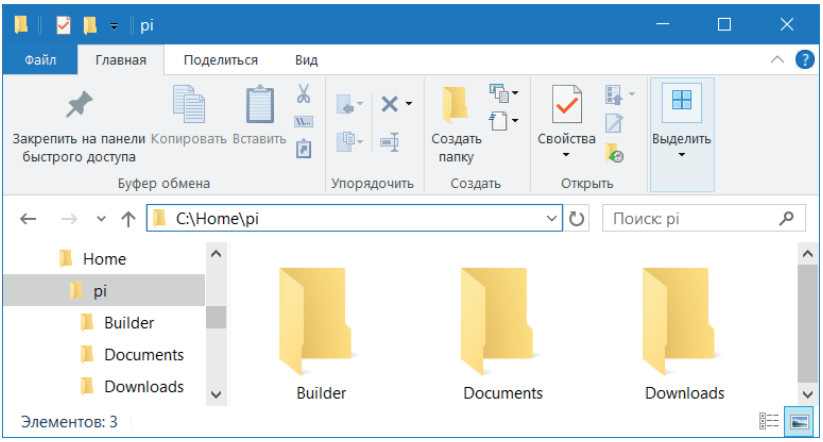


Рис. 1.21. *Так выглядит файловый менеджер*

У системы Linux пользователей, как правило, много. Учитывая, что единственным пользователем Raspberry Pi, или по-другому «юзером» (от англ. *User*), являешься ты, в верхней части окна файлового менеджера ты увидишь свое имя пользователя – **pi**. Ниже будет показано содержимое твоего каталога пользователя **pi**. Если в системе будет зарегистрирован еще один пользователь, то имя пользователя у него будет другим. У него тоже будет свой собственный пользовательский каталог с его личными данными (по-английски *home directory*). Каждый такой каталог носит имя своего пользователя, а его папка в виде домика изображена в верхней части файлового менеджера, слева от адресной строки (см. рис. 1.21). Всякий раз, когда ты жмёшь на значок этой папки, то попадаешь в свою директорию.

Каждый каталог и каждый файл имеет своё **Имя** и свой **Путь**. Заметь: **Путь** и **Имя** – это не одно и то же. Путь директории – это адрес, по которому находится данная папка.

Это как твой адрес проживания с указанием страны, города, улицы и номера дома. Указывая его при создании файла или папки, можно выбрать точное место сохранения файла или папки. Этот адрес указывается в адресной строке файлового менеджера (выделен кружком на рис. 1.21). Например, путь *\home\pi* означает, что папка *pi* расположена в папке *Home*.

Что такое «дерево файлов» в системе Linux

Каталог – это своего рода хранилище папок, в котором могут содержаться файлы, а также другие папки (подпапки). Каталог может стать подпапкой для каталога более высокого уровня. Таким образом все каталоги связаны между собой. Эта структура носит название *файлового дерева,* но дерево это – перевёрнутое. Корни находятся на уровне листьев, а листья – на уровне корней. Все компьютеры Linux имеют одинаковую структуру файлового дерева. Корень обозначен символом /. Это высший каталог в иерархии, и он единственный из всех, который не является подпапкой других каталогов.

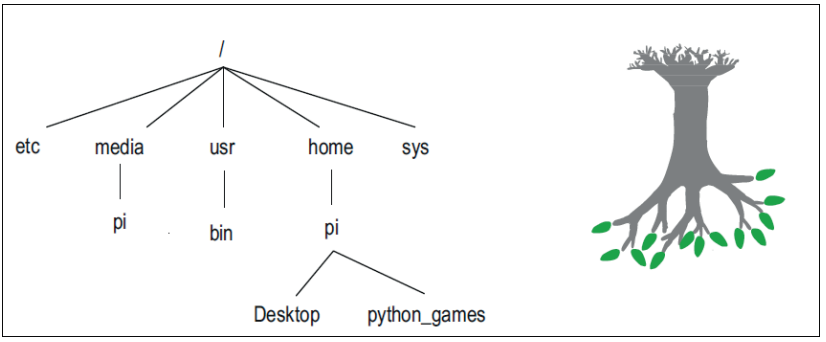


Рис. 1.22. *Файловое дерево Linux (упрощённая форма)*

С помощью файлового менеджера ты можешь изучить каталоги своего Raspberry Pi. На рис. 1.22 представлены самые важные каталоги твоего компьютера.

/etc – здесь размещены файлы конфигурации. Эти файлы содержат настройки для различных пользовательских программ, установленных на твоём компьютере.

/media – данный каталог устанавливает соединение с внешней памятью (например, USB-накопителем). Если подключить флеш-карту к Raspberry Pi, то в подкаталоге Pi появится новый каталог с файлами. Говорят, что это монтируемый каталог. То есть в нем показываются подключаемые

к компьютеру устройства типа флеш-карты. Более подробно об этом в следующем разделе.

/usr – это самая обширная область системы. Здесь хранится большая часть установленных программ. Например, игры располагаются в каталоге */usr/games*. Другие программные файлы находятся в каталоге *usr/bin*.

/home – «домашний» каталог содержит каталоги всех пользователей. В Raspberry Pi изначально установлен только один пользователь (**pi**), поэтому здесь ты увидишь папку *pi*.

/sys – в этом разделе собраны все системные файлы и данные. Например, файлы температурного датчика. Подробнее это мы рассмотрим в главе 9.

Путь показывает, где именно в файловом дереве хранится каталог. Это своего рода описание направления, которое начинается с корня. Между названиями каталогов, через которые проходит путь, всегда стоит значок *косая черта /*.

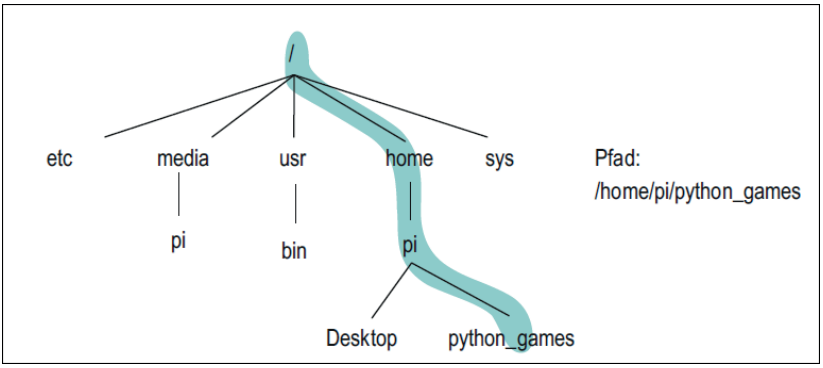


Рис. 1.23. *Путь в файловом дереве*

Как скопировать файлы на флеш-карту

Возможно, тебе понадобится скопировать файлы с твоего компьютера Raspberry Pi на флешку. Как это сделать? В качестве примера мы возьмём файл изображения из каталога *python_games*. Данный файл уже сохранен в каталоге *python_games* по умолчанию.

- Открой файловый менеджер, если он ещё не открыт. Для этого щёлкни мышью в правом верхнем углу рабочего стола на кнопке .
- Вставь флешку в USB-разъём Raspberry Pi. На экране появится окно с надписью **Флеш-карта установлена**. Если же флешка была вставлена ранее, подобная надпись не появится. В этом случае вытащи флешку из разъёма и через пару секунд снова вставь в разъём.

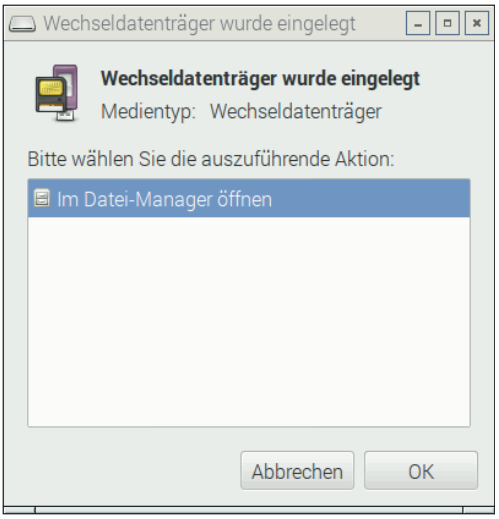


Рис. 1.24. *Открываем флешку в менеджере задач*

- Чтобы открыть новое окно файлового менеджера, нажми в появившемся окне кнопку **OK**. В адресной строке ты увидишь путь, который начинается с */media/pi/*... (на рис. 1.25 этот путь обведён кружком). Имя твоей флешки будет показано после обозначения пути */media/pi/*. Здесь система Linux будет представлять флешку как каталог. Ниже адресной строки вы увидите поле, в котором показаны папки и файлы, хранящиеся в открытом каталоге.

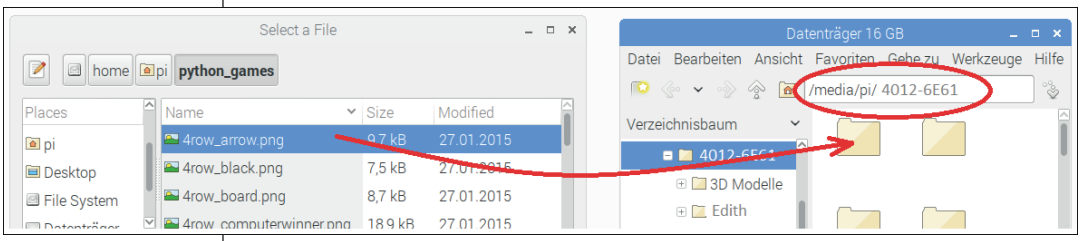


Рис. 1.25. *Копирование файла на флешку*

- Дважды щелкни мышью на папку, в которой ты хочешь сохранить картинку. Это может быть любая папка. Возможно, на твоей флешке уже есть папка **Изображения**.
- Открой новое окно менеджера задач, щёлкнув в левом нижнем углу .
- В новом окне, которое ты открыл, ты увидишь каталог пользователя. Щелкни мышью на подкаталоге *python_games*.

- Щёлкни мышью на значке загружаемой картинки (например, *4row_arrow.png*) и нажми комбинацию клавиш **Ctrl**+**C**.
- Щёлкни мышью в другом окне диспетчера файлов, в котором открыта твоя флешка, и дважды щелкни мышью на целевой папке (папке, в которую будет скопирован файл), чтобы открыть ее.
- Нажми комбинацию клавиш **Ctrl**+**V**.
- Готово. Файл скопирован.

Естественно, ты точно так же можешь скопировать файлы и в обратном порядке, т. е. с флешки на свой компьютер. Так, например, можно фотографию, сделанную в отпуске, установить в качестве фона на рабочий стол.

Монтирование

Как только ты подключишь свою флешку к Raspberry Pi, Linux зафиксирует её содержимое в файловом дереве. Это называется *монтированием файловой системы* (от англ. *to mount* – монтировать, закреплять). Имя флешки станет именем каталога. Флешка закреплена за подкаталогом, обозначенным как */media/pi*. Если эту карту памяти снова извлечь, то ее содержимое опять исчезнет из дерева файлов.

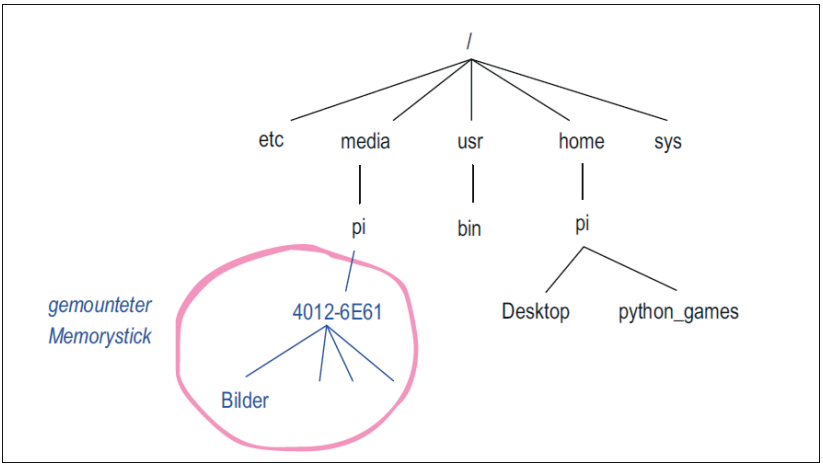


Рис. 1.26. *Файловое дерево с закреплённой флешкой*

Ввод Unix-команд в приложении LXTerminal

В Linux терминал **LXTerminal** является одним из самых важных приложений. Это своего рода *консоль*, которая понадобится тебе для ввода Unix-команд. Установленная на твоем компьютере операционная система Raspbian – это один из многих вариантов операционной системы Unix – родоначальницы таких операционных систем, как Windows и Linux.

Одну из важных команд мы попробуем прямо сейчас. Остальные ты можешь попутно изучить в следующих главах книги.

Навигация по файловому дереву

Запусти терминал **LXTerminal**, щёлкнув мышью на кнопке  , расположенной в левом верхнем углу рабочего стола. На экране появится командная строка:

```
pi@raspberrypi ~ $
```

В начале командной строки вы увидите приглашение для ввода команд. В приведенном примере приглашение выглядит как `pi@raspberrypi ~ $` и состоит из имени пользователя **pi** и названия установленной операционной системы, в конце которого ты снова увидишь имя пользователя: **raspberrypi**. Далее следует значок ~ («тильда»). Команды вводятся после значка доллара `$`. Командная строка – это своеобразный диалог с компьютером. Как только ты введешь команду и нажмёшь клавишу **Enter** ↵, компьютер начнёт выполнять введенную команду и что-то отвечать. В этом разделе мы попробуем ввести те команды, которые имеют отношение к управлению файлами.

Для того чтобы в книге отображение командной строки не было таким длинным, первую часть приглашения (`pi@raspberrypi ~`) мы показывать не будем и ограничимся знаком доллара `$`. Но при вводе команд в командную строку левую часть приглашения нужно вводить. Сокращать в книге приглашение – это общепринятое действие.

Что такое рабочий каталог?

Дерево файлов компьютера можно представить в виде здания с этажами и помещениями. «Помещения» очень важны: ты всегда находишься в одном из них и можешь пере-

мещаться из одного в другое. То пространство, в котором ты находишься прямо сейчас, называется *рабочий каталог* (от англ. *working directory*, *work* – работа).

Как упорядочить содержимое каталога

После того как ты открыл окно **LXTerminal**, ты «находишься» в пользовательском каталоге. То есть путь к текущему рабочему каталогу на данный момент будет */home/pi*.

С помощью команды `ls` (от англ. *list* – список) ты сможешь просмотреть содержимое текущего рабочего каталога. Введи в командную строку команду `ls` и нажми клавишу **Enter** ↵.

После нажатия данной клавиши нижевведенной команды появится ответ от компьютера. Данный ответ будет состоять из нескольких файлов и подкаталогов, сохранённых в текущем каталоге. Это будет несколько файлов и папок, выделенных голубым цветом.

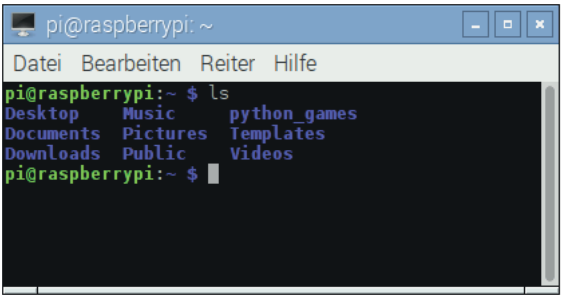


Рис. 1.27. *Содержимое рабочего каталога, вызванное командой* `ls`

Как изменить рабочий каталог

С помощью команды `cd` (от англ. *change directory* – смена каталога) ты можешь перейти в другой каталог. Это то же самое, если ты перейдешь в другую комнату. Выполни следующую команду:

```
$ cd python_games
```

(Обрати внимание, что в тексте книги мы убрали левую часть приглашения `pi@raspberrypi ~`. При вводе команды тебе, скорее всего, левую часть приглашения придется ввести.)

Итак, после исполнения команды `pi@raspberrypi ~ $ cd python_games` текущим каталогом станет папка *python_games*. Название текущего каталога будет показано в строке заголовка в верхней части терминала **LXTerminal**.

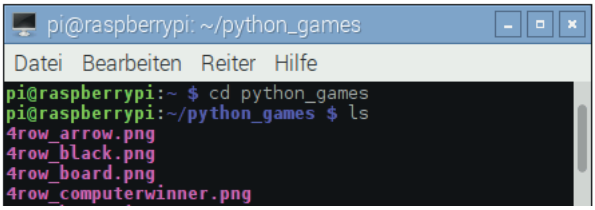


Рис. 1.28. *Переход в каталог /home/pi/python_games*

Если снова ввести команду `ls`, то ты увидишь содержимое каталога *python_games*.

Зеленым цветом будет выделена левая часть приглашения, введенные команды – белым, имя каталога – синим, а содержимое открытой в данный момент папки – фиолетовым цветом.

Чтобы снова вернуться в каталог более высокого уровня, т. е. «домой» */home/pi*, введи команду:

```
$ cd ..
```

Автозаполнение

Автозаполнение командной строки – это очень удобная функция, облегчающая ввод команд в командную строку.

В «домашнем» каталоге введи следующую команду:

```
$ cd p
```

и нажми клавишу табуляции ⇆. Поскольку в рабочем каталоге указан только один-единственный подкаталог, чьё имя начинается с буквы «p», то название этого каталога заполнится автоматически. В командной строке появится команда:

```
$ cd python_games/
```

Чтобы выполнить эту команду, нажми клавишу **Enter** ↵. В результате этих действий ты снова окажешься в каталоге *python_games*.

Тебе, наверное, уже не терпится запустить одну из игр. Эти игры программируются с помощью программы Python. Ты можешь запустить игру Python, используя функцию автозаполнения.

Введи следующую команду:

```
$ python sl
```

и нажми клавишу [⇆]. Так как есть только один файл, имя которого начинается с «ls», то вскоре в командной строке ты увидишь полную команду:

```
$ python slidepuzzle.py
```

Нажми клавишу **Enter** [↵] и запусти игру:

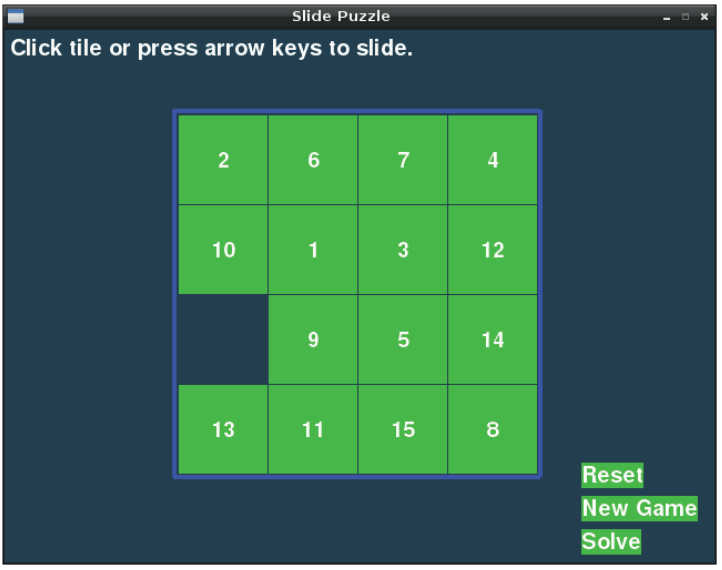


Рис. 1.29. *Игру slidepuzzle можно запустить из командной строки*

Как вернуть команду

Чтобы вернуть недавно введенную команду, достаточно нажать клавишу [↑]. Нажав на клавишу со стрелкой вверх два раза, ты отобразишь в командной строке позапрошлую команду. Эта функция очень удобна при вводе большого количества команд.

Управление пользователями

На компьютере Raspberry Pi изначально настроены два пользователя: администратор под логином *root* и «обычный» пользователь под логином *pi*. Администратор, в отличие от обычного юзера, имеет множество прав, поэтому носит статус привилегированного пользователя. Он может, к примеру, добавить или удалить новых пользователей и поменять пароль для них. Но ты, как правило, зарегистрирован под именем *pi*.

Как сменить пароль

Ты можешь поменять свой пароль Unix прямо из командной строки. Введи команду:

```
$ passwd
```

Система из соображений безопасности предложит тебе сначала ввести старый пароль. Затем нужно дважды ввести новый.

Также из соображений безопасности пароль в консоли отображаться не будет. Кодовое слово не может быть слишком коротким и, кроме букв, должно включать в себя цифры.

Но что делать, если ты находишься под именем пользователя *pi*, но забыл свой пароль? В этом случае ты не сможешь авторизоваться и получить доступ к своим файлам. Поменять пароль другого пользователя может только администратор. Тем не менее даже на правах «обычного» пользователя ты всё-таки можешь ввести команду от лица администратора, если перед командой введёшь значение `sudo` (do us superuser):

```
$ sudo passwd pi
Введите новый пароль Unix:
Введите новый пароль Unix ещё раз:
```

Кстати, эту команду `sudo` может использовать не каждый пользователь. Для этого потребуется специальное разрешение. Такое разрешение есть у пользователя *pi*.

Создание нового пользователя

Возможно, иногда на твоем Raspberry Pi будет работать кто-то из твоих друзей или твои родители. Для работы им понадобятся собственные пользовательские имена. С помощью команды `adduser` администратор может создать нового пользователя. Если ты авторизован в системе под именем *pi*, то перед командой введи слово `sudo`. Создадим нового пользователя и присвоим ему имя *machael*:

```
$ sudo adduser Michael
```

После ввода данной команды компьютер создаст нового пользователя с его собственным каталогом */home/Michael*. Теперь при входе в систему компьютер запросит у данного пользователя его имя и пароль. Важный момент: новый

пользователь не сможет воспользоваться командой sudo. Для таких действий ему не хватает прав администратора, так что никакие изменения в системе он внести не сможет.

Активируем систему

Обновление системы проводится не только в начале работы на Raspberry Pi, но и при установке нового программного обеспечения. Дважды щелкни мышью на кнопке  , расположенной в левом верхнем углу рабочего стола. Приложение **LXTerminal** будет открыто. Позаботься о том, чтобы компьютер был подключён к интернету. Введи в командную строку команду `$ sudo apt-get update` и нажми клавишу **Enter** ↵.

```
$ sudo apt-get update
```

Первое слово (sudo) означает, что эту команду ты запустил от лица системного администратора. Если слово «sudo» не ввести, система выдаст ошибку. Во время обновления данные будут изменены, а права к доступу и изменению этих данных есть только у администратора (не у пользователя *pi*). APT означает *Advanced Packaging Tool*, т. е. *Программа для установки, удаления и обновления программных пакетов*. Следующий пункт – запуск обновления. Оба действия необходимы для того, чтобы операционная система активировала обновление.

После того как программа данные обновит, введи команду:

```
$ sudo apt-get upgrade
```

Обрати внимание, что процесс обновления может занять некоторое время (иногда до часа). Поэтому терпеливо жди, пока приглашение не появится вновь:

```
pi@raspberrypi ~ $
```

Вопросы

1. Что означают команды `cd` и `ls`?
2. Выясни, что произойдёт в случае, если ввести команду `cd` без указания на каталог?
3. Какую Unix-команду ты используешь, если захочешь запустить программу на правах администратора (например, Leafpad)?

4. В чём отличие файла от каталога?
5. Что произойдет, если при вводе Unix-команды в консоли нажать клавишу ↑?
6. Что произойдет, если при вводе Unix-команды на консоли нажать клавишу табуляции ⇆?
7. Почему у администратора прав больше, чем у «обычного» пользователя?
8. С помощью каких команд ты сможешь обновить операционную систему?

Задание: установка фона для рабочего стола

Открой в своём домашнем каталоге папку **Изображения**. Скачай любую картинку из интернета и сохрани её в этой папке. Сделай её фоном рабочего стола.

Совет

Как скачать картинку из интернета. В верхнем левом углу рабочего стола щелкни мышью на кнопке . Будет запущен браузер Хромиум. Найди в интернете картинку, которая тебе нравится. Она должна быть достаточно большой, чтобы заполнить всю поверхность экрана (это как минимум 1920×1080 пикселей). Щелкни правой кнопкой мыши на выбранном изображении. На экране появится контекстное меню. Выбери из появившегося контекстного меню команду **Сохранить картинку как...** и сохрани картинку в папке */home/pi/Bilder* (рис. 1.30).

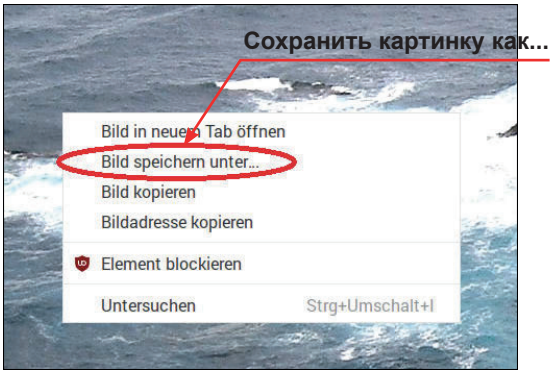


Рис. 1.30. *Кликнув правой кнопкой мыши по картинке, можно вызвать контекстное меню*

Ответы на вопросы

1. Команда `cd`: смена текущего каталога (*change directory*), команда `ls`: вывод содержимого текущего каталога (*list*).
2. Если не указать название каталога, команда `cd` вернётся в домашнюю папку (*/home/pi*).
3. С помощью команды `sudo` можно запустить программу на правах администратора.
4. Файл – это одна картинка, или один текстовый документ, или одно музыкальное произведение. Один каталог (папка) может содержать в себе много разных файлов. Кроме файлов, в каталоге могут храниться другие подкаталоги (вложенные папки).
5. С помощью клавиши ↑ повторяется последняя введенная в командную строку команда.
6. Клавишу табуляции ⇆ используют для автоматического ввода команд в командную строку. Начальные буквы вводимых команд нужно набрать на клавиатуре и нажать клавишу табуляции. Эта функция значительно экономит время при вводе команд.
7. Определённые команды, такие как `halt` или `reboot`, может ввести только администратор. Если ты входишь в систему под логином *pi*, ты не сможешь выключить Raspberry, введя команду `halt`. Тебе необходимо воспользоваться комбинацией `sudo halt`. Кроме того, файлы конфигурации (например, автоматический запуск файлов) могут быть изменены только администратором.
8. С помощью команд `sudo apt-get update` и `sudo apt-get upgrade` можно обновить операционную систему.

Решение задачи: установка фона рабочего стола

Итак, ты скачал изображение и сохранил его в папке */home/pi/Bilder*. Теперь, чтобы сделать его фоновым изображением рабочего стола, выполни следующие действия. Щелкни правой кнопкой мыши на свободном месте рабочего стола. На экране появится контекстное меню. Выбери из появившегося контекстного меню команду **Appearance Settings** (Настройки рабочего стола). Появится окно, как на рис. 1.31. Открой в правой части строки **Layout** список и выбери из него команду **Fit image onto screen**.

На экране тут же появится картинка, которая сейчас находится на рабочем столе. В правой части строки **Picture** щелкни мышью на поле ввода с названием текущего изображения. Откроется окно **Выбрать фон рабочего стола**. В этом окне перейди в папку */home/pi/Bilder* (для этого в адресной строке щелкни мышью на pi, а затем в правом списке щелкни мышью на строке **Изображения**). Щёлкни мышью на названии файла картинки, которую ты хочешь поместить на рабочий стол, и нажми кнопку **OK**, расположенную в правом нижнем углу.

Теперь у тебя новый фон рабочего стола.

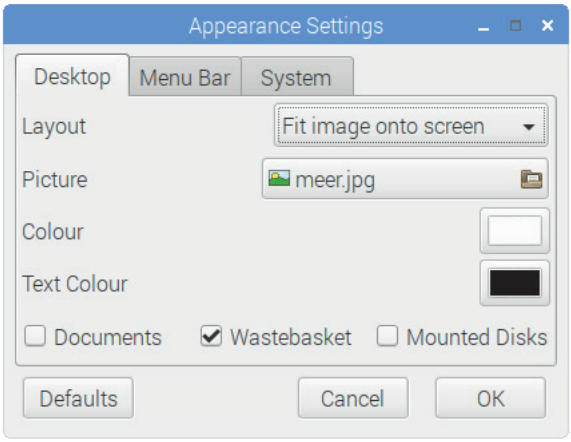
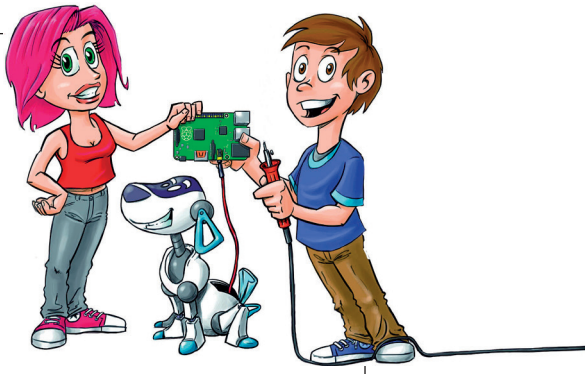


Рис. 1.31. *Настройка фона рабочего стола*



2 Создаём медиацентр и киоск (интерактивный терминал)

Теперь ты знаешь, как за один сеанс, даже без подключения клавиатуры, с помощью другого компьютера запустить Raspberry Pi. Если к компьютеру подключить колонки, у тебя получится музыкальный центр, управлять которым ты сможешь на расстоянии с помощью своего телефона.

Raspberry Pi поможет тебе настроить информационную систему общего доступа. С её помощью пользователи смогут лишь бродить в интернете, не более. Такое устройство называется «интерактивным терминалом» или «киоском».

В заключение ты установишь на вторую SD-карту другую операционную систему и сможешь создать на своём Raspberry Pi медиацентр или домашний кинотеатр. Ты сможешь слушать на нём музыку, смотреть фильмы, онлайн-трансляции и ТВ-программы.

Как слушать музыку с помощью MOC

Да, на Raspberry Pi можно слушать музыку. Для этого тебе придётся подключить к гнезду звуковой карты компьютера диаметром 3,5 мм штекер от наушников или колонки. Если ты подключил к компьютеру через HDMI-разъём монитор, то звук будет передаваться по этому же HDMI-кабелю.

Аудиоформаты: MP3 и OGG

Музыкальный формат MP3 сохраняет музыку таким хитрым образом, что, невзирая на высокое качество, файл занимает на диске совсем немного места. Формат OGG предназначен для сжатия и сохранения мультимедийных файлов. В этом формате могут одновременно сохраняться не только аудио- и видео-, но и текстовые файлы.

Для проигрывания музыки потребуется установить на компьютере аудиоплеер. Для Raspberry Pi наилучшим решением будет установка программы MOC (Music on Console), которую разработал Дэмиан Пьетрес (Damian Pietres). Этот аудиоплеер может работать как с MP3, так и с OGG-файлами. У MOC простой интерфейс, а играет плеер – как и следует из его названия – с консоли. Чтобы установить плеер с терминала **LXTerminal**, введите следующую команду:

```
$ sudo apt-get install moc
```

Для запуска плеера введите команду

```
$ mocp
```

После ввода этой команды в окне программы **LXTerminal** появится простой интерфейс аудиоплеера MOC, как на рис. 2.1.

В левой части интерфейса ты увидишь дерево каталогов Linux. Справа находится плей-лист: список следующих одна за другой музыкальных композиций, которые будет проигрывать MOC. Сейчас этот список пуст.

Аудиоплеер управляется с помощью клавиш. Кнопка табуляции ⇆ поможет тебе переходить из левой части интерфейса в правую часть. А клавиши со стрелками, нажимаемые вместе с клавишей **Enter** ↵, позволят тебе «путе-

шествовать» по каталогам в левой части. Чтобы проиграть одну мелодию, выбери её с помощью клавиш со стрелками и нажми **Enter** ↵.

Если нажать клавишу **q**, то интерфейс MOC закроется, а музыка продолжит играть в фоновом режиме. Чтобы завершить работу проигрывателя, работающего в фоновом режиме, сначала введи в терминале команду `mocp`. В окне терминала появится панель проигрывателя. Далее останови воспроизведение. Для этого нажми клавишу **s**. А теперь можно и завершить работу самой программы. Нажми клавишу **q**. Проигрыватель завершит работу.

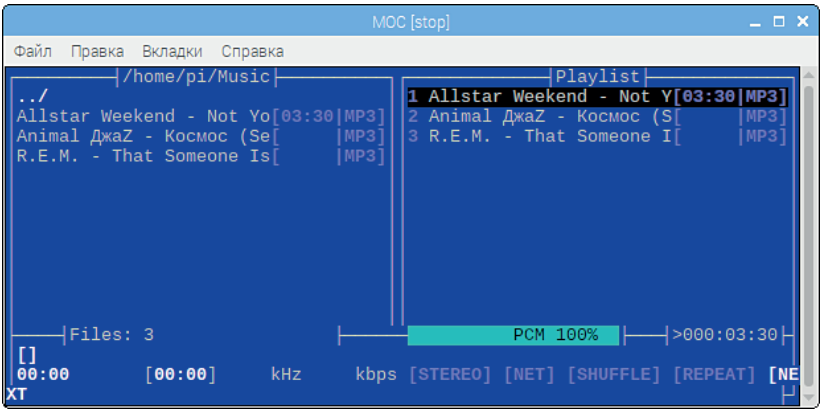


Рис. 2.1. *Интерфейс аудиоплеера MOC*

Итак, проигрыватель установлен и запускается. Если ты закрыл проигрыватель, введи в терминале команду `mocp`. Теперь составь плей-лист (плей-лист – очередность воспроизведения композиций). Для этого подключи флешку с музыкальными файлами к Raspberry Pi. Далее следует найти USB-карту. Это устройство должно быть в каталоге */media/pi/*. Выбери звуковой файл, который ты хочешь послушать, и нажми клавишу **a** (*add* в переводе с англ. означает «добавить»). Композиция будет добавлена в плей-лист.

Таблица 2.1 объяснит и другие функции плеера. Чтобы увидеть полный список опций, нажми клавишу **h** (*help* – это «помощь, справка»).

Таблица 2.1. Некоторые функции аудиоплеера MOC

Клавиша	Значение
a	Добавить музыкальный файл в плей-лист
d	Удалить выбранный трек из плей-листа
h	Вызвать справку, посмотреть весь список функций

Таблица 2.1 (окончание)

Клавиша	Значение
n	Выбрать следующий трек
<, >	Регулятор громкости
p	Пауза
q	Закрыть интерфейс с возможностью слушать музыку дальше
⇧ + q	Завершить программу
s	Остановить воспроизведение

Как изменить громкость? Используем микшер

На компьютер Rpi уже установлена программа для настройки и регулировки звука Advanced Linux Sound Architecture (сокращённо ALSA). Программа является частью операционной системы и отвечает за воспроизведение музыки. Настроить громкость можно с помощью команды `alsamixer`:

```
$ alsamixer
```

Когда ты её введёшь, в окне **LXTerminal** появится маленькая интерактивная программа с вертикальной строкой. С помощью клавиш ↑ и ↓ громкость музыки регулируется в бóльшую или меньшую сторону. Значение строки в середине показывает громкость на текущий момент.

Обрати внимание, у тебя в окне микшера может появиться несколько регуляторов. Выбрать регулятор громкости **PCM** ты можешь с помощью клавиш ← и →.

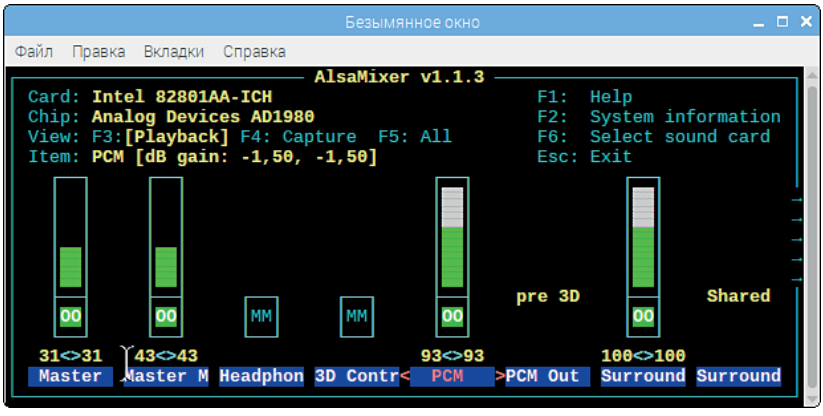


Рис. 2.2. *Настраиваем громкость с помощью микшера*

«Безголовый» Raspberry

Ты можешь подключиться к своему Raspberry Pi и удаленно, через сеть интернет. В этом случае к компьютеру подключать монитор и клавиатуру совсем не обязательно. Но компьютер должен быть включён и подключён к локальной сети. Это называется «безголовым» рабочим режимом (от анг. *headless* – «без головы», «без управления»). В таком режиме ты можешь подключиться к своему Raspberry Pi и авторизоваться с любого компьютера, подключённого к локальной сети, и запустить нужную тебе программу. Для этого нужно соблюсти несколько правил:

- тебе необходимо знать свой IP-адрес Raspberry Pi;
- активировать режим SSH (Secure Shell);
- на компьютере, с которого ты будешь управлять своим компьютером, тебе понадобится SSH-клиент, например PuTTY, с помощью которого ты создашь соединение с Raspberry Pi.

Что такое IP-адрес?
Компьютеры постоянно передают файлы по кабелю или беспроводной связи. Роутер отвечает за то, чтобы каждый пакет данных нашёл своего получателя. Принцип работы такой же, как и на почте:

- каждый компьютер имеет свой уникальный адрес, который известен другому компьютеру. Этот адрес состоит из набора цифр, так называемого IP-номера (IP – это интернет-протокол). Номер состоит из чисел от 0 до 255, разделённых точками. Например: 192.168.178.56;
- пакет данных содержится в так называемом заголовке IP-номере получателя, который напоминает наклейку с адресом на посылке. Так компьютер получателя определяет, для него предназначено это послание или нет.

Какой IP-адрес у моего Raspberry Pi?

Открой **LXTerminal** и введи следующую команду:

```
$ ifconfig
```

Имя программы означает *интерфейс конфигуратора* (*interface configurator*). Эта команда выводит на экран характеристики активного интернет-соединения. Это может быть как беспроводное соединение, так и подключение по локальной сети.

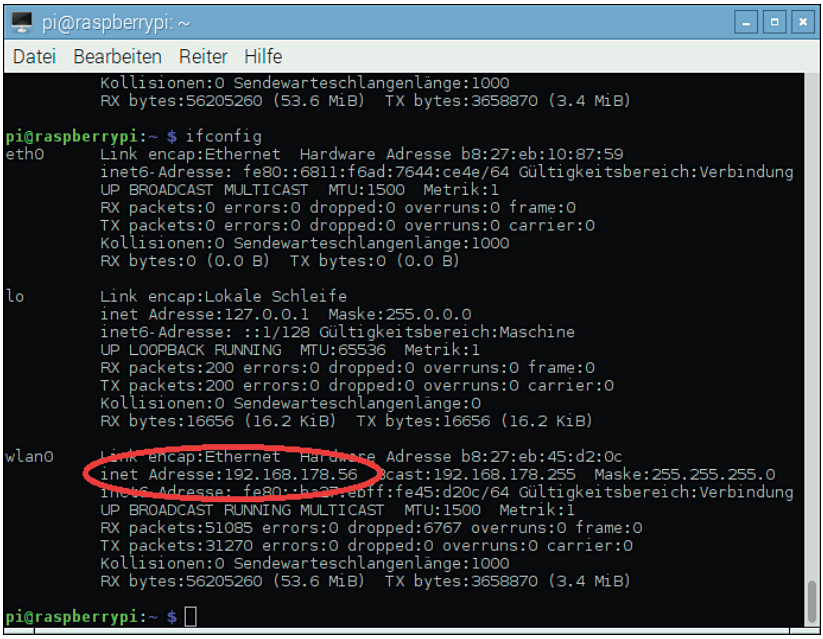


Рис. 2.3. *Поиск IP-адреса с помощью команды* `ifconfig`

В разделе о WLAN0 ты и найдёшь IP-адрес своего Raspberry Pi. Он находится рядом с адресом inet.

Удалённый доступ с помощью PuTTy

Прежде чем ты сможешь подключиться к своему Raspberry Pi с другого компьютера, тебе следует запустить на своём компьютере службу SSH (Secure Shell). В левом верхнем углу экрана щёлкни мышью на кнопке и выбери из появившегося меню команду **Настройки конфигурации Raspberry-Pi**.

В появившемся окне **Raspberry Pi Configuration** щёлкни мышью на ярлыке вкладки **Interfaces** и убедись, что опция **SSH** активирована (переключатель установлен в положение **Enable**). Закрой окно **Raspberry-Pi Configuration**, нажав кнопку **OK** (рис. 2.4).

SSH (*Secure Shell*) – это программа (сетевой протокол) для удалённого управления компьютером. SSH обеспечивает безопасное соединение с Raspberry Pi, так как все файлы передаются в зашифрованном виде. Никто не сможет их «подсмотреть» или «прослушать».

Существует целый список бесплатных SSH-клиентов, предназначенных для установки на смартфоны. С помощью такой программы ты сможешь связаться с Raspberry Pi по

локальной сети с другого компьютера (или смартфона). Самым популярным SSH-клиентом является программа **PuTTY**. Это бесплатная программа, и её легко найти в интернете. Ты сможешь её установить в течение нескольких секунд.

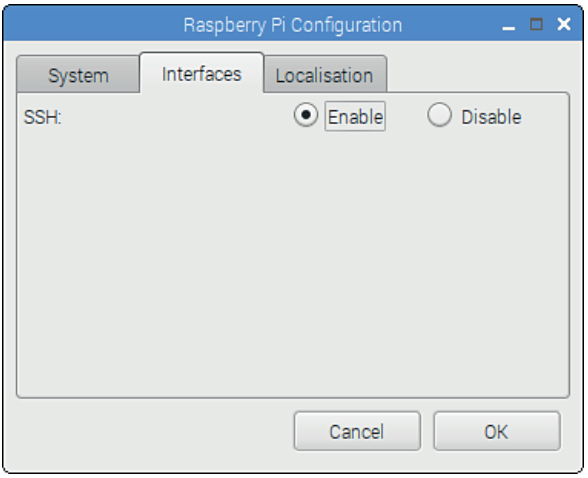


Рис. 2.4. *SSH активирован*

Чтобы подключиться с помощью **PuTTY** к своему Raspberry Pi, выполни следующие действия (процесс подключения с помощью других программ такой же).

Запусти **PuTTY**. Введи в поле ввода **Host Name (or IP-address)** (Имя хоста или IP-адрес) IP-адрес своего Raspberry Pi. Проследи, чтобы в поле ввода **Port** (Порт) было введено значение **22**, а в группе переключателей **Connection Type** (Тип соединения) переключатель стоял в положении **SSH**. Нажми кнопку **Save** (Сохранить). Итак, все данные для соединения введены (рис. 2.5).

Запусти сессию, щёлкнув мышью на кнопке **Open**.

Удалённый компьютер подключится к твоему компьютеру, откроется окно терминала **pi@raspberrypi:**, и ты увидишь приглашение в командную строку:

```
login as:
```

Введи свой логин, нажми клавишу **Enter** ↵ и в следующей строке введи свой пароль. Ещё раз нажми клавишу **Enter** ↵ – и ты авторизован! После этого увидишь уже хорошо знакомое тебе приглашение в командную строку `$`.

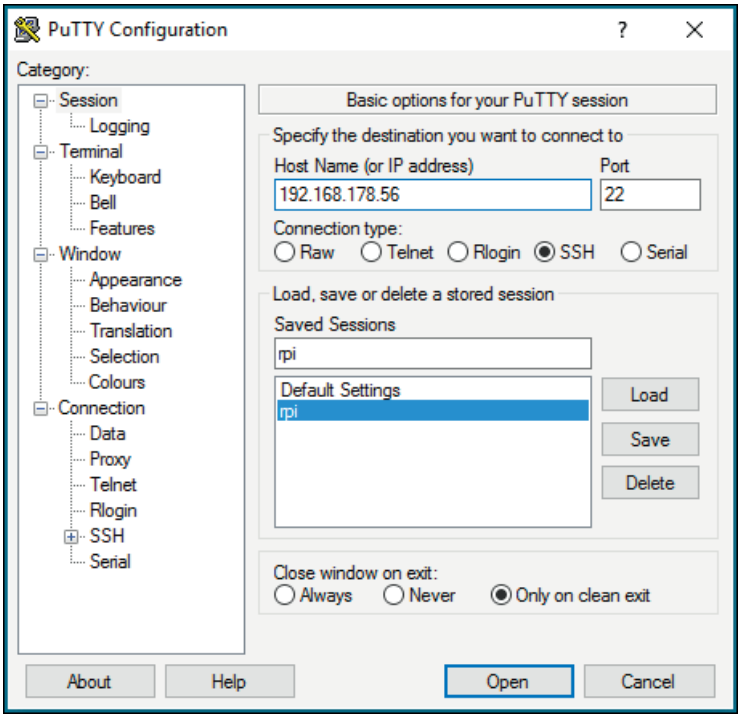


Рис. 2.5. *Запускаем программу PuTTY*

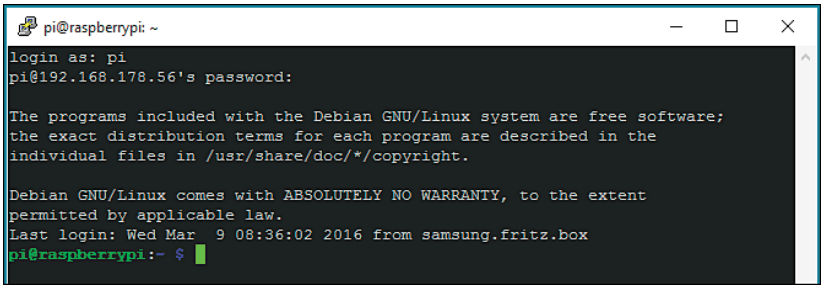


Рис. 2.6. *Так выглядит окно SSH-терминала PuTTY под управлением Windows 10*

Правда, в этом терминале ты не увидишь никакого графического интерфейса, а все команды требуется вводить в командной строке. Но при этом здесь есть куча всего остального. Ты можешь, к примеру, проходить через каталоги или запускать маленький текстовый блокнот nano. Зачастую это очень удобно, т. к. при желании ты сможешь быстро изменить файлы конфигурации.

Сессия завершается с помощью команды

```
$ exit
```

Операционные системы Lunux и OS X

Удалённо подключиться к Raspberry Pi можно и с компьютера под управлением операционной системы Mac или любого другого компьютера (такого как Raspberry Pi) под управлением Linux. Для этого нужно только ввести в консоль команду:

```
$ ssh pi@IP-Addres
```

Вместо `pi@` вписываешь IP-адрес своего Raspberry Pi.

Система удалённого доступа к компьютеру (VNC)

Новая модель Raspberry Pi 3 уже оснащена встроенным сервером VNC. С помощью этого сервера, мышки и клавиатуры ты можешь передать всё содержимое экрана твоего Raspberry Pi на экран другого компьютера. С помощью сервера VNC можно работать с Raspberry Pi, используя ноутбук. При этом ты, сидя у ноутбука или настольного компьютера, который ты подключил к своему Raspberry Pi, управляешь своим компьютером так, как если бы ты сидел прямо перед Raspberry Pi. Для этого тебе нужно выполнить два действия:

- активировать сервер VNC на своём Raspberry. Затем запустить программу конфигурации **Пуск** ⇨ **Настройки** ⇨ **Raspberry-Pi-Конфигурация**. Выбери вкладку **Порт** и активируй VNC;
- установи на удалённом компьютере программу **VNC-Viewer**. Наиболее подходящий вариант есть у компании RealVNC (скачать бесплатно можно по ссылке http://realvnc.com/download/viewer).

После установки в первую очередь создай для Raspberry Pi собственное соединение. Как это сделать – было рассказано ранее (SSH-сервер). Далее запусти на удалённом компьютере программу VNC-Viewer и выбери команду меню **File** ⇨ **New connection...** (Файл ⇨ Новое соединение...).

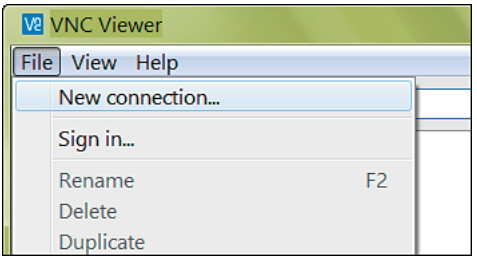


Рис. 2.7. *Создаём новое соединение (программа VNC-Viewer)*

Откроется окно **Properties** (Свойства), в котором тебе нужно ввести IP-адрес твоего Raspberry Pi. IP-адрес вводится в поле ввода **VNC Server**. Ты также можешь дать имя интернет-соединению. Имя вводится в поле ввода **Name** (Имя). Теперь жми **OK**.

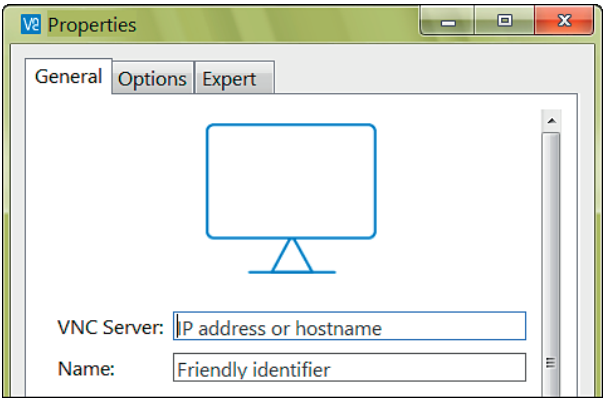


Рис. 2.8. *Самое важное свойство соединения – это IP-адрес Raspberry Pi*

После всех действий на панели VNC-Viewers появится картинка для соединения. Если ты дважды щёлкнешь по этой картинке, то увидишь контакт, созданный для Raspberry Pi.

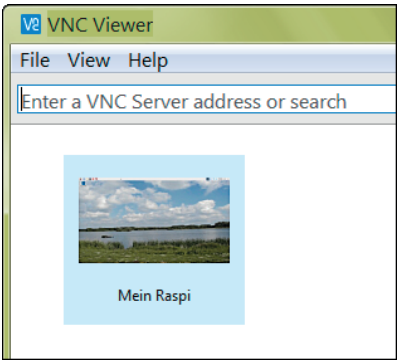


Рис. 2.9. *Изображение рабочего стола, подключенного Raspberry Pi*

После авторизации (например, через установленный логин *pi* и пароль *raspberry*) откроется большое окно с графической панелью Raspberry Pi. В этом окне ты можешь управлять своим Raspberry Pi с другого компьютера. Обрати внимание: в середине верхнего края окна ты увидишь неприметную тонкую «чёрточку».

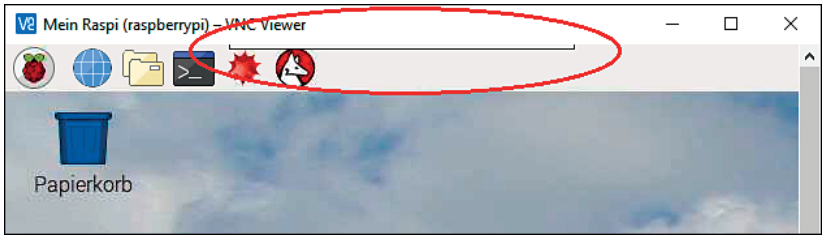


Рис. 2.10. *Закрытое диалоговое окно в верхнем поле*

Если на эту область навести курсор, откроется маленькое диалоговое окно программы Viewers, с помощью которого ты можешь включить или выключить полноэкранный режим.

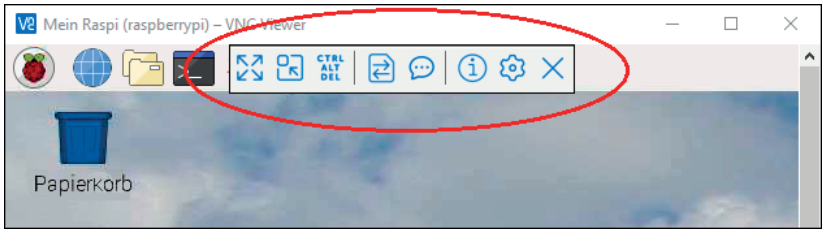


Рис. 2.11. *Окно с настройками программы VNC-Viewer*

Примечание к разрешению экрана
Если твой Raspberry Pi включён в локальную сеть, а к монитору не подключён, будет использовано минимальное разрешение экрана 720×480 пикселей. Это значит, что в окне VNC-Viewer ты увидишь сравнительно маленький рабочий стол RPi. Это, конечно, неудобно. Разрешение в 720×480 пикселей задаётся предварительными настройками. Если тебе нужна более качественная картинка с большим размером, то разрешение можно изменить. Для этого открой на своём Raspberry Pi программу конфигураций (**Старт** ⇨ **Настройки** ⇨ **Raspberry-Pi-Конфигурация**) и открой вкладку **Система**. Нажми на кнопку **Задать разрешение** и выбери удобный для себя формат.

Проект 1. Музыкальный центр с дистанционным управлением

Из своего Raspberry Pi ты можешь сделать музыкальный центр и управлять им с мобильного телефона. Для этого нужно выполнить несколько простых действий.

Подключи колонки к аналоговому аудиовыходу с помощью кабеля с разъёмом 3,5 мм. Колонки будут динамиком с усилителем. Если Raspberry Pi подключён через беспроводное соединение, компьютеру не нужны ни клавиатура, ни монитор, ни мышка. Для прослушивания музыки ты можешь использовать флешку или любой другой внешний носитель с записанными музыкальными файлами.

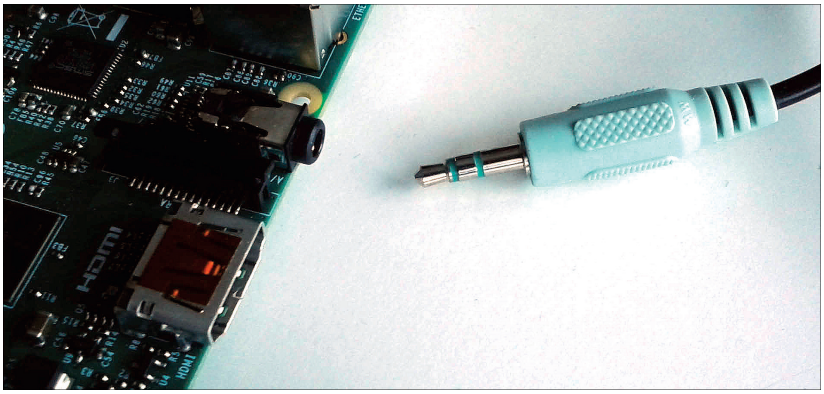


Рис. 2.12. *Подключаем колонки к аналоговому аудиовыходу*

Теперь, подключив к компьютеру микро-USB-разъём подключённого к сети блока питания, запусти Raspberry Pi.

На мобильном телефоне (или другом компьютере), который ты хочешь использовать для управления Raspberry Pi с установленным ранее SSH-клиентом, запусти эту программу удалённого доступа, подключись и авторизуйся в Raspberry Pi (см. предыдущий раздел).

Проследи за тем, чтобы звук шёл через аналоговый разъём для наушников (а не через HDMI). Делаешь это с помощью такой вот команды:

```
$ amixer cset numid=3 1
```

Обрати внимание: цифры 3 и 1 в команде пишутся через пробел. Последняя цифра показывает, какой канал ты используешь: 0 – автоматический выбор, 1 – аналоговый, 2 – HDMI.

Amixer – это служебная программа системы ALSA, которая уже установлена на Rpi по умолчанию.

С помощью команды

```
$ mocp
```

ты запустишь музыкальную программу MOC (см. раздел «Как слушать музыку с помощью MOC», стр. 56). Ты можешь отредактировать свой плей-лист или воспроизвести музыку.

С помощью клавиши **q** (маленький регистр) ты можешь выйти из MOC, оставив играть музыку в фоновом режиме. А введя команду

```
$ alsamixer
```

ты запускаешь программу, с помощью которой можно отрегулировать громкость воспроизведения. Для регулирования громкости используются кнопки ↑ и ↓ (подробнее в разделе «Как слушать музыку с помощью MOC», стр. 56).

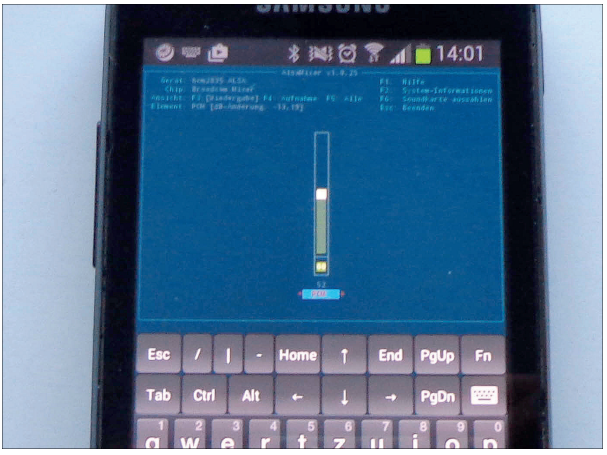


Рис. 2.13. *Программу alsamixer можно запустить прямо с телефона*

Проект 2. Создание интерактивного терминала

Интерактивный терминал, или, как его ещё называют, киоск, – это компьютер с монитором, который находится в общественных местах и доступен для всех желающих. Однако у него не так много функций. Такие терминалы есть в музеях, на вокзалах, в школах и университетах. Через него можно получить информацию о выставках, транспортном сообщении или актуальных мероприятиях. Наш второй проект как раз будет посвящён созданию такого терминала через Raspberry Pi. Подключаем мышку и монитор. Как только Raspberry Pi начнёт свою работу, автоматически запустится браузер, открыв определённую веб-стра-

ницу. Щёлкнув мышкой по гиперссылке, можно попасть на сторонние сайты, но других действий произвести нельзя. Нет возможности ввести адрес с помощью клавиатуры, закрыть браузер или вызвать другую программу.

Использование Chromium в терминале

Запустить Хромиум можно, нажав вверху символ  . Также для запуска можно использовать **LXTerminal**. Запусти терминал, введи в командную строку следующую команду, а затем нажми **Enter** ↵

```
$ chromium-browser --noerrdialogs --kiosk http://www.wdr.de
```

Что же означают отдельные части этой команды?

`chromium-browser` – это собственно имя браузера, который мы и будем использовать.

`--noerdialog` – это параметр. Он отвечает за то, чтобы во время работы браузера не появлялись сообщения об ошибке. В режиме киоска такие уведомления можно не открывать.

`--kiosk` – ещё один параметр. Помогает браузеру работать в режиме терминала.

В конце строки вводится адрес сайта, который браузер должен показать в первую очередь. В режиме терминала окно открывается в полноэкранном формате. В полноэкранном режиме отсутствует адресная строка, панель с открывающимися меню. Также отсутствует полоса прокрутки, с помощью которой можно изменять предложенный размер кадра.

В полноэкранном режиме ты не можешь завершить работу Chromium. Чтобы выйти из полноэкранного режима, нажми клавишу **F1**. Если работа браузера не завершится, воспользуйся следующим способом: сначала нажми комбинацию клавиш **Ctrl**+**N**, чтобы открыть новое окно. В правом верхнем углу нового окна найди кнопку с тремя вертикальными точками  и нажми её. Выбери в нижней части появившегося меню команду **Exit** (Завершить). Chromium завершит свою работу. Все окна будут закрыты. Если тебе нужно закрыть браузер в режиме киоска, воспользуйся клавиатурой. Если клавиатура отсутствует, значит, у тебя самый настоящий интерактивный терминал.

Автозапуск

Ты можешь настроить компьютер так, чтобы при включении Raspberry Pi все нужные программы запускались авто-

матически. Например, ты можешь сделать так, чтобы браузер в режиме киоска запускался автоматически. Всё, что тебе нужно, – это просто вписать соответствующую команду в специальном текстовом файле с именем *autostart*. Порядок действий такой.

Нажми кнопку . Откроется файловый менеджер. Он покажет тебе содержимое твоего пользовательского каталога */home/pi*. Но сам файл *autostart* прячется в скрытом каталоге с именем *.config*. Выбери команду меню **Вид** ⇨ **Показать скрытые файлы**. Все скрытые ранее папки и файлы станут видимыми.

Скрытые каталоги и папки
Папки и каталоги, чьё имя начинается с точки, например .config, обычно в окне файлового менеджера не отображаются, потому что в этих папках хранятся системные файлы, которые случайно можно повредить. Чтобы такие файлы и папки сделать видимыми, выберите команду меню **Вид** ⇨ **Показать скрытые файлы**.

При помощи мышки переместись в каталог */home/pi/.config/1xsession/LXDE-pi*.

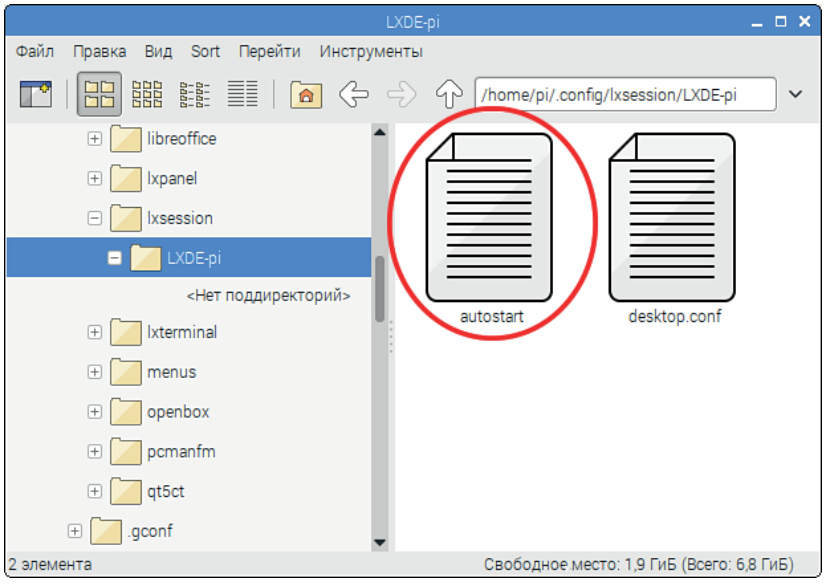


Рис. 2.14. *Ищем файл autostart*

Здесь ты найдёшь файл *autostart*. Дважды щёлкни мышью по значку этого файла. Будет запущен Leafpad, и ты увидишь содержимое этого файла. Текст состоит из трёх команд:

```
@1 xpanel -profile LXDE
@pcmanfm -desktop -profile LXDE
@xscreensaver -no- splash
```

Эти строчки менять нельзя. Просто допиши ниже этих строчек собственную команду, например:

```
@chromium-browser --noerrdianlogs --kiosk http://www.wdr.de
```

Любая команда текста *autostart* начинается значком @. А уже после него в строке вводится нормальная Linux-команда.

Впиши собственную команду, сохрани документ, закрой редактор, закрой файловый менеджер и с помощью команды основного меню **Shutdown** ⇨ **Reboot** (Завершение работы ⇨ Перезагрузить) перезапусти Raspberry Pi. Если ты всё сделал правильно, то при новом старте Raspberry Pi браузер будет запущен автоматически, и ты увидишь главную страницу WDR в полноэкранном режиме. Если же клавиатура у тебя отсутствует, значит, у тебя получился настоящий интерактивный терминал.

Удалённое отключение
Для отключения Raspberry Pi нужно отключить от электропитания. Это ты тоже можешь сделать удалённо. Запусти с другого компьютера в своей домашней сети программу PuTTY или другой SSH-клиент и авторизуйся в Raspberry Pi (см. раздел «Устанавливаем OSMC», стр. 72). Далее введи в командную строку терминала команду:

```
$ sudo halt
```

Raspberry Pi аккуратно отсоединится от сети.

Обработка с помощью Nano и файла autostart

Режим терминала не позволит тебе закрыть браузер и попасть в графический интерфейс. То есть в этом режиме ты не можешь использовать Rpi в качестве чего-то другого. Чтобы этот режим отменить, выполни следующие действия.

Авторизуйся с помощью PuTTY или другого SSH-клиента, но с другого компьютера (см. раздел «Устанавливаем OSMC», стр. 72).

Затем с помощью текстового редактора Nano открой файл *autostart*:

```
$ nano .config/ lxsession/LXDE-pi/autostart
```

Nano – это блокнот, работающий с консоли. Здесь ты не сможешь использовать мышку, но можешь управлять курсором с помощью клавиш со стрелками. Удали строку с комментарием в начале третьей строки. С помощью клавиши со стрелками установи курсор перед строкой со знаком решётки # (третья строка) и нажми клавишу **Delete**. Перед тем как ввести новую команду, вставь знак решётки # и сделай строку неактивной. В конце текста всё должно выглядеть как на рис. 2.15.

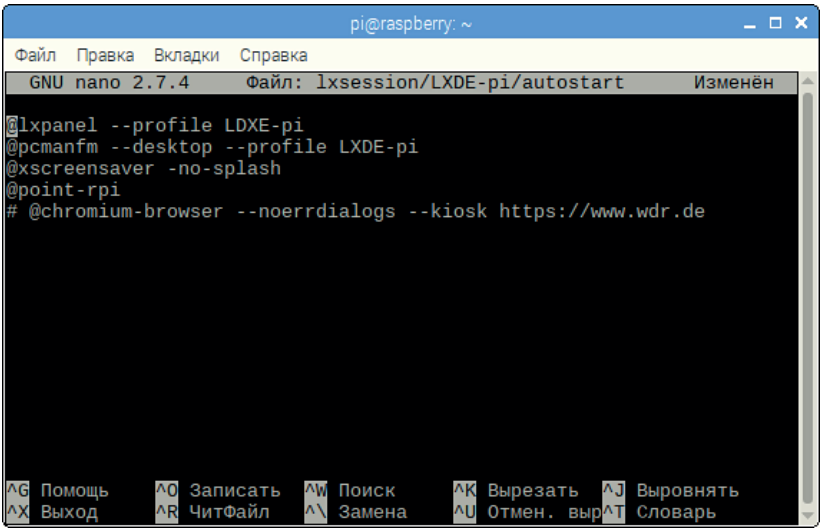


Рис. 2.15. *Тестовый редактор nano обрабатывает файл autostart*

Для сохранения текста нажми комбинацию клавиш **Ctrl**+**O**. Nano попросит у тебя подтверждения на выполняемое действие. Нажми клавишу **Enter** ↵. Файл со всеми внесёнными изменениями будет сохранён. Чтобы закрыть блокнот, нажми комбинацию клавиш **Ctrl**+**X**.

Перезагрузи компьютер. Для этого введи в терминал следующую команду:

```
$ sudo reboot
```

Проект 3. Raspberry Pi как медиацентр

А теперь пора нам сделать что-нибудь эдакое. Ты можешь настроить Raspberry Pi в качестве полноценного медиацентра. Представь, что ты уютно сидишь в кресле

и просматриваешь фото или видео и слушаешь музыку, смотришь телевизионные программы или слушаешь радио через интернет. Медиацентр не предназначен для эффективной работы с компьютером. То есть здесь не нужно вводить текст, обрабатывать графику или программировать. Поэтому для работы с медиацентром клавиатура тебе не понадобится. Тебе хватит мышки или пульта дистанционного управления.

Для использования Raspberry Pi в качестве медиацентра потребуется установить другую операционную систему. Мы будем использовать OSMC (Open Source Media Center) – разработку от медиаплеера Kodi.

Устанавливаем OSMC

Данную программу ты можешь скачать в интернете.

Для этого ты можешь воспользоваться двумя способами:

- скачай OSMC на другую флешку (её объём должен быть не менее 8 Гб). Для этого тебе нужно скопировать файлы NOOBS на свежеотформатированный носитель (см. раздел «Форматирование SD-карты», стр. 26). Далее вставь отформатированную флешку в разъём Raspberry Pi и подключи его к сети;
- можно установить OSMC вместе с Raspbian на SDHC-карту. В этом случае ты будешь запускать Raspberry Pi с той же SDHC-карты, на которую ранее был установлен Raspbian. Для запуска программы-установщика NOOBS нажми и удерживай клавишу ⇧.

Вместе с NOOBS на флешке находится операционная система Raspbian и LibreELEC (это зависит от версии NOOBS). Когда появится окно загрузчика NOOBS, щёлкни мышью на кнопке **Wifi networks (w)**.

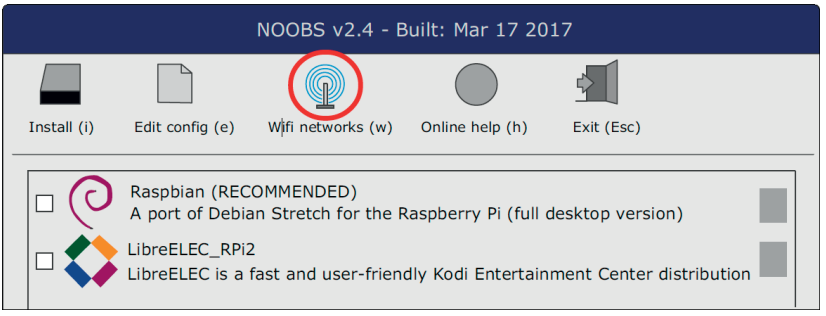


Рис. 2.16. *Создаём соединение с беспроводной сетью*

Следуй указаниям по подключению к беспроводной сети. Как только Raspberry Pi получит доступ в интернет, NOOBS предложит для скачивания большой выбор различных операционных систем.

Чтобы выбрать OSMC, щёлкни мышью на поле левее символа OSMC и установи там крестик, отметив таким образом программу для установки. После чего нажми кнопку **Install** (Установить).

Выбранные операционные системы будут установлены. Если операционная система была установлена ранее, то в следующий раз перед кнопкой появится выпадающее меню.

Щёлкни мышью на символе OSMC и ожидай загрузки. Через некоторое время появится графический интерфейс этой системы. Пока ничего не делай. Спустя ещё пару секунд картинка сменится, и ты увидишь главную страницу сайта.

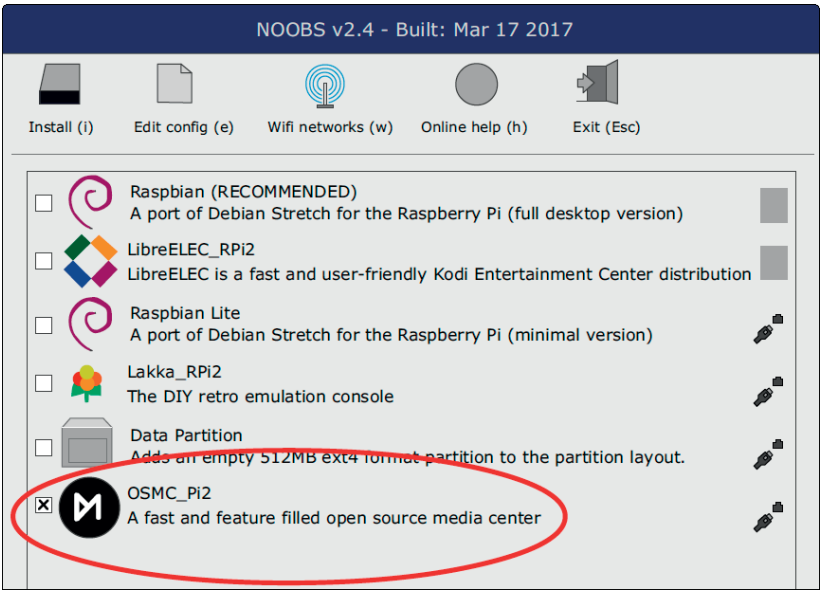


Рис. 2.17. *Лист ожидания операционной системы для Rpi*

Основные настройки

Добро пожаловать!

Первым делом ты увидишь на экране длиннющий список языков. Один из них всегда окрашен светлым цветом. С помощью клавиш со стрелками ты можешь перемещаться по списку. С помощью клавиш ↑ и ↓ выбери язык (например, **русский**) и нажми клавишу **Enter** ↵.

Часовой пояс

Следующий шаг – выбор часового пояса. Выбери город, соответствующий твоему часовому поясу (например, **Москва**).

Имя хостинга

Теперь тебе нужно дать имя своему устройству. Или просто нажми на кнопку **Accept**. В этом случае Raspberry Pi получит одно из предустановленных имён системы OSMC.

SSH

Обрати внимание, что на экране высвечивается сервер SSH Service is Enabled. SSH предназначен для Secure Shell. Нажми на кнопку **Accept**.

Лицензия

Нажми **Продолжить** (Continue).

Работа с сетями

Выбери опцию **I'd like to set up networking manually** (Я хотел бы настроить сеть вручную).

Затем нажми расположенную в левой части экрана кнопку **Wireless** и нажми кнопку **Enable Adapter** (Включить адаптер). Цвет кнопки изменится, как будто у кнопки включилась подсветка. После этого ты увидишь список обнаруженных беспроводных сетей. Выбери свою сеть и введи пароль. Нажми расположенную в нижней части окна кнопку **Применить** (Apply). После этого будет установлено соединение, и ты увидишь статус **Connected** (Подключено).

Смотри и чувствуй

Теперь ты можешь выбрать наиболее понравившийся дизайн интерфейса. Предлагается интерфейс OSMC и классический вариант. В книге мы выбрали классический интерфейс (**Classic**) от Kodi.

Для завершения настроек щёлкни мышью на кнопке **Exit** (Выход).

Выключение и навигация

Кнопка ⏻ выключения Raspberry Pi находится в левом верхнем углу главной страницы. Чтобы вернуться на главную страницу, щелкни *правой* кнопкой мыши или нажми клавишу **Esc**.

Рис. 2.18. *Так выглядит главная страница OSCM после установки*

Проигрывание музыкальных файлов с флешки

С помощью OSMC ты можешь слушать музыку или смотреть видео, записанные на карту Raspberry Pi или на флешку. Возьмём следующий пример. Предположим, на твоей флешке есть папка **Музыка** с сохранёнными в ней музыкальными файлами. Ты вставляешь эту флешку в разъём своего Raspberry Pi. Выбираешь в левой части главной страницы команду **Musik**. Откроется новая страница. В правой части страницы щёлкни мышью на команде **File**.

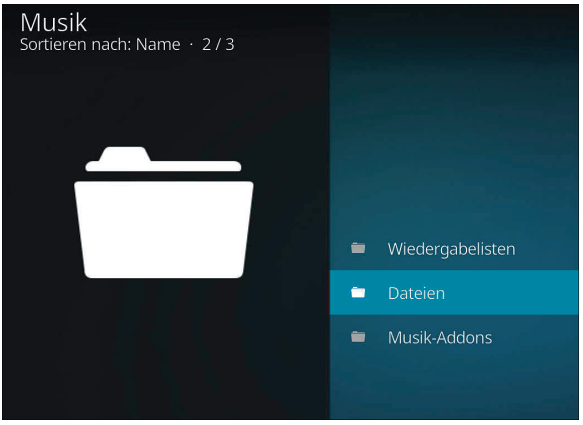


Рис. 2.19. *Выбираем музыкальные файлы*

Появится диалоговое окно со списком. Найди в списке имя своей флешки и щелкни на нём мышью. Флешка откроет-

ся, и ты увидишь название сохранённых папок. Дважды щёлкни мышью на папке, в которой хранятся твои музыкальные файлы. Появится список сохранённых композиций. Список состоит из имени исполнителя, названия песни и её продолжительности. Это дополнительные данные, или *метаданные*. Метаданные сохраняются в дополнение к звуку.

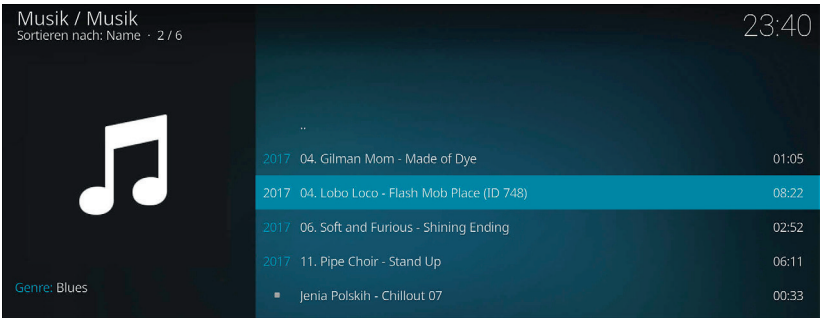


Рис. 2.20. *Так выглядит композиция в музыкальной папке*

Чтобы воспроизвести композицию, щёлкни на ней мышью. Для остановки композиции на паузу или выключения воспроизведения смести указатель мыши в левую часть окна проигрывателя. На экране появится меню с элементами управления проигрывателем.

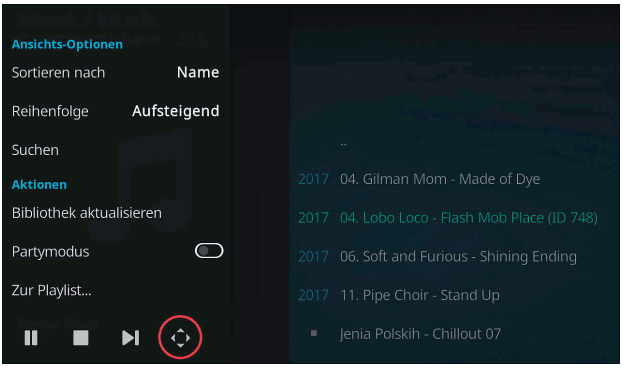


Рис. 2.21. *Меню с элементами управления проигрывателем*

Нажми кнопку . На экране появится страница для управления музыкальным треком.

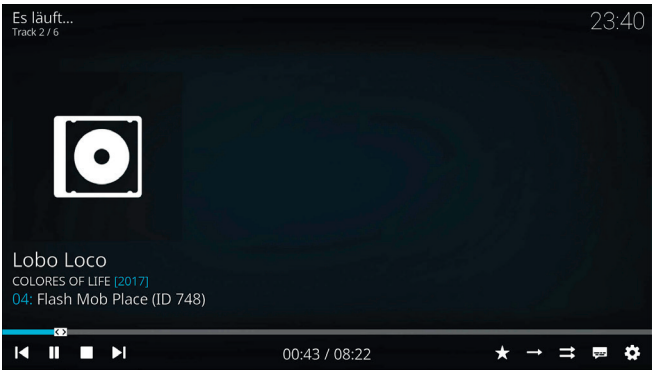


Рис. 2.22. *Панель управления для воспроизведения музыки*

OSMC предлагает множество возможностей для создания плей-листов и управления ими. Успехов тебе в этом деле!

GEMA – доступная музыка

Даже у музыкальных файлов существуют авторские права. Нельзя просто так взять и скачать музыку из интернета и использовать её для собственного проекта. Компания GEMA (Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte) – (Общество по защите авторских прав на воспроизведение и тиражирование музыкальных произведений) следит за тем, чтобы музыканты не остались без денег при публичном использовании их произведений. Но и здесь есть подарочные музыкальные произведения, которые можно скачать в свободном доступе. Список таких произведений помечен как GEMA-frei (frei – свободный). Найти этот список можно по адресу: https://www.medienpaedagogik-praxis.de/kostenlose-medien/freie-musik/.

Просмотр видео- и ТВ-программ

В интернете просматривать видео можно с помощью прямой трансляции (потоковое видео) через так называемый стрим (stream – трансляция). Во время этого процесса скачать и сохранить видео нельзя, а сам ролик загружается на компьютер по частям. Пока идёт трансляция одной части видеоролика, система в фоновом режиме загружает следующую часть. Поэтому сохранить можно только те части файла, которые в данный момент воспроизводятся. Про остальные система как бы «забывает», поскольку они больше не нужны.

Для просмотра трансляции из интернета тебе придётся дополнительно установить программный модуль под названием **аддон** (addon от англ. *to add on* – добавить, расширить).

Это делается следующим образом.

Вернись на главную страницу (просто нажми несколько раз на кнопке **Esc**) и щёлкни мышью на команде **Addons**. Дальше в появившемся меню щёлкни мышью на команде **Скачать**.

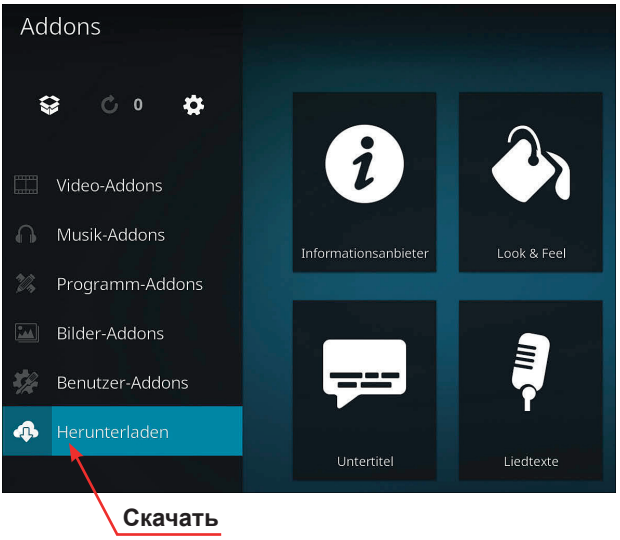


Рис. 2.23. *Скачиваем Addon*

Появится список с различными хранилищами (репозиториями). Такое хранилище представляет собой сайт в интернете, который подготавливает программу к скачиванию. Нажми **Video-Addons**.

Рис. 2.24. *Хранилище Addons для OSMC*

Ты увидишь список аддонов. Чаще всего это медиатреки с телевизионными программами.

Рис. 2.25. *В списке хранилища представлены 378 аддонов*

Щёлкни левой кнопкой мыши на одном из аддонов. Например, на медиатреке канала ARD. Появится новое окно с описанием программы. Если эта программа тебе понравилась, нажми кнопку **Установить**.

Рис. 2.26. *Addon-модуль для медиатреков канала ARD*

После установки надстройки появится список доступных аддонов. Вернись в главное меню. Для этого щёлкни правой кнопкой мыши или нажми клавишу **Esc**. Установи указатель мыши на загруженные аддоны. На экране появятся значки установленных видеомодулей.

Рис. 2.27. *Главное меню с установленными аддонами*

Теперь собери свой собственный список аддонов! Точно так же ты можешь настроить аддоны и для информационных приложений (погода, тексты песен и т. д.). Создай свой собственный медиацентр!

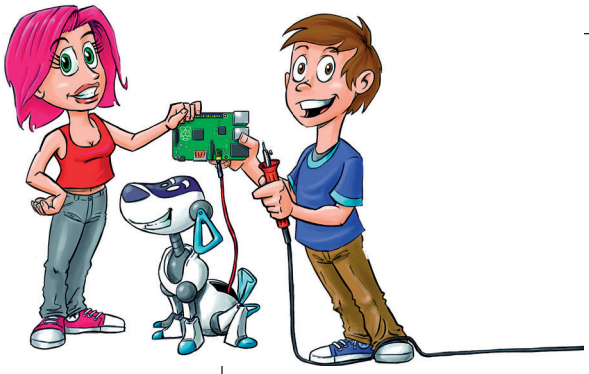
Вопросы

1. С помощью какой команды можно узнать IP-адрес Raspberry Pi?
2. Какие устройства не подключаются, если Raspberry Pi находится в режиме «без головы»?
3. Для чего используется SSH-соединение?
4. Какие задачи выполняет DHCP-сервер?
5. Почему в системе OSMC ты можешь щёлкать мышкой в нескольких местах экрана?
6. Зачем для просмотра видео в интернете использовать поток (способ прямой трансляции)?

Ответы

1. IP-адрес Raspberry Pi можно запросить с помощью команды ifconfig.
2. В «безголовом» режиме Raspberry Pi может работать без мышки, монитора и клавиатуры, а также без устройства ввода-вывода данных.

3. Соединение SSH (например, с помощью PuTTY) позволяет использовать Raspberry Pi с другого компьютера. Для этого нужно запустить SSH-клиент (тот же PuTTY) и с помощью IP-адреса Raspberry Pi создать подключение. Авторизация проходит через удалённый компьютер.
4. DHCP-сервер назначает недавно подключённому по локальной сети компьютеру уникальный IP-адрес. Этот IP-адрес будет предоставлен только твоему компьютеру. Другим компьютерам будут предоставлены другие уникальные IP-адреса.
5. Система OSMC была разработана как медиацентр. Управлять ею можно без клавиатуры с помощью мышки или пульта управления. Везде, где нужно вводить текст, есть поле ввода.
6. Через стрим (поток) ты можешь начать смотреть фильм сразу после нажатия кнопки воспроизведения. Фильм загружается по частям, и каждая последующая часть будет загружаться во время воспроизведения предыдущей части. Чтобы посмотреть этот видеоролик на своём компьютере, сначала нужно его скачать целиком. Скачивание может занять некоторое время.



3 Автогонки и метеоры: как их программировать в Scratch?

На твоём Raspberry Pi уже установлено несколько языков программирования. С двумя языками программирования – Scratch и Python – мы познакомимся ближе. Для первых шагов в программировании лучше всего подойдёт язык Scratch.

Во время работы с ним ты соберёшь программу из различных элементов, просто перемещая их мышкой по экрану. Это так же просто, как и с конструктором «Лего». Scratch был разработан исследовательской группой «Lifelong Kindergarten» в Массачусетском институте технологий (Massachusetts Institute of Technology). Данный институт расположен в Кембридже, в США (сокращённо MIT, но произносить надо как «эм ай ти»). MIT – это один из наиболее известных технических университетов мира. Несмотря на то что Scratch позволяет создавать игры, это не безделушка, а очень серьёзная программная среда.

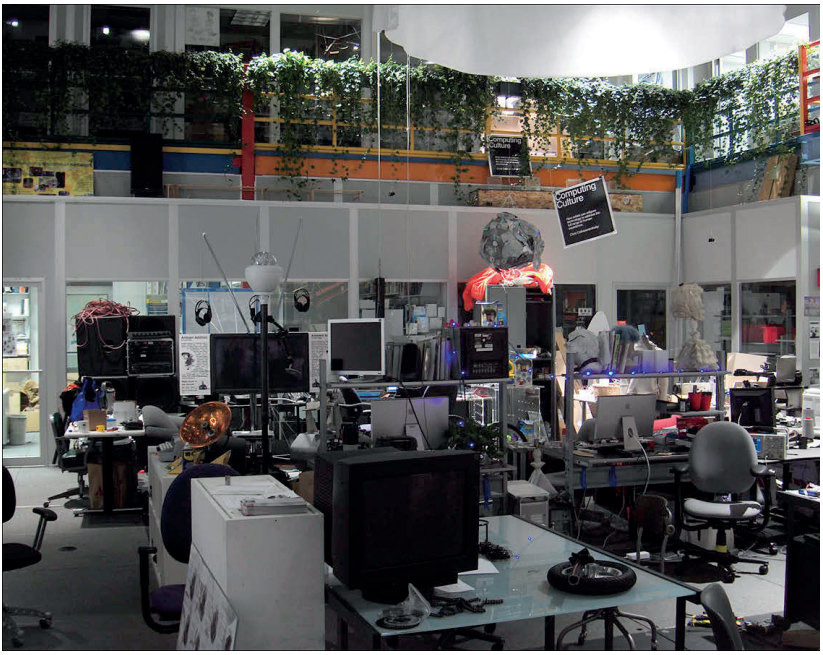


Рис. 3.1. *Здесь создавался Scratch. Так выглядит офис «Lifelong Kindergarten» в MIT (США)*

Запуск Scratch

Открой главное меню, нажав на кнопку . В открывшемся главном меню наведи указатель мыши на строку **Программирование** и в появившемся подменю щёлкни мышью на строке **Scratch** (обрати внимание: нужно выбрать строку **Scratch**, а не Scratch 2). Откроется пользовательское окно программы.

Для начала изучи интерфейс и попробуй сделать несколько практических шагов.

1. На сцене ты увидишь котёнка. Кот – это *объект,* а сцена – *это твоя рабочая площадка*. Все фигуры, которые ты будешь создавать, называются объектами. Это своего рода актёры Scratch-проекта. Они могут двигаться по сцене и могут быть запрограммированы.
2. Над сценой, в полосе меню ты увидишь три кнопки: **Переключиться к маленькому экрану** , **Переключиться к среднему экрану** и **Перейти в режим презентации** . С помощью этих трёх кнопок ты можешь изменять пространство сцены. На рисунке сцена выглядит очень маленькой (нажата кнопка **Пе-**

реключиться к маленькому экрану). Ты же можешь изменить размер сцены и сделать её большой или очень большой. Попробуй! Если сцена стала большой, а вокруг сцены появилось чёрное поле, значит, ты находишься в режиме презентации. В этом режиме слева вверху над сценой ты увидишь изогнутую стрелку. Нажав на неё, ты вновь вернёшься к нормальному режиму сцены. Но из режима презентации можно выйти и по-другому – просто нажать клавишу **Esc**.

3. Ниже сцены находится символ выбранного Scratch-объекта. Сейчас это котёнок. Объект котёнок используется сейчас, поэтому его символ обведён синей рамкой. Имя этого объекта – **Спрайт1**. Но это имя ты можешь изменить.
4. На каком языке написаны слова в Scratch-окне – на русском или английском? Если всё на английском, значит, тебе нужно настроить языковую панель. Для этого щёлкни мышью на значке, обозначающем глобус, и выбери из открывшегося списка языков **Русский**.

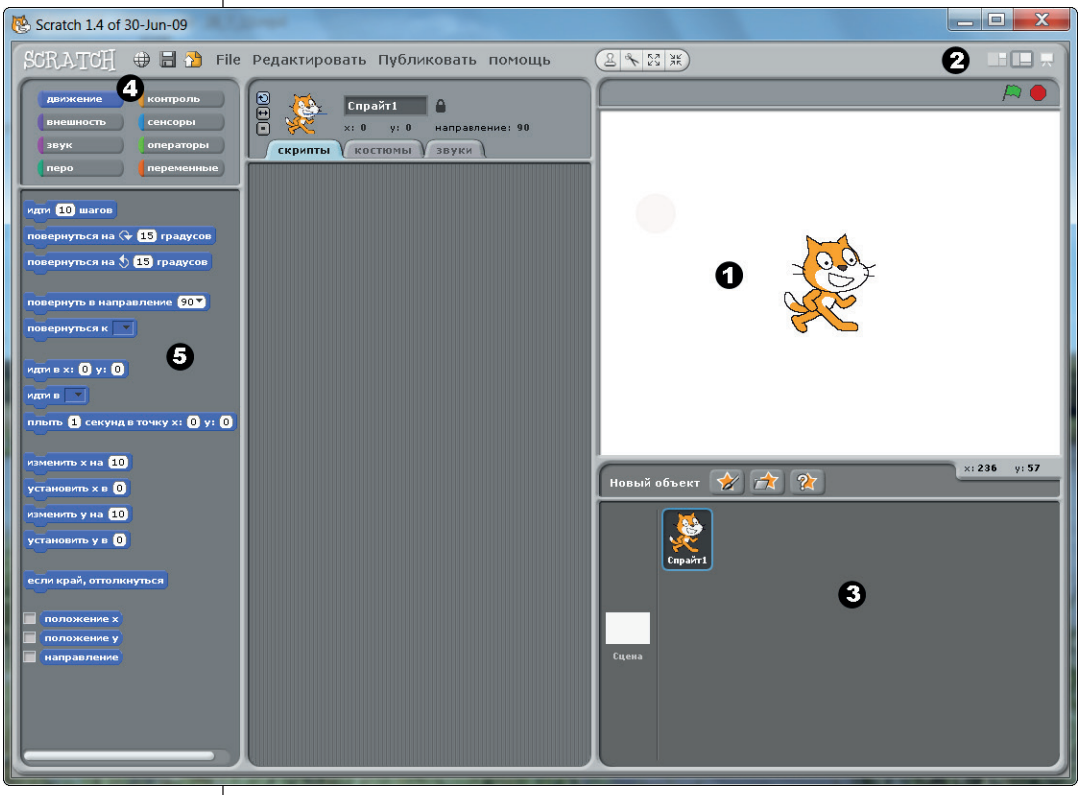


Рис. 3.2. *Интерфейс программы Scratch*

5. В левой части окна Scratch находится панель с командами для управления объектами. Эта панель разделена на две части. В верхней части ты увидишь 8 кнопок с названиями групп команд. Команды выбранной группы находятся в нижней части панели. Сами команды представляют собой блоки, на которые можно нажимать, да и сами эти блоки передвигаются. Например, чтобы немного повернуть расположенного на сцене котёнка, щёлкни мышью на команде 

. Ты сразу увидишь реакцию котёнка – он наклонится вправо. Верни котёнка в первоначальное положение. Для этого щёлкни мышью на команде 

. Ты можешь переместить блоки в середину рабочей поверхности и составить скрипт для объекта. Один скрипт управляет движением одного объекта. Более подробно мы поговорим об этом в следующем разделе. Как уже говорилось ранее, существует восемь групп команд по различным темам. Сейчас по умолчанию у нас выбрана коллекция **движение**. Ты это понял уже по тому, что кнопка **движение**, расположенная в верхней левой панели, окрашена в синий цвет.

Scratch и Scratch 2

У Raspberry Pi установлено две версии программы – классический Scratch и современный Scratch 2. В этой книге для создания проектов мы будем использовать классическую версию. Более новая версия содержит дополнительные блоки и векторный графический редактор. Но новая версия также потребует и большей вычислительной мощности, а это проблема для маленького Raspberry Pi. Так что остановимся на классическом варианте.

Проект 4. «Ухуху-у-у!» – первый Scratch-проект

Первым проектом у нас будет компьютерный мультфильм: призрак появляется на экране и произносит: «Ухуху-у-у!»

Порядок действий следующий. Первым делом ты программируешь движение. Затем создаёшь внешний облик своего призрака и в завершение рисуешь фон.

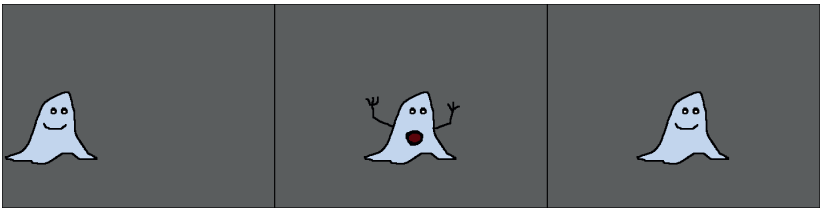


Рис. 3.3. *Скриншот компьютерного мультфильма*

Начинаем новый проект

Запускаем Scratch. Сохраняем ещё не начатый проект следующим образом.

- Выбери команду меню **Файл** ⇨ **Сохранить как**. На экране появится окно, в котором нужно будет выбрать папку для сохранения проекта. В поле ввода **Новое имя файла** введи имя проекта, например Ghost (что значит «призрак»), и нажми кнопку **OK**. Проект сохранён.
- По умолчанию для сохранения предлагается папка Scratch Projects. Щёлкни мышью на кнопке **Новая папка**, введи в появившемся окне имя новой папки, например **Мои проекты Скретч**, и нажми кнопку **OK**. Дважды щёлкни на строке с названием этой папки, чтобы её открыть. В этой папке и будут сохранены все твои проекты.
- Введи в поле ввода **Автор проекта** своё имя или псевдоним. Это поле ввода находится в правой части окна **Сохранить проект**.
- В поле ввода **Об этом проекте** кратко опиши свой проект. Напиши, например, Страшный летающий призрак.
- Введи внизу имя файла (например, Ghost) и нажми кнопку **OK**. Твой проект сохранён.

Важно
Сохранение Scratch-проекта в Raspberry Pi потребует времени, поэтому запасись терпением. Лишь после того, как растущий из линий прямоугольник проскочит по экрану, процесс сохранения будет завершён.

Как заставить кота двигаться и говорить?

На сцене сидит наш котёнок, а в нижнем окне этот персонаж выбран как объект. Первым нашим шагом станет написание программы (её ещё называют *скрипт*), которая

заставит котёнка двигаться. Чуть позже ты этого котёнка превратишь в призрак.

Посередине окна программы Scretch расположена панель, в которой и собирается скрипт из командных блоков, которые находятся в левой панели. Панель, на которой собирается скрипт, называется рабочей поверхностью и состоит из трёх вкладок: **скрипты**, **костюмы** и **звук**. Каждый командный блок содержит в себе команду. Чтобы перетащить командный блок на рабочую поверхность, выполни следующие действия: установи указатель мышки на блоке с нужной командой, нажми левую кнопку мыши и, не отпуская её, перетащи командный блок на рабочую поверхность. После чего левую кнопку мыши отпусти. В дальнейшем мы будем писать «перетащи блок».

Все блоки подходят друг к другу, как детали пазла. Когда ты будешь устанавливать один командный блок под другим, проследи, чтобы выступ верхнего блока попал в паз устанавливаемого блока. Когда ты подставляешь новый блок, то можешь увидеть, как он «встаёт» в нужное место. Удалить лишний блок с рабочей поверхности очень просто. Для этого перетащи ненужный блок в правую панель.

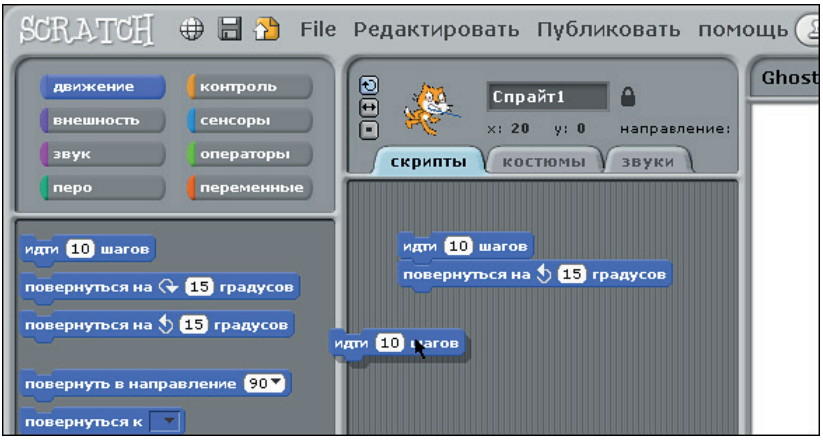


Рис. 3.4. *«Строим» скрипт из блоков-команд*

Ранее уже упоминалось, что в программе есть восемь групп команд. Кнопки с названиями этих групп команд находятся в панели, расположенной в левом верхнем углу программы. Каждая кнопка, как и открываемая группа соответствующих команд, окрашена в свой цвет. Кнопка группы команд **контроль** окрашена в оранжевый цвет, **движение** – в синий и т. д. Для каждой группы команд тебе понадобятся поля **управление**, **вид**, **движение**.

Теперь собери скрипт, как показано на рис. 3.5.

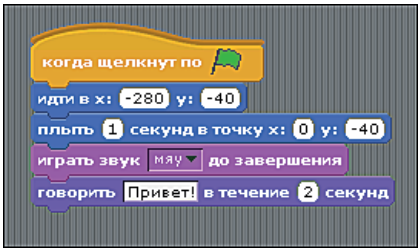


Рис. 3.5. *Скрипт программы Scratch*

Наш первый скрипт состоит из пяти команд: **когда щёлкнуть по**; **идти в**; **плыть**; **играть звук** и **говорить**. Второй, третий и пятый блоки имеют маленькие поля ввода, куда вводятся координаты объекта и время движения, слова приветствия и время, на протяжении которого кот будет тебя приветствовать. Приветствие будет напечатано на экране.

Верхний блок мы взяли из группы команд **контроль**. Этот блок, после того как ты щёлкнешь мышью по зеленому флажку, расположенному справа над сценой, запустит нашу программу. Две следующих команды **идти в** и **плыть** взяты из группы команд **движение**. Обрати внимание: в этих командах есть поля ввода **X:** и **Y:**, в которые вводятся координаты, по ним двигается персонаж. Команда **идти в** перемещает объект согласно введенным в поля ввода координатам. То есть за левый край экрана. Команда **плыть** определяет, в какое место сцены и за какое время должен переместиться наш котёнок. Впрочем, чуть ниже, в разделе «Координаты сцены», я всё подробно объясню. Команда **играть звук** взята из группы элементов управления **звук**. По умолчанию выбран звук «**Мяу**». Команда **говорить** взята из группы команд **внешность**. Чтобы при желании изменить текст приветствия, щёлкни мышью в поле ввода со словом «**Привет!**» и введи свой текст.

Щёлкни мышью на зелёном флажке, который находится справа над сценой, и наблюдай, что произойдёт. Если ты сделал всё правильно, то котёнок переместится с левого края на середину сцены, мяукнет и скажет: «Привет!»

Координаты сцены

Чтобы запустить движение, нам понадобится указать то место на сцене, в которое должен переместиться персонаж. Любое место на сцене может быть описано двумя цифрами,

которые называются координатами. Это координата по горизонтальной оси ***x*** и по вертикальной оси ***y***. Про систему координат тебе рассказывали на уроках математики. Центральная точка находится на пересечении координатных осей (***x*** = **0** и ***y*** = **0**). Это и есть центр нашей сцены. Значение координаты левого края сцены **–240**, правого края сцены **240**. Значение координаты верхнего края сцены **180**, а нижнего **–180**. То есть если ты хочешь, чтобы котёнок появился не с середины левого края сцены, а с нижнего левого угла, введи в поле ввода **y:** команду **идти** в значение **–200** (это вторая сверху команда).

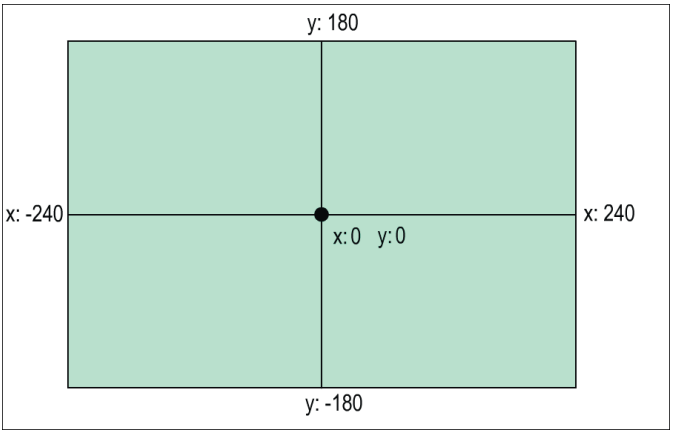


Рис. 3.6. *Оси координат*

Собственные координаты и помощь

Программа Scratch разработана таким образом, что все команды ты можешь изучить самостоятельно. Чтобы узнать, как использовать тот или иной блок, щёлкни правой кнопкой мыши на интересующем тебя блоке и выбери в появившемся меню команду **помощь**. Появится окошко с пояснениями и маленькой инструкцией на английском языке.

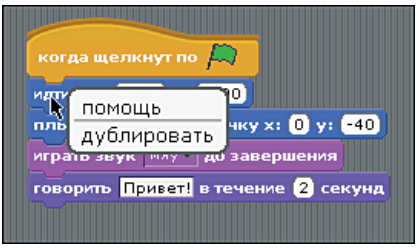


Рис. 3.7. *Для каждого блока можно запросить справку*

Костюмы: как сделать из кота привидение

Кот в программе Scratch – это объект, который умеет двигаться, и у этого объекта может быть множество разных костюмов. «Костюм» в данном случае – это не одежда, а внешний вид объекта в целом.

Итак, щёлкни мышью на ярлыке вкладки **костюмы**. Откроется вкладка с различными вариантами для нашего кота и списком костюмов.

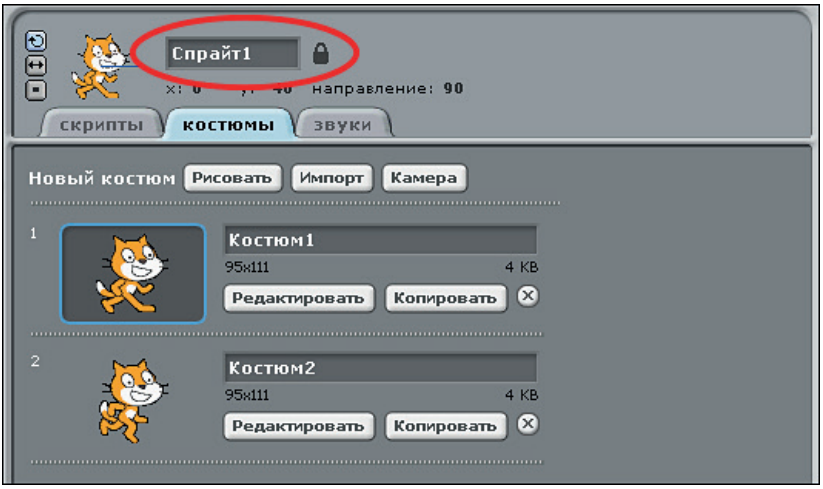


Рис. 3.8. *Костюмы (внешний вид) объектов*

Представь себе, что объект – это актёр, играющий свою роль на сцене театра. Так же, как и актёру, нашему объекту нужны костюмы и возможность менять свой облик. У Scratch-кота есть два костюма. В дальнейшем ты можешь добавлять другие костюмы, удалять их или обрабатывать. В нашем проекте кот должен стать привидением. Поехали!

Шаг первый – даём имя объекту

Прежде всего у нашего объекта должно появиться имя. Назовём его **Привидение**. Тебе необходимо ввести имя в верхней части поля, где изначально находится строка **Спрайт1** (на рис. 3.8 это поле ввода обведено красным овалом).

Шаг второй – удаляем ненужные костюмы

Чтобы удалить поле ввода с надписью **Костюм2**, нажми кнопку ⊗, расположенную под этим полем.

Шаг третий – рисуем привидение

Теперь нажми кнопку **Редактировать**, расположенную под полем ввода **Костюм1**. Откроется новое окно под названием **Графический редактор**. Он работает так, как и любой другой графический редактор (например, Microsoft Paint).

- Нажми кнопку **Очистить**. Нарисованный кот исчезнет с экрана.
- Выбери в группе инструментов кнопку и щёлкни мышью на кнопке **Кисточка**, а в группе элементов управления щёлкни мышью на верхнем квадратике чёрного цвета. Этим ты для кисти выберешь чёрный цвет.
- Нарисуй контур и лицо привидения, как на рис. 3.9.
- Выбери светло-серый цвет заливки. Для этого в палитре цветов щёлкни мышью на ячейке с понравившимся цветом.
- В панели инструментов щелкни мышью на кнопке инструмента **Заливка**. Установи указатель мыши, принявший форму ведёрка, внутри контура нарисованного тобой привидения и щёлкни мышью. Нарисованное тобою привидение будет окрашено в выбранный тобою цвет (рис. 3.9).
- Когда закончишь рисовать, нажми в правом нижнем углу окна графического редактора кнопку **ОК**.

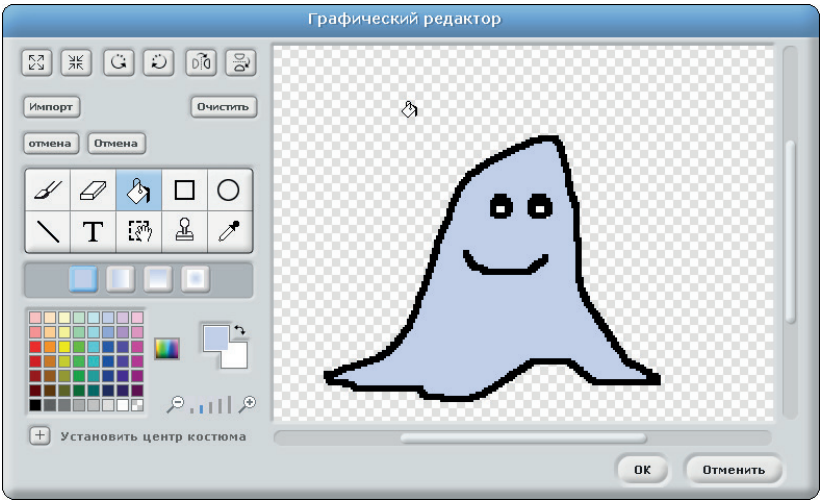


Рис. 3.9. *Меняем костюм (внешний вид) призрака*

Шаг четвёртый – создаём второй костюм

Чтобы скопировать костюм, нажми кнопку **Копировать**, расположенную на вкладке **Новый костюм** под полем ввода **Костюм1**. Теперь у тебя получилось два одинаковых костюма для привидения. Когда будешь менять второй костюм, позаботься о том, чтобы рот призрака оставался открытым, потому что он должен что-то говорить. Для этого выполни следующие действия:

- нажми кнопку **Редактировать**, которая находится под полем ввода **Костюм2**. Откроется окно графического редактора, в котором ты увидишь нарисованное тобой привидение;
- в панели инструментов графического редактора нажми кнопку **Ластик** и сотри очертания рта;
- теперь выбери инструмент **Пипетка** . Установи указатель мыши, который примет форму пипетки , внутри контура привидения и щёлкни мышью. Пипетка скопирует тот оттенок, на который была установлена;
- выбери в панели инструментов инструмент **Заливка** и щёлкни мышью (в виде ведра) на месте стёртого рта;
- выбери инструмент **Кисточка** , в палитре цветов щёлкни мышью на ячейке с чёрным цветом и кисточкой нарисуй открытый рот. Можешь выбрать тёмно-красный цвет и закрасить им внутри рта;
- если хочешь, снова выбери чёрный цвет и дорисуй привидению руки, как на рис. 3.10;
- когда редактирование будет завершено, закрой окно графического редактора, нажав кнопку **OK**;
- переименуй оба костюма, например так, как на рис. 3.10.

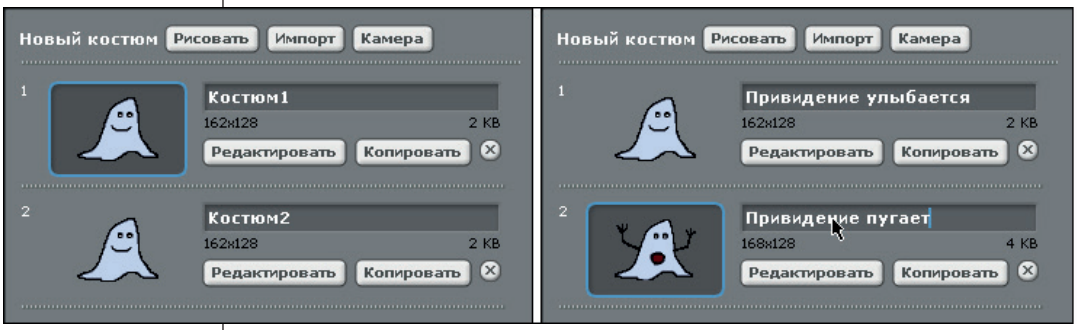


Рис. 3.10. *Костюм до и после изменения*

Шаг пятый – расширяем скрипт

Теперь тебе нужно так изменить скрипт, чтобы в то время, когда призрак пугает, его рот был открыт, руки (если ты их пририсовал) подняты и слышался звук «**Мяу**». Для этого тебе нужно добавить три блока команд: **перейти к костюму**, **следующий костюм** и **играть звук**. Блок команд **следующий костюм** устанавливается после блока **говорить**. Далее мы добавим блок **играть звук** и завершим блоком **следующий костюм**. Блок **следующий костюм** меняет костюмы следующим образом. Вначале мультфильма у нас применяется костюм **Приведение улыбается**. В этом виде персонаж появляется на сцене. Далее выше и правее улыбающегося персонажа появляется надпись «Ухуху-у-у!». После чего, повинуясь команде **следующий костюм**, на экране появляется персонаж с открытым ртом и поднятыми руками и слышится звук «Мяу!». Звук выдаёт команда **играть звук**. Ну и завершающая команда **следующий костюм** возвращает на экран костюм **Привидение улыбается**.

Чтобы привидение всё это сделало, выполни следующие действия:

- открой вкладку **скрипты**;
- щёлкни мышью в поле ввода командного блока **говорить** и введи слово, которое должно сказать привидение, появившись на экране: «**Ухуху-у-у!**»;
- отстыкуй командный блок **говорить** (рис. 3.11 левый верхний);
- в левой верхней панели нажми кнопку **Внешность**;
- выбери командный блок **перейти к костюму** (находится в самом верху панели), установи на него указатель мыши, нажми левую кнопку мыши и, не отпуская её, перемести выбранный командный блок в рабочую область вкладки **скрипты.** Далее подстыкуй этот командный блок под команду **плыть** и отпусти левую кнопку мыши;
- открой находящийся в правой части командного блока **перейти к костюму** открывающийся список и выбери название улыбающегося персонажа. На рисунке (рис. 3.11 посередине) это строка **Привидение улыбается**;
- подстыкуй командный блок **говорить**;
- выбери и пристыкуй блок команд **следующий костюм**;
- в левой верхней панели щёлкни мышью на кнопке **звук**, выбери командный блок **играть звук**, перетащи этот блок в рабочую область вкладки **скрипты** и подстыкуй этот блок под команду **следующий костюм**;

- снова выбери в левой верхней панели группу команд **внешность**. Перетащи в рабочую область вкладки **скрипт** командный блок **следующий костюм** и подстыкуй его к блоку команд (рис. 3.11 правый). Скрипт готов.

Рис. 3.11. *Расширяем скрипт*

Да наступит ночь! Изменяем фон

Как все мы знаем, привидения гуляют только по ночам. Поэтому тебе ещё нужно изобразить на сцене ночь. Щёлкни мышью на значке **Сцена**, который ты найдешь на правой нижней панели. В средней панели появятся три вкладки: **скрипты**, **фоны** и **звуки**. Щёлкни мышью на ярлыке вкладки **фоны** и нажми кнопку **Редактировать** (рис. 3.12).

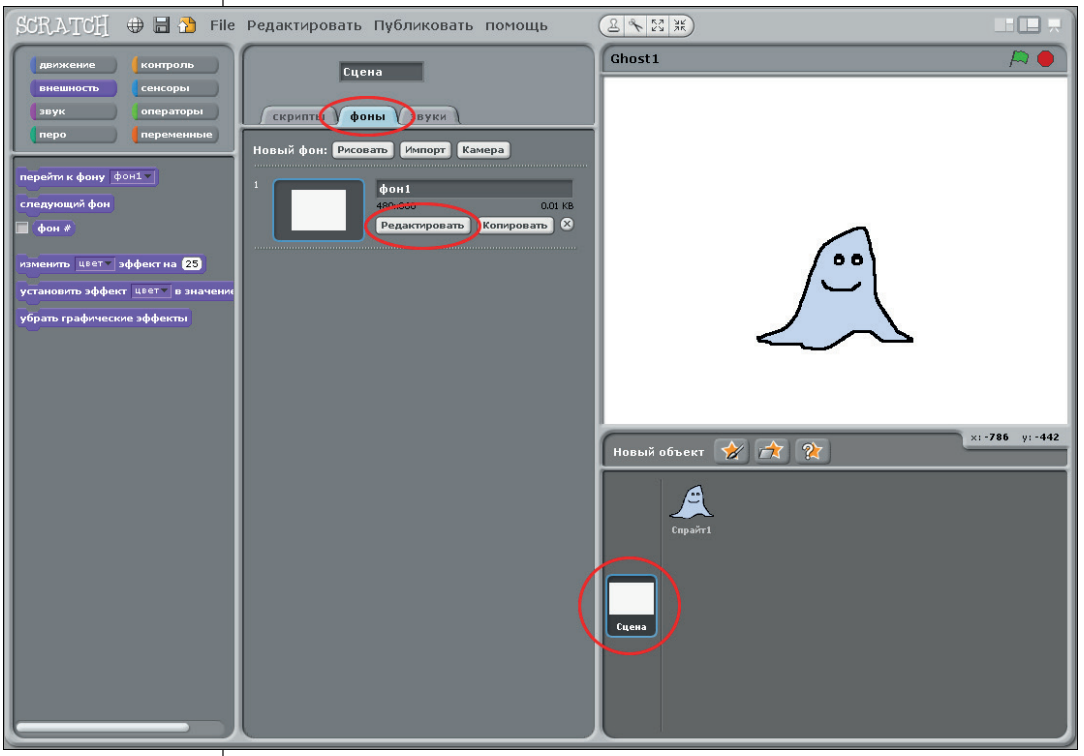


Рис. 3.12. *Меняем фон сцены*

Появится окно знакомого нам графического редактора. Выбери в палитре цветов серый цвет (будем считать, что настали сумерки). В панели инструментов нажми кнопку **Заливка** и залей сцену чёрным цветом (установи указатель мыши в рабочую область графического редактора и щёлкни мышью). Нажми кнопку **OK** в правом нижнем углу окна, чтобы закрыть его. Сцена окрасится в выбранный цвет.

На этом проект можно считать законченным. Не забудь этот проект сохранить. Как создать мультфильм со многими персонажами, ты узнаешь из проекта этой главы «Проект 6: “На помощь! Метеориты!”».

Проект 5. Гоночная трасса Формулы 1

Этот проект автосимулятора. На сцене у нас будет гоночная трасса, по которой ездят машины. Машинами ты можешь управлять с помощью клавиш со стрелкой. Всякий раз, когда ты будешь нажимать клавишу ←, объект будет поворачиваться вправо. При нажатии кнопки → машинка повернёт налево. Сначала автомобиль будет двигаться с неизменяемой скоростью, а останавливаться только при сходе с трассы. Это будет обозначать конец игры.

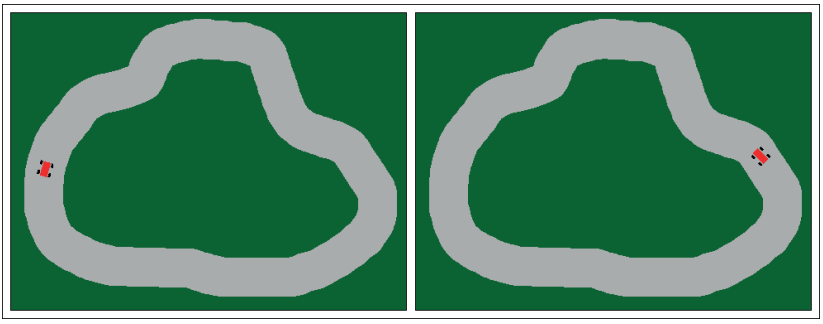


Рис. 3.13. *Снимок экрана в режиме автосимулятора*

Гоночная трасса

Создай новый проект и сохрани его, назвав, например, *Auto*.

Сначала тебе нужно будет нарисовать трассу.

В левой части нижней правой панели щёлкни мышью на кнопке **Сцена** и выбери вкладку **фоны**, после чего нажми кнопку **Редактировать** (рис. 3.14).

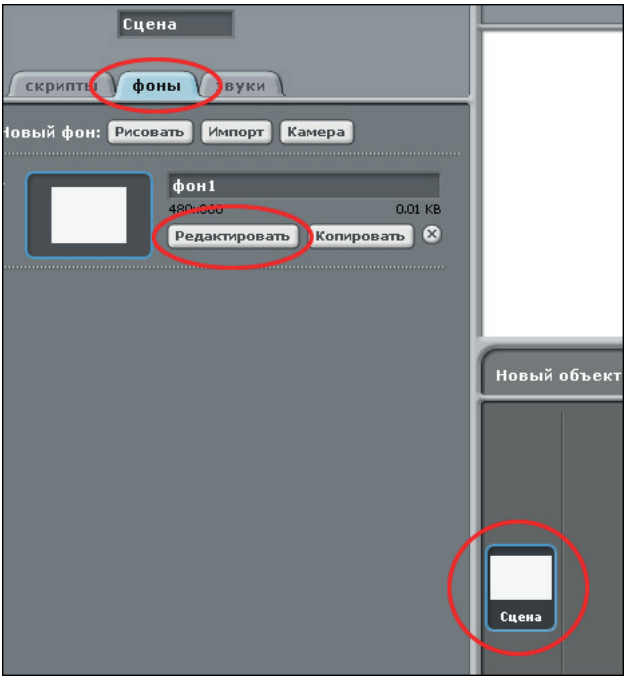


Рис. 3.14. *Рисуем фон сцены*

В панели инструментов нажми кнопку **Заливка**, выбери в палитре темно-зелёный цвет и залей им рабочее поле графического редактора. Нажми в панели инструментов кнопку **Кисточка**, щёлкни мышью на расположенном под панелью инструментов поле **Размер кисти** и в появившемся меню размеров кистей щёлкни мышью на третьей кнопке нижнего ряда кнопок, обозначающей средний размер кисти (рис. 3.15). Выбери в палитре светло-серый цвет (им будет нарисована асфальтированная дорога), нарисуй трассу и закрой графический редактор, нажав кнопку **OK**.

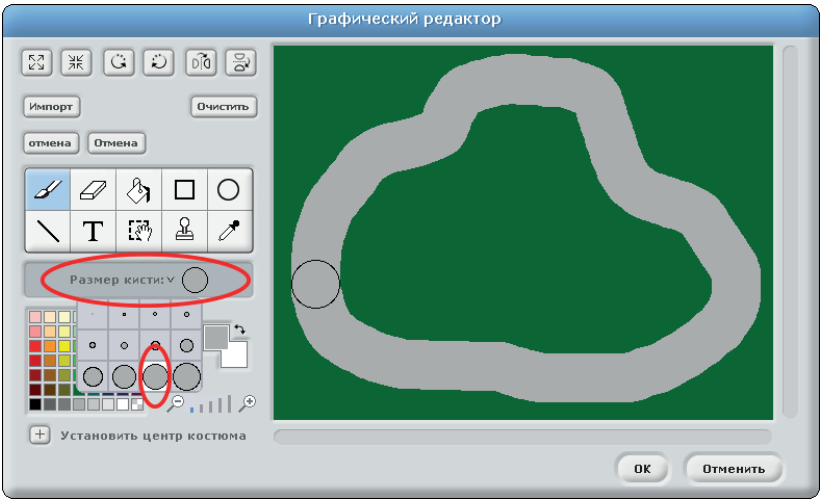


Рис. 3.15. *Чертим трассу кистью*

Автомобиль

Каждый новый проект в Scratch-программе уже содержит объект – кота. Но сейчас кот тебе не нужен, поэтому его нужно удалить. В правой нижней панели щелкни правой кнопкой мыши на изображении кота и выбери из появившегося меню команду **удалить**.

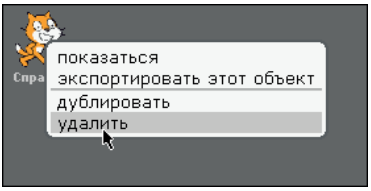


Рис. 3.16. *Удаляем объект*

В панели инструментов правой нижней панели нажми на кнопку **Рисовать новый объект** . На экране появится окно графического редактора. В этом редакторе ты и нарисуешь внешний вид машины. В программе изображение машинки будет называться костюмом.

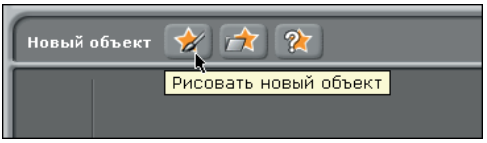


Рис. 3.17. *Кнопка* ***Рисуем новый объект***

Когда будешь рисовать машинку, учти: так как машинка на экране будет маленькой, её форма должна быть простой и чёткой.

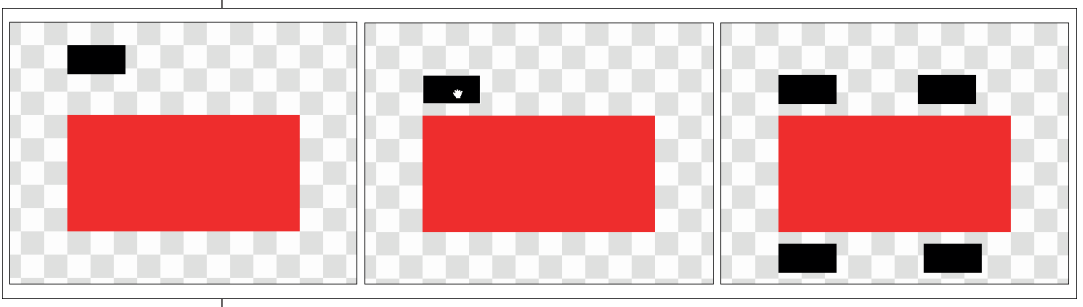


Рис. 3.18. *Автомобиль с шаблонными колёсами*

Чтобы нарисовать четыре одинаковых колеса, нарисуй одно колесо и после воспользуйся шаблоном. Как это делать, рассказано ниже и показано на рис. 3.18. Обрати внимание: поле для рисования расчерчено на светло-серые и белые квадратики, похожие на шахматное поле. С помощью этих квадратиков удобно рисовать фигуры нужного размера.

- Нажми в панели инструментов кнопку ◻. Ниже панели инструментов появится вспомогательная панель с двумя кнопками: закрашенный и незакрашенный квадратики. Закрашенный квадратик обозначает, что будет нарисован полностью закрашенный квадрат, незакрашенный – что будет нарисован контур. Нажми левую кнопку с изображением закрашенного квадрата. Выбери в палитре красный цвет.
- Чтобы тебе было удобнее рисовать, увеличь масштаб. Для этого нажми два раза кнопку 🔎, расположенную правее палитры для выбора цвета. Установи указатель мыши в поле для рисования и нарисуй прямоугольник длиной 10 квадратиков и шириной 5 квадратиков. Это будет корпус машинки. Далее выбери в палитре чёрный цвет и нарисуй прямоугольник с размерами 2,5 квадратика длиной и 1,25 квадратика шириной (рис. 3.18 левый).
- Нажми в панели инструментов кнопку ⬚, обведи черный прямоугольник. Для этого установи указатель мыши левее и выше левого верхнего угла колеса, нажми кнопку мыши и, не отпуская её, перемести указатель правее и ниже правого нижнего угла выделяемой фигуры. Далее установи указатель мыши на выделенный прямоугольник, нажми левую кнопку мыши и, не отпуская её, перемести колесо на место (рис. 3.18 сред-

ний). При этом указатель мыши примет вид руки. Когда прямоугольник займет своё место, левую кнопку мыши отпусти.

- В панели инструментов нажми кнопку **Штамп**, обведи колесо и перемести копию прямоугольника на место переднего колеса (рис. 3.18 правый). Когда колесо займёт своё место, щелкни мышью. Установи таким же образом колёса с другой стороны машинки. Далее данный инструмент следует отключить. Для этого перемести указатель мыши на панель инструментов и нажми любую кнопку.

Но нарисовать машинку недостаточно. Для правильной работы скрипта необходимо установить центр костюма. То есть определить точку, относительно которой будет поворачиваться персонаж. В нашем случае персонаж – это машинка.

- Установи машинку примерно по центру рабочего поля графического редактора. Для этого можешь воспользоваться горизонтальной и вертикальной полосами прокрутки, которые установлены справа и внизу рабочего поля. Выбери с помощью кнопок и удобный для тебя масштаб отображения. Нажми расположенную в левом нижнем углу окна графического редактора кнопку **Установить центр костюма** и наведи перекрестье из вертикальной и горизонтальной линий на центр машинки (рис. 3.18 правый нижний).
- Закрой графический редактор, нажав кнопку **ОК** в правом нижнем углу.

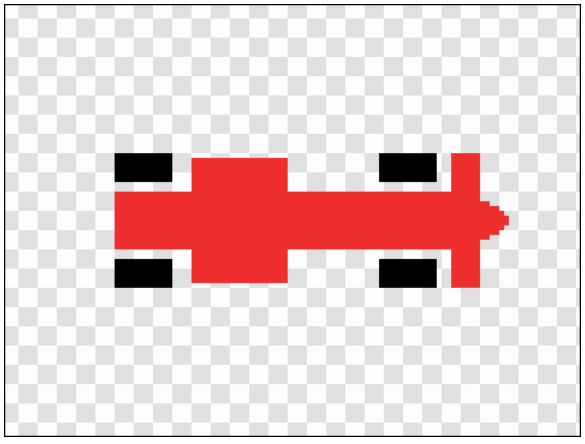


Рис. 3.19. *Это вполне реалистичная картинка болида Формулы 1, вид сверху*

Обрати внимание, если размер нарисованной машинки по отношению к дороге получился большим или маленьким, нужно изменить масштаб нарисованной модели. Для этого нужно открыть вкладку **костюмы**, нажать кнопку **Редактировать** и в окне графического редактора нажимать кнопку [⛶], если размер нужно увеличить, или кнопку [⛶], если размер нужно уменьшать.

Совет
Сначала нарисуй красный прямоугольник, а затем с помощью ластика удали маленькие детали.

Не забудь переименовать объект. Для этого дважды щёлкни мышью на имени **Костюм1** и введи с клавиатуры новое имя, например **Авто1**.

Для нашей машины уже разработана программа, состоящая из трёх скриптов управления. Верхний скрипт запускает и ведёт автомобиль до тех пор, пока он не отклонится от курса и не коснётся зелёной области. Два других скрипта используются для поворота автомобиля вправо на 15° или влево на 15° при каждом нажатии клавиши [←] или [→].

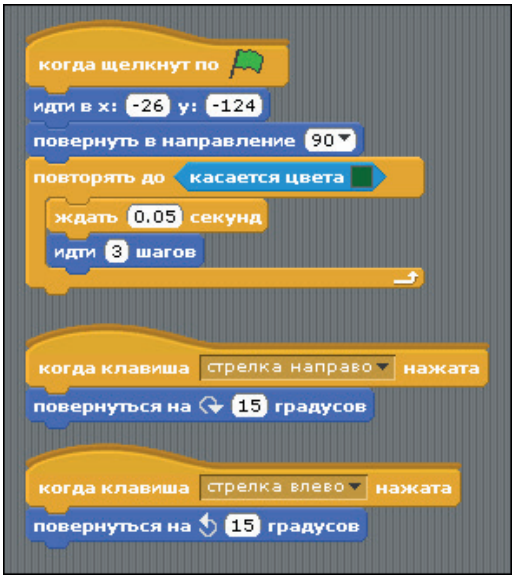


Рис. 3.20. *Три скрипта для управления автомобилем*

Собери скрипт, как показано на рис. 3.20. Обрати внимание: командный блок **касается цвета** находится в группе команд **сенсоры**.

При сборке верхнего скрипта необходимо в командном блоке **касается цвета** 

указать цвет области, находящейся вне трассы. На самом деле это сделать очень просто. После того как ты установишь команду **касается цвета** в командный блок **повторять до**, выполни следующие действия. Щёлкни мышью на квадратике образца цвета, который расположен в правой части командного блока. Вид указателя мыши изменится на  . Далее щёлкни мышью на зелёной области сцены. Цвет выбран (рис. 3.21 верхний).

Рис. 3.21. *Выбираем образец цвета*

Кроме того, нужно определить кнопки, с помощью которых будет управляться наш автомобиль. Для этого открой в среднем командном блоке **когда клавиша** открывающийся список и щёлкни мышью на строке **стрелка направо** (рис. 3.21 нижний).

Таким же способом назначь для нижнего командного блока клавишу **стрелка влево**.

Итак, игра создана. Проверь, как работает игра, и сохрани проект.

Вопросы по автогонкам

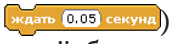
1. Как можно ускорить автомобиль? Для этого есть два варианта!
2. Что изменится, если из открывающегося списка командного блока **направление** вместо значения 90 выбрать 0?

Ответы ты найдёшь в конце главы.

Добавляем функции – газ и тормоз

Разумеется, все машины в гонках Формулы 1 не ездят с одинаковой скоростью. Ну разве что у них повреждён двигатель. Настоящий болид умеет ускоряться и тормозить. В этом разделе ты узнаешь, как это можно запрограммировать.

Что такое переменная?

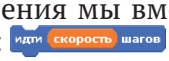
Взгляни ещё раз на скрипт. Машина едет с одинаковой скоростью. Скорость движения задаётся вложенными в командный блок **повторять до** командами **ждать** и **идти**. Согласно этим командам каждые 0,05 с (командный блок **ждать** 

) она сдвигается вперёд на три пикселя (командный блок **идти** 

). Если ты хочешь, чтобы машина ехала быстрее, тебе нужно увеличить длину шага. Первый способ увеличить скорость – это в командном блоке 

увеличить значение, то есть вместо значения 3 поставить, например, значение 6. Если же это значение уменьшить, например, до 1, скорость машинки уменьшится. Здесь всё логично. За каждые 0,05 с машинка проходит меньшее расстояние. Это можно представить так: ты по квартире идёшь мелкими шажками, называемыми «гусиным шагом», медленнее, чем большими шагами. То есть изменять скорость можно, изменяя длину шага. Если ты в своём автомобиле прибавишь газу, то скорость увеличится. Когда тормозишь, скорость падает. То есть, прибавляя газу ли тормозя, мы изменяем расстояние, пройденное за определённое время, или длину шага. Но весь вопрос в том, что значение длины шага задаётся в поле командного блока 

, и это значение в ходе игры нужно постоянно изменять.

Как же нам этот вопрос решить? К нам на выручку приходит *переменная*. В поле ввода значения мы вместо цифры вставляем переменную, например: 

. Как в программе Scratch работают переменные, мы подробно рассмотрим в следующем разделе.

Если прочесть команду, то звучит она довольно странно: идти – скорость – шагов. Что же под этим подразумевается? Представь себе переменную в виде ряда чисел, например от 1 до 10. Значение переменной (то есть собственно число, выдаваемое переменной в данный момент) может *изменяться*, изменяя количество шагов в командном блоке **идти**. Значение переменной равно количеству шагов, вводимых в поле ввода командного блока. Если значение переменой равно 1, значит, количество шагов за время, задаваемое в командном блоке **ждать**, будет равно 1, и скорость движения автомобиля будет маленькой. Если же значение переменной будет равно 10, то и количество шагов будет 10, и, соответственно, скорость движения будет очень большой.

Создаём переменную

Твой компьютер должен каким-то образом запомнить текущую скорость. Числа, как и скорость, сохраняются в переменной. Но изначально никаких переменных здесь нет, и ты должен их создать. Итак, поехали.

- Щёлкни мышью на кнопке коллекции блоков **переменные**.
- Щёлкни мышью на кнопке **Создать переменную**. На экране появится диалоговое окно **Имя переменной**.
- Введи в поле ввода имя переменной, например **Скорость**, и закрой окно, нажав кнопку **ОК**. В левой панели для вновь созданной переменной появятся новые блоки команд. Кроме того, в левом верхнем углу сцены появится поле, в котором ты увидишь текущее значение переменной (рис. 3.22). Если это поле тебе мешает, его можно скрыть. Для этого в левой панели убери флажок, расположенный слева от названия созданной переменной.

Расширяем скрипт

С помощью нового командного блока **Скорость** ты для объекта авто сможешь расширить функции скрипта. Обрати внимание: два скрипта для управления направлением движения остаются без изменения (рис. 3.23).

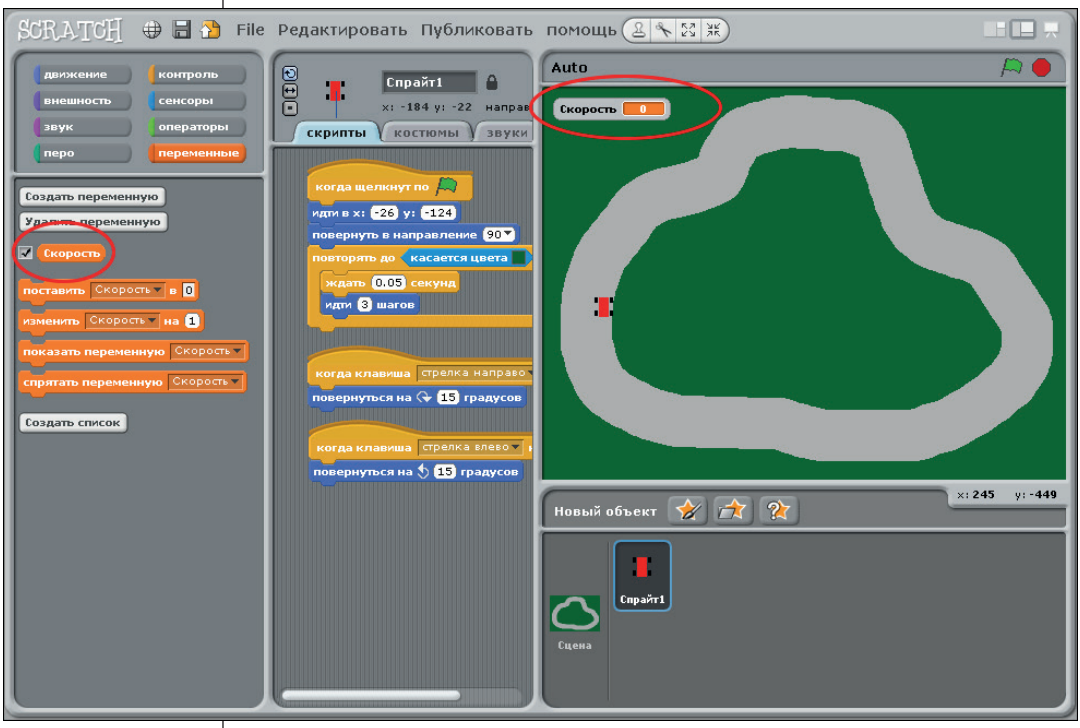


Рис. 3.22. *Переменная для обозначения скорости*

Как это работает

1. Когда значение переменой равно нулю, машина стоит на месте.
2. Скорость (значение переменной **Скорость**) – это количество шагов в секунду. Поскольку мы определили 20 движений в секунду, а время ожидания составляет 0,05 с, автомобиль движется вперед со скоростью 20 шагов. На рис. 3.24 показано, как построить эту функцию из готовых командных блоков. В первую очередь тебе понадобятся командные блоки из группы **операторы**.
3. С помощью этого командного блока скорость движения автомобиля увеличивается на 1 шаг при каждом нажатии клавиши ↑.
4. При каждом нажатии клавиши ↓ скорость машины будет уменьшаться на 1 шаг. Обрати внимание, если при нажатии клавиши ↓ в поле ввода **Скорость** в левом верхнем углу сцены появится отрицательное значение, машинка будет двигаться задним ходом.

Рис. 3.23. *Измененные и новые скрипты для гоночного болида*

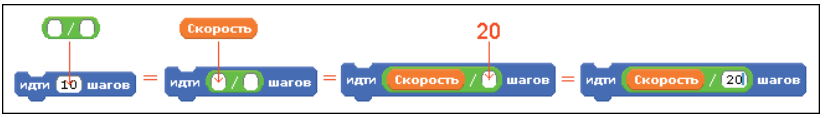


Рис. 3.24. *Вот так при помощи функции* ***Операторы*** *(например, операция* ***Деление****) можно построить математическое выражение*

Сохрани игру, назвав её, например, *Speed auto*.

На этом моделирование гоночных трасс с расширенными функциями можно закончить. В заданиях в конце главы ты найдёшь предложение по дальнейшему развитию этого проекта. Возможно, тебе захочется проложить трассу большего размера, она будет занимать несколько экранных страниц. Как это сделать, ты узнаешь, читая следующие главы.

Проект 6. «На помощь! Метеориты!»

Смысл игры заключается в защите Земли от падающих метеоритов. Ты, наверное, знаешь, что нашей планете постоянно угрожает падение этих объектов из космоса.

Итак, на земле стоит ракета. С помощью кнопок со стрелками ← или → ты можешь передвигать ракету вправо или влево. Чтобы ракету запустить, достаточно нажать клавишу **Пробел**. При встрече с ракетой метеорит вспыхнет и исчезнет. При этом на земле уже стоит следующая ракета, а сверху летит очередной метеорит.

- Создай новый проект. Для этого выбери команду меню **File** ⇨ **Новый**. В появившемся диалоговом окне **Сохранить текущий проект** нажми кнопку **Да**.
- Сохрани созданный проект под именем **Метеориты**.

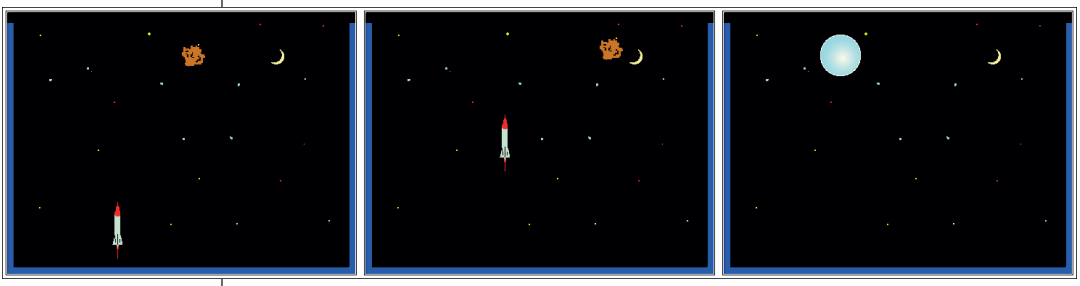


Рис. 3.25. *Запуск метеорита*

Сцена

В первую очередь тебе будет нужно нарисовать фон для сцены. При этом обрати внимание на следующий эффект: справа, слева и внизу фона должна быть тонкая полоса, цвет которой отличается от основного фона. Назначение этих полос следующее: когда метеорит касается этой полосы, он, подпрыгнув, исчезнет.

Ну что, приступаем к созданию игры?

- Нарисуй основной фон. Для этого в левой части правой нижней панели щёлкни мышью на значке **Фон**, в средней панели открой вкладку **фон** и нажми кнопку **Редактировать**.

- В рабочем окне графического редактора создай фон примерно как на рис. 3.26. Обрати внимание: ты этот фон можешь украсить изображением звезд и молодой луны (месяцем).

Совет

Звезды рисуются кистью. Ты в панели инструментов выбираешь инструмент – **Кисть**, из открывающегося меню размеров кистей, расположенного под панелью инструментов графического редактора, выбираешь размер кисти (размер кисти должен быть маленьким), в палитре выбираешь цвет и щёлкаешь указателем мыши на фоне. Один щелчок – одна звёздочка. Звезды можно сделать разных цветов и размеров. Чтобы нарисовать месяц, выбери инструмент **Эллипс** ○, в меню ниже панели нажми кнопку сплошной заливки (левая кнопка), выбери цвет (жёлтый), установи указатель мыши на фоне (в правом верхнем углу рабочего окна редактора), нажми левую кнопку мыши и, не отпуская левую кнопку мыши, перемести её вниз и вправо. На экране появится желтый круг. Когда луна приобретёт нужный размер, отпусти левую кнопку мыши. Выбери черный цвет и таким же образом нарисуй чёрный круг так, чтобы он частично перекрывал нарисованный ранее жёлтый круг.

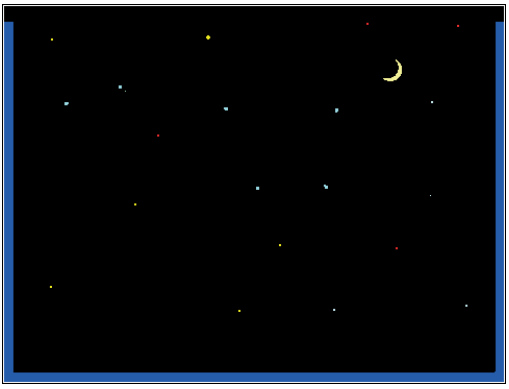


Рис. 3.26. *Фон*

Метеорит

Теперь перейдём к созданию персонажа, то есть метеорита.

Удали нашего кота, в панели инструментов правой нижней панели нажми кнопку, нарисуй новый объект, примерно как на рис. 3.27, и назови его метеорит.

У метеорита будет два костюма. Первый похож на обломок горной породы, а второй – на вспышку. Когда ракета появ-

ляется перед метеоритом, метеорит сразу же изменит свой костюм и ненадолго превратится во вспышку.

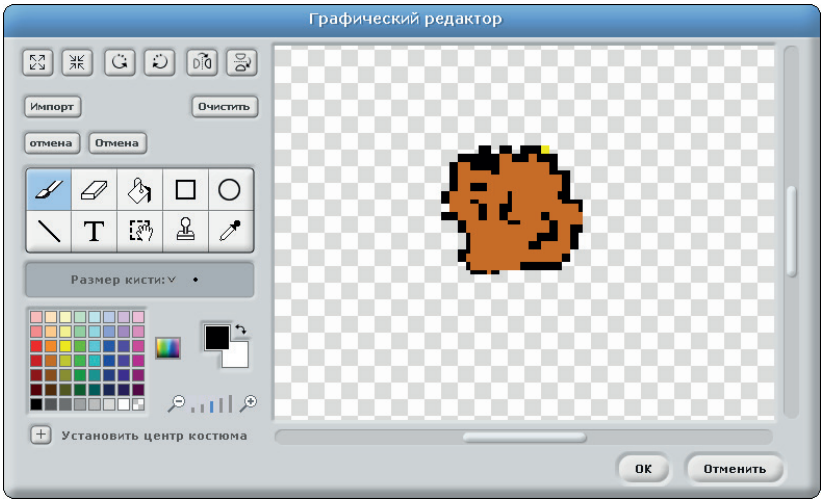


Рис. 3.27. *Костюм объекта* ***Метеорит***

Вспышку лучше всего нарисовать с помощью градиентной заливки. Делается это так:

- в верхней части вкладки **костюмы** (средняя панель) нажми кнопку **Рисовать**. На экране появится графический редактор;
- нарисуй с помощью инструмента **Эллипс** ○ круг, превышающий по размеру нарисованный ранее Метеорит (на рис. 3.28 кнопка № 1);
- выбери инструмент **Заливка** (на рис. 3.28 кнопка № 2);
- выбери в палитре сначала светло-голубой или любой другой светлый цвет. Вторым цветом выбери *прозрачный*. Этот цвет обозначен в виде шахматной доски серо-белого цвета (на рис. 3.28 кнопка в палитре № 3);
- теперь нажми кнопку **Градиентная заливка** (кнопка № 4 обведена красным кружком);
- установи указатель мыши в виде и залей круг (рис. 3.28).

  Обрати внимание, если у тебя в центре круга получается тёмное пятно, а по краям прозрачное, щёлкни мышью на изогнутой стрелке между двумя квадратиками с образцами выбранного цвета (кружок 5), чтобы цвета поменять местами, и залей круг заново. Если с первого

раза ничего не получилось, нажми кнопку **Очистить** и нарисуй вспышку заново.

- Сохрани созданный костюм под именем *Вспышка*.

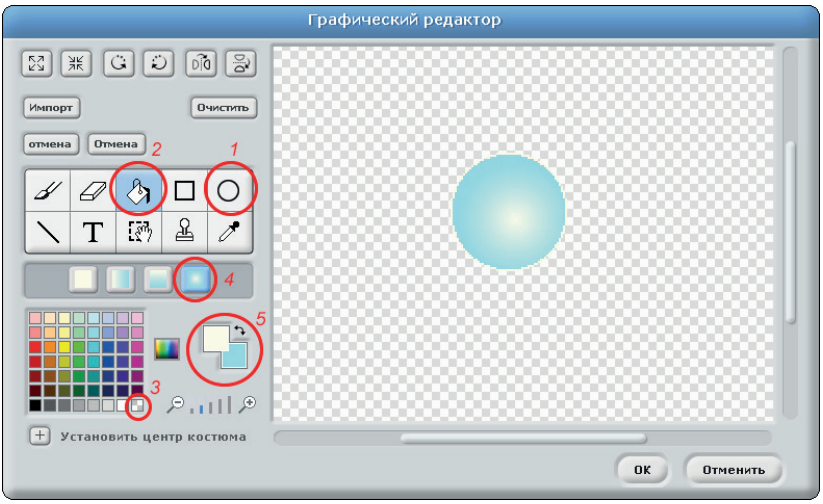


Рис. 3.28. *Вот так мы рисуем круг с градиентной заливкой от светло-голубого до бесцветного*

Теперь нам нужно нарисовать ракету.

Нажми в панели инструментов правой нижней панели кнопку и нарисуй в появившемся графическом редакторе ракету, например такую, как на рис. 3.29. Не забудь задать центр костюма. Сохрани костюм под названием **Ракета**.

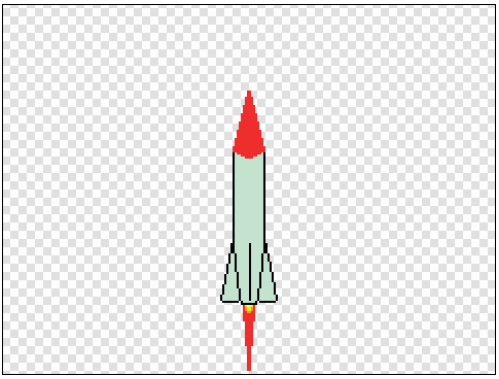


Рис. 3.29. *Ракета нарисована*

Теперь создай скрипт для падающего метеорита. Повинуясь этому скрипту, наш метеорит будет падать бесчисленное множество раз.

- На вкладке **костюмы** щёлкни мышью на изображении нашего метеорита, чтобы выбрать его.
- Щёлкни мышью на вкладке **скрипты**.
- Собери для метеорита скрипт так, как показано на рис. 3.29.

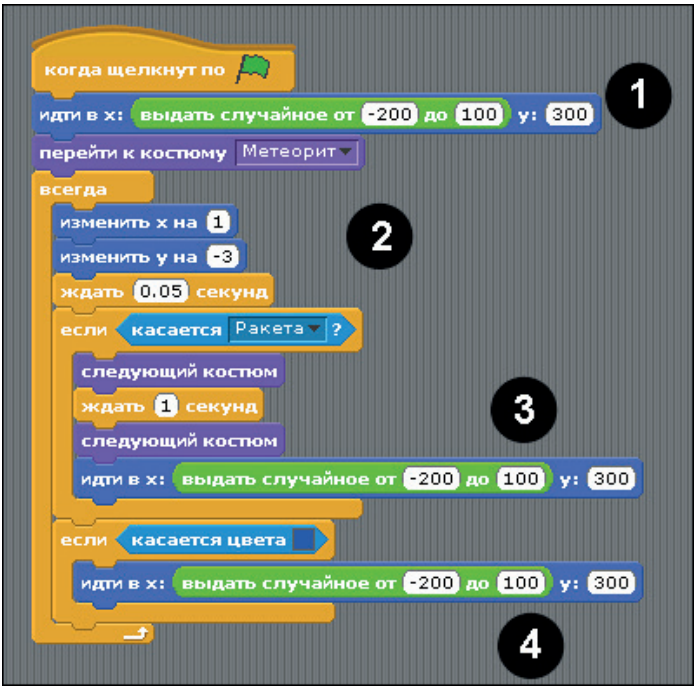


Рис. 3.30. *Готовый скрипт для объекта метеорит*

Как это работает

1. Командный блок **идти в** перемещает метеорит (объект) в верхнюю часть экрана. Позиция в горизонтальном направлении (координата **x**, задаваемая командой **выдать случайное от** и **до**) – это случайное число. Это значит, что при любом запуске метеорит будет в другом месте.
2. Метеорит движется вниз (значение **y** увеличивается).
3. Когда метеорит касается ракеты, он меняет свой костюм и становится вспышкой. Через секунду он вновь становится метеоритом и снова начинает перемещаться вниз со случайной позиции в верхней части. При касании ракеты цикл повторяется. Для игрока же всё выглядит так, будто появляется новый метеорит.
4. Метеорит при касании синей полосы справа, слева или внизу экрана должен исчезнуть и снова появиться

в верхней части экрана со случайной позиции. За эту функцию отвечает команда **касается цвета**, которая встроена в цикл **если** в нижней части скрипта.

Ракета

Вторым движущимся объектом является ракета. Ракету мы уже нарисовали (рис. 3.29). Теперь в правой нижней панели щёлкни мышью на значке этой ракеты и открой вкладку **скрипты**.

Собери скрипт для ракеты, как это показано на рис. 3.31.

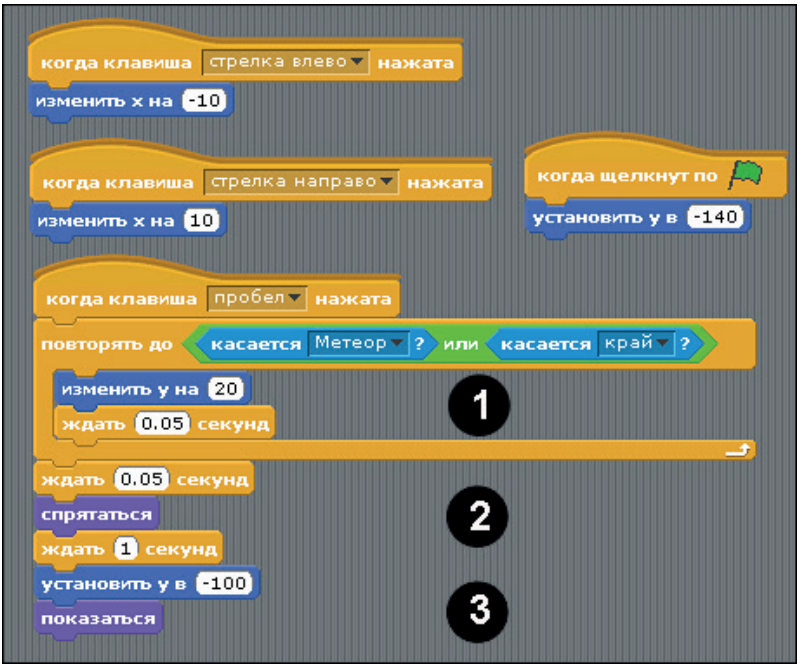


Рис. 3.31. *Скрипт для управление ракетой*

Как это работает

Два верхних скриптах **когда клавиша нажата** предназначены для сдвига ракеты вправо или влево. Скрипт **когда щёлкнуть по** предназначен, чтобы при нажатии на флажок ракета заняла положение внизу экрана. Дальше рассмотрим скрипты 1, 2 и 3.

1. Этот скрипт служит для того, чтобы с помощью клавиши **Пробел** запускать ракету. Возможно, для запуска ракеты нужно будет клавишу **Пробел** нажать дважды. При первом нажатии ракета появится над синей линией. А при повторном нажатии клавиши **Пробел** она по-

летит. Главное, чтобы после первого нажатия хвост или пламя ракеты не касалось синей линии. Если ракета будет касаться синей линии, на вкладке **костюмы** нажми кнопку **Редактировать** и с помощью кнопки [⇱] уменьши размер ракеты.

2. Этот скрипт делает следующий фокус. После столкновения с метеоритом ракета ждёт ещё немного, пока метеорит «догадается» о своём ударе с ракетой.
3. Через секунду ракета исчезает и потом снова появляется внизу.

Расширяем функции: время и числа

Первая версия программы, конечно, хорошая, но для настоящей игры в ней чего-то не хватает. А не хватает именно духа соревнования. Мы расширим правила игры. После запуска программы (клавиша **Пробел**) у нас есть 30 с, чтобы сбить как можно больше метеоритов. Когда время игры истечёт, на экране появится надпись **Конец**.

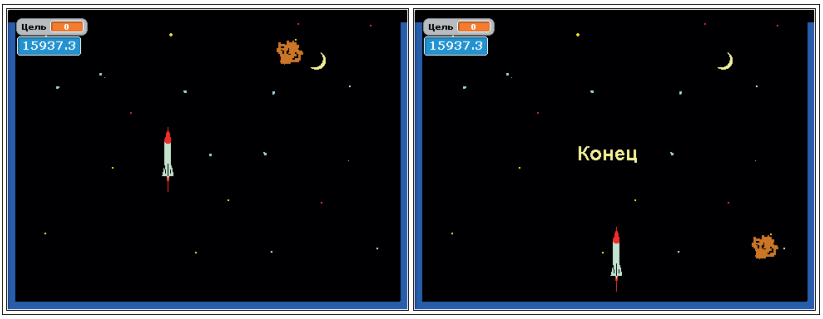


Рис. 3.32. *Расширенная версия игры с часами и количеством ударов*

Как же это запрограммировать? Тебе понадобятся часы и переменная с числами. При каждом попадании в цель эти числа будут увеличиваться на одну единицу. Кроме этого, тебе потребуется ещё один фон для сцены.

Переменная и время

- В коллекции блоков **переменные** щёлкни мышью на кнопке **Создать переменную**.
- В появившемся диалоговом окне **Имя переменной** введи имя **Цель** и нажми кнопку **ОК**.
- В левой панели окна Scratch появятся новые блоки.
- Установи флажок слева от элемента **Цель**. Возможно, данный флажок будет установлен по умолчанию.

Теперь в левом верхнем углу сцены появится поле со счётчиком **Цель**.

- В коллекции **сенсоры** ты найдёшь командный блок для управления таймером (см. рис. 3.32). Установи перед этим блоком флажок. В левом верхнем углу сцены появится элемент **Таймер** с полем, в котором отображается время.
- Щёлкни правой кнопкой мыши на таймере и выбери из появившегося меню строку **велик для чтения**.

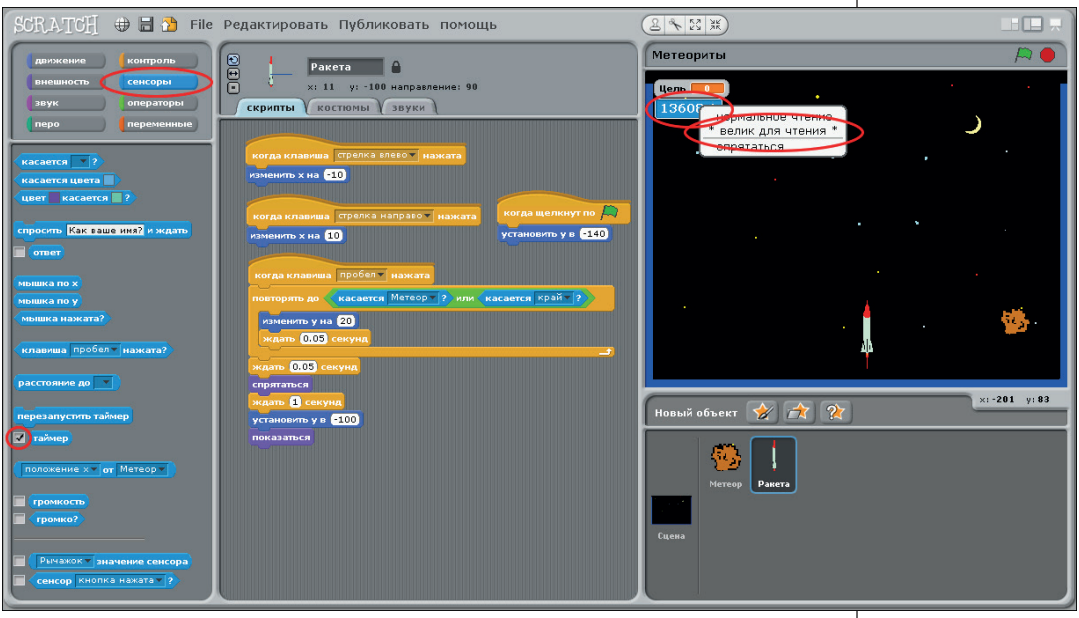


Рис. 3.33. *Выбираем функцию таймера*

Сцена

Когда время игры закончится, сцена должна продемонстрировать другой фон. Скопируй созданный ранее фон. Назови ранее созданный фон **Игра**, а только что скопированный – **Конец** и отредактируй копию фона, как на рис. 3.34.

На новом фоне впиши слово **Конец**. Для этого в графическом редакторе выбери инструмент **Текст**, выбери желаемый цвет и шрифт и напиши нужное слово (см. рис. 3.35).

Новая фоновая картинка появится через 30 с после начала игры. Для смены фона сцены следует создать отдельный скрипт. Сделать это можно так.

- В правой нижней вкладке щёлкни мышью на значке **Сцена** и перейди на вкладку **скрипты**.

- Собери скрипт, как показано на рис. 3.36.

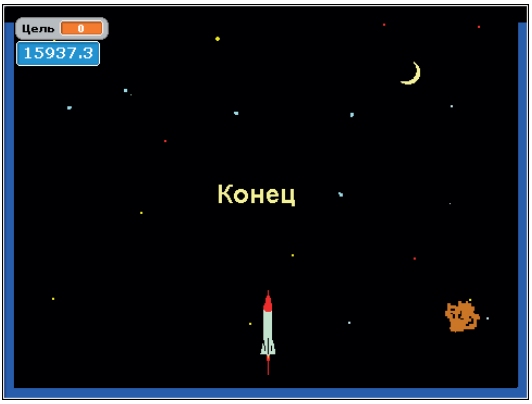


Рис. 3.34. *Второй фон для сцены*

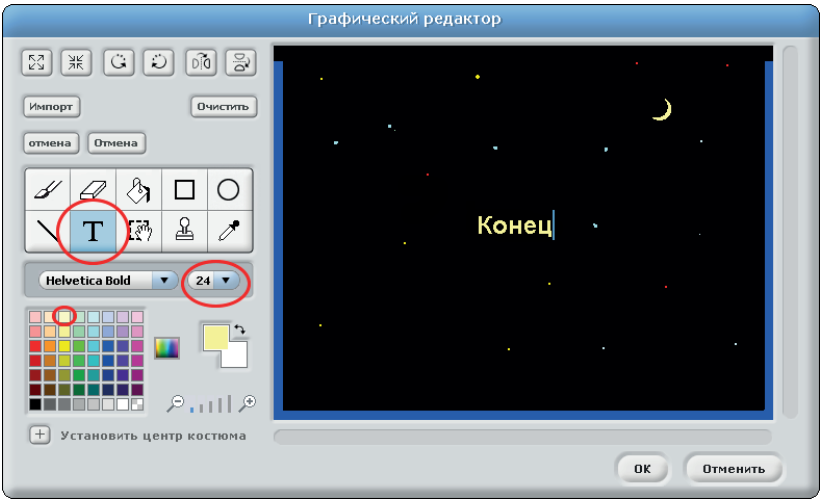


Рис. 3.35. *Выбираем инструмент* ***Текст****, цвет и размер шрифта*

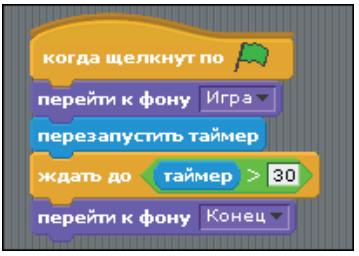


Рис. 3.36. *Создаём скрипт для сцены*

Я думаю, что этот скрипт можно и не объяснять.

Метеорит

Во время столкновения ракеты с метеоритом значение переменной **Цель** нужно увеличить на единицу. Для этого в скрипт метеорита тебе нужно будет лишь добавить одно указание и в конце расширить скрипт.

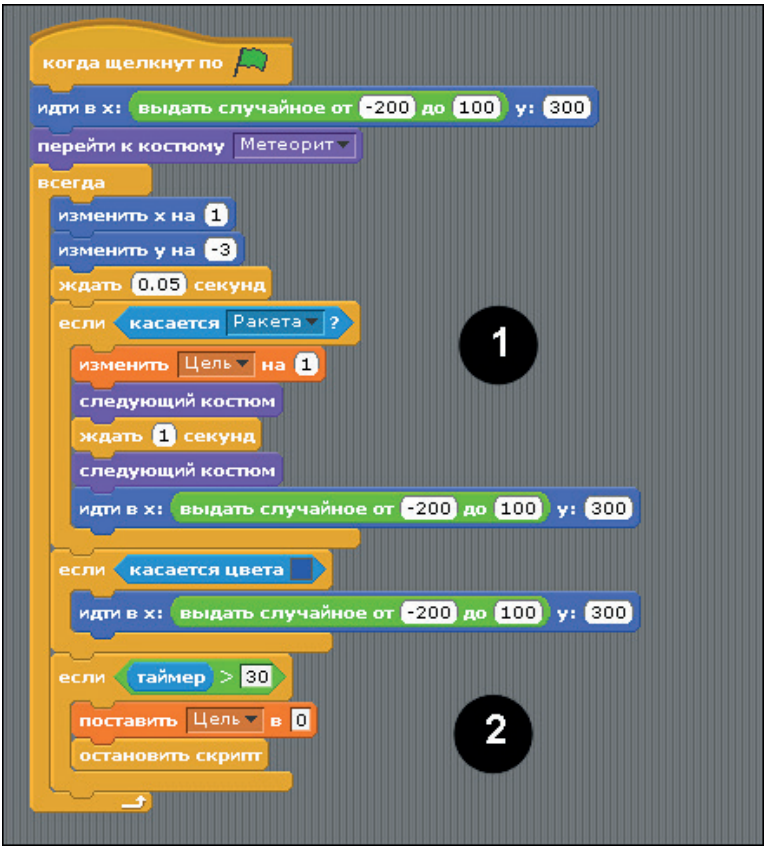


Рис. 3.37. *Расширяем скрипт для метеорита*

Как это работает:

1) команда **изменить Цель на 1** увеличивает значение сбитых метеоритов;
2) нижний блок команд по прошествии 30 с обнулит счётчик и остановит скрипт.

Ракета

По окончании игры ракета должна стать невидимой. Для этого тебе необходимо изменить два её скрипта.

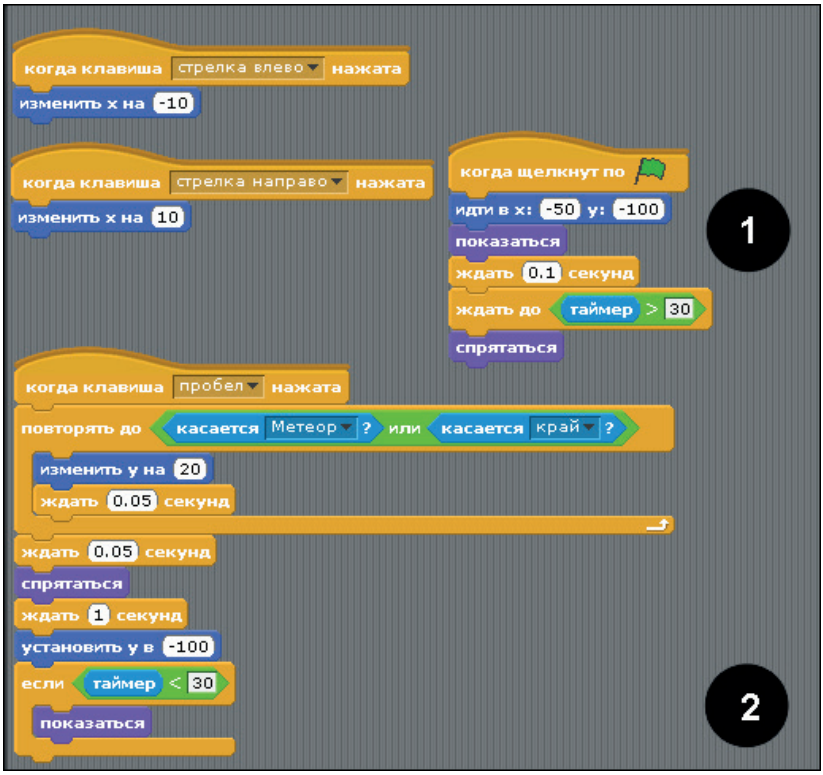


Рис. 3.38. *Меняем оба скрипта для ракеты*

Как это работает

1. Первый скрипт запускается один раз в начале игры. В этой строке присутствует одна хитрость, к которой я тоже не сразу пришёл. Без этой дополнительной команды ожидания программа будет работать некорректно. Функция ожидания гарантирует, что скрипт сцены перезапустит часы. В противном случае может произойти так, что часы получат ещё большее значение, и тогда ракета «решит», что игра закончена, и вновь станет невидимой.
2. Прежде чем ракета станет видимой, ей нужно проверить, не завершилась ли игра. Когда время игры закончится, ракета исчезнет из поля зрения, став незаметной.

Вопросы по метеоритам

1. Ты можешь собрать из блоков скрипт для сцены. Но команды для управления сценой немного отличаются от команд для управления объектами. Какие из команд *не* предназначены для сцены?

2. Какая разница между


и


?

Ответы ты найдёшь в конце главы.

Расширяем функции – даёшь больше метеоритов!

С одним-единственным метеоритом игра выглядит совсем простой. Сделать её интереснее можно при помощи следующих расширений:

- вместо одного можно увидеть *множество метеоритов*, с треском падающих на планету;
- когда один метеорит касается земли или края картинки, ты получаешь два штрафных очка.

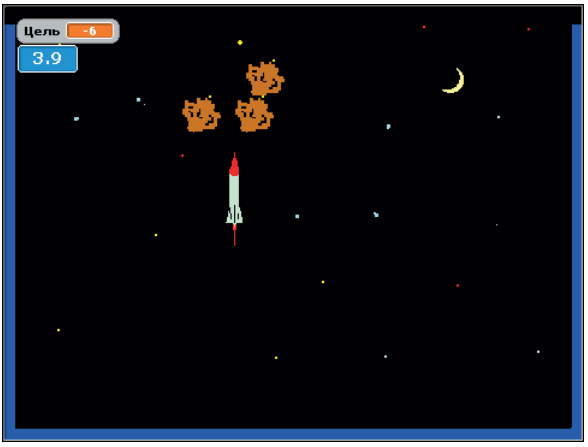


Рис. 3.39. *Множество метеоритов, падающих в случайном порядке*

Метеориты

Измени скрипт метеорита, как показано на рис. 3.40. Добавь три обведённых цветом указания в это место. В чём смысл новых команд? Чуть позже мы вернёмся к этому вопросу.

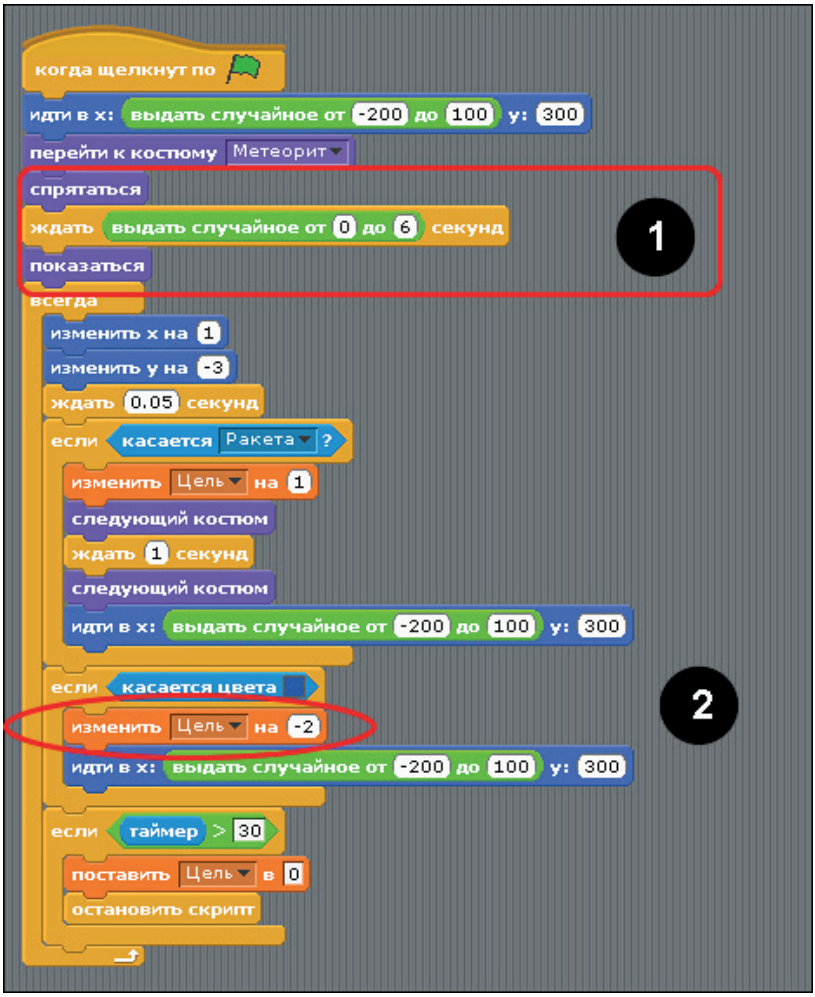
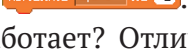


Рис. 3.40. *Добавляем команды в скрипт метеорита*

В конце скрипта нужно внести кое-какие изменения. Если метеорит столкнулся с краем, количество очков снизится. Добавь в нужное место (на рис. 3.40 № 2)  .

Протестируй программу. Всё хорошо работает? Отлично! Теперь можешь дважды скопировать свой метеорит. Наведи курсор мышки на символ объекта и выбери команду **дублировать**.

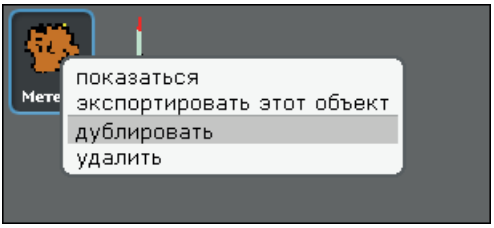


Рис. 3.41. *Копируем объект*

Если ты ещё раз продублируешь объект, то получишь три одинаковых метеорита. Дай всем трём названия М1, М2 и М3.

Рис. 3.42. *Три метеорита*

Теперь протестируй свой проект. Нажми на зелёный флажок . Метеориты начнут падать с неба. За то, что метеориты будут стартовать в разное время, а не одновременно, отвечает строка **ждать** со встроенной функцией **выдать случайное от и до**. Это выглядит интереснее, чем одновременное падение.

Важно!
Если ты хочешь изменить скрипт метеорита позже, то действуй следующим образом:

- удали все метеориты, кроме одного, кликнув правой кнопкой мыши и выбрав **удалить**;
- измени скрипт одного метеорита и затем протестируй программу;
- продублируй изменённый скрипт и дай копии правильное имя.

Ракета

Ну а теперь ракету можно направить на сближение с тремя разными объектами. Поэтому необходимо изменить тест на столкновение.

В нижней части рисунка подробно показано, как собрать эту функцию.

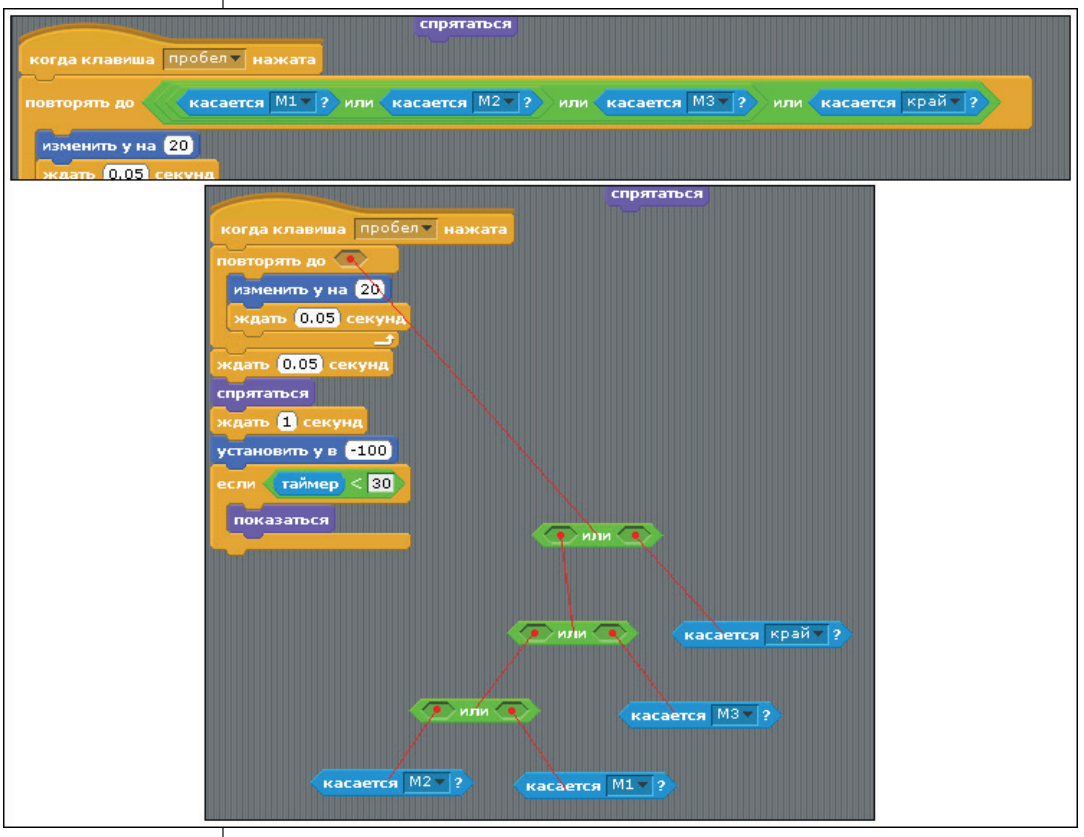


Рис. 3.43. *Редактируем скрипт на столкновение в команде повторения*

Студии программы Scratch

Наверняка ты уже посещал сайт Scratch в интернете? Если ещё нет, вот его адрес: http://scratch.mit.edu. Этот сайт – своего рода виртуальный музей, в котором содержатся миллионы Scratch-программ, которые создавались людьми по всему миру. Все проекты размещены в студиях. Студии – это так называемые «выставочные залы» музея, которыми управляют кураторы. Ты можешь посмотреть эти Scratch-проекты и прямо там их попробовать, но только онлайн, не на Raspberry Pi. Прости, но это невозможно. Для просмотра и тестирования проектов твой компьютер должен иметь последнюю версию флеш-плеера, а для Raspberry Pi такая программа слишком тяжела. Словом, если ты хочешь посетить сайт Scratch, то лучше воспользуйся более мощным компьютером (это может быть ноутбук, планшет или ПК). Посмотреть это стоит в любом случае.

Стать скретчером

Возможно, тебе захочется загрузить собственный проект, чтобы твои друзья могли его увидеть. Это возможно сделать с помощью Raspberry Pi. Для этого тебе всё равно придётся зарегистрироваться на сайте Scratch и войти в сообщество так называемых «скретчеров». Щёлкни мышкой **Join Scratch** вверху на главной странице Scratch (если страница на английском). На русском ты увидишь надпись **Присоединиться**.

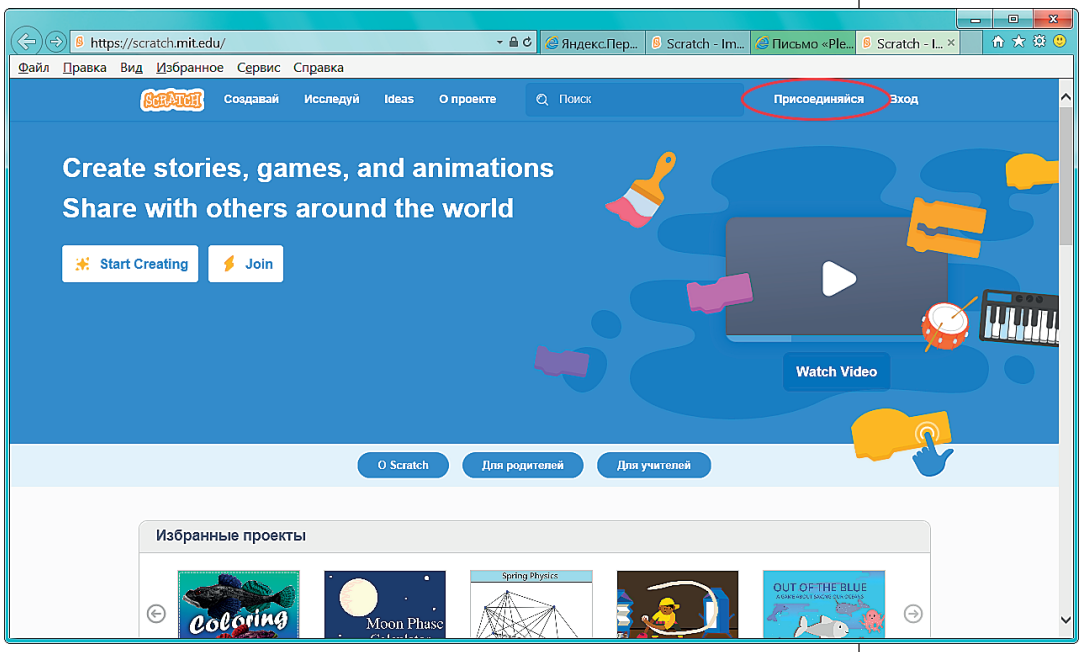


Рис. 3.44. *Нажав эту кнопку, ты регистрируешься на сайте Scratch*

Все остальные действия на сайте подробно описаны. Ты придумываешь и вводишь свой логин (ник). Никогда и никому не сообщай свои личные данные – кто ты и где живёшь. Но программа в любом случае потребует при регистрации ввести некоторую информацию (месяц рождения, страна и электронная почта), которая будет использована в научных целях. В конце концов, MIT – это университет. Но, так как ты несовершеннолетний, перед тем как зарегистрироваться, обсуди это со своими родителями. Получилось? Ну что ж, теперь ты участник международного сообщества Scratch. Внимательно прочти правила. Одно из самых важных гласит: «Будь почтительным», т. е. уважай каждого участника.

Став авторизованным скретчером, ты можешь загружать, скачивать и изменять собственные проекты, создать свою студию и ещё многое-многое другое.

Загружаем проект

На сайте программы ты можешь показать миру свои проекты. Выбери в строке меню программы Scratch команду **Публиковать ⇨ Опубликовать проект в сети**. Появится диалоговое окно **Загрузка на сервер Скретч** (рис. 3.45). Введи туда свой логин и пароль и не забудь добавить пару комментариев к своему проекту. Теперь жми **ОК**.

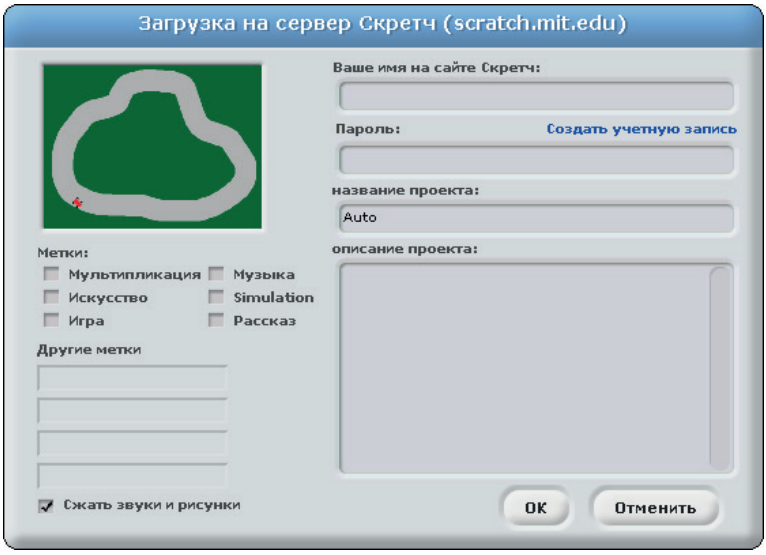


Рис. 3.45. *Публикуем проект в интернете*

На сайте Scratch ты можешь своим проектом управлять и поместить его в какую-нибудь студию (папку). Щёлкни мышкой в верхнем углу, где твоё имя, и выбери из меню **Мои работы** (если страница на английском, то **My stuff**).

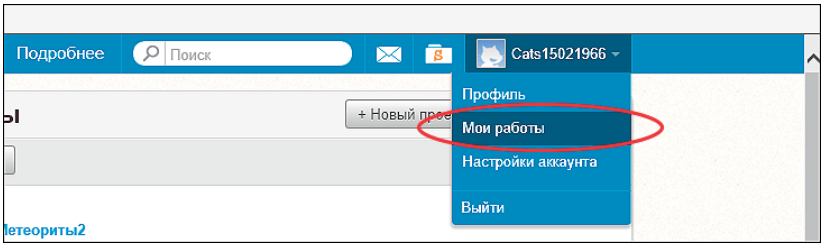


Рис. 3.46. *Свои проекты ты найдёшь в меню под своим именем*

Делаем ремикс проекта

Одна из идей Scratch состоит в том, чтобы вдохновляться работами других людей. Если какой-то проект тебя особенно заинтересовал, ты можешь его скачать и потом внести в него свои изменения. Это называется *сделать ремикс*. Но для скачивания тебе потребуется обычный компьютер и браузер со встроенным флеш-плеером. В общем, поищи проект, который тебе захочется изменить. На странице проекта в правом верхнем углу нажми вкладку **Войти внутрь проекта**. Ты окажешься в редакторе Scratch. Затем выбери кнопку **Ремиксовать** и скачай проект. Ты можешь сохранить его на флешке, а потом перенести на Raspberry Pi.

Задания

Загрузка на астероиды

Космический корабль медленно опускается вниз. С помощью клавиш ← и → ты можешь регулировать его направление влево и вправо. Нажми **Пробел**, чтобы повернуть корабль на 90°. Корабль должен приземлиться в туннель точно на астероид и своей верхушкой коснуться голубой линии на земле (рис. 3.47 справа). Когда корабль в тоннель не попадает и касается астероида (он обозначен жёлтым), происходит авария (рис. 3.47 в середине).

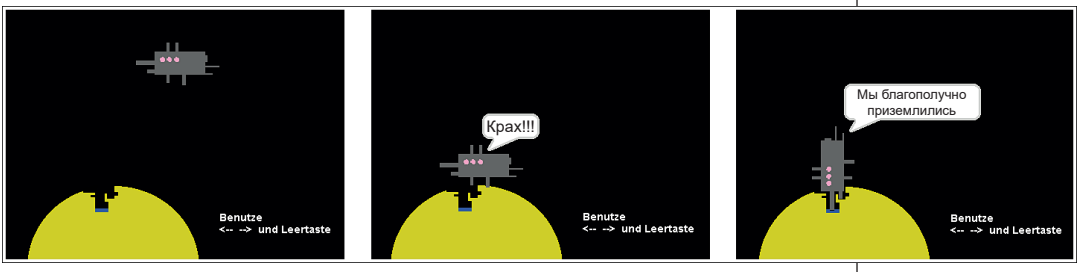


Рис. 3.47. *Скриншот проекта – плохой и удачный конец игры*

Совет

Тебе потребуется фон с астероидами и объект для космического корабля. Для скриптов корабля можешь воспользоваться блоками из рис. 3.48.

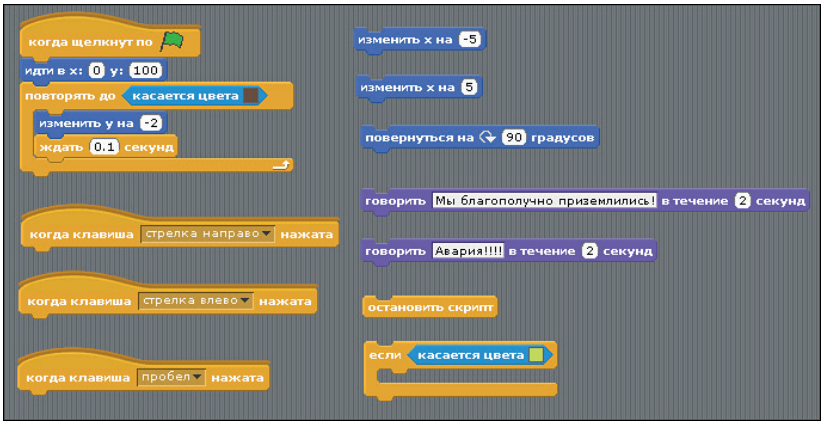


Рис. 3.48. *Из этих блоков ты можешь собрать четыре скрипта для управления своим космическим кораблём*

Мокрица в лабиринте

Голодная мокрица обитает в лабиринте, состоящем из множества подземных ходов, и ищет, чем бы ей там поживиться. Через все ходы она передвигается самостоятельно. Как только она найдёт что-то съестное (обозначено красным на рисунке), то произносит какую-нибудь фразу.

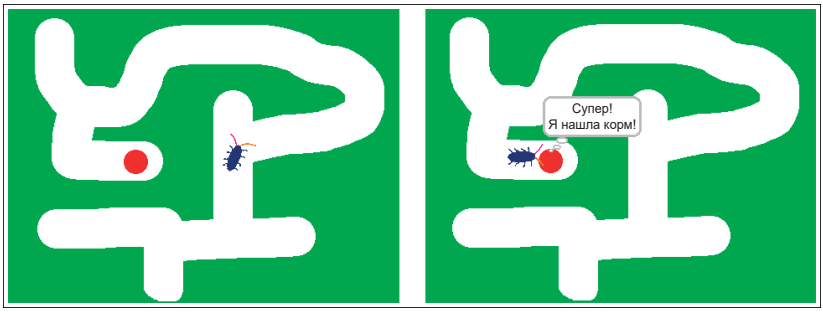


Рис. 3.49. *Мокрица ползает по лабиринту в поисках съестного*

Совет

Здесь тебе понадобится фон для сцены и объект с двумя костюмами для ползающей мокрицы. Этот зверёк является вымышленным, поэтому ему не обязательно выглядеть как настоящая мокрица. (Кстати, а ты знаешь, сколько на самом деле ног у настоящей мокрицы?) Оба «мокричных» костюма должны немного отличаться друг от друга. Если их всё время менять, то будет казаться, что усики и лапки движутся (для этого воспользуйся командой **следующий костюм**). Ещё один важный момент – усики. Они должны

иметь разный цвет. С помощью усиков можно настроить мокрицу так, словно она натыкается то на левую, то на правую стену. Она также может реагировать на то, бежит ли прямо на стену или от неё.

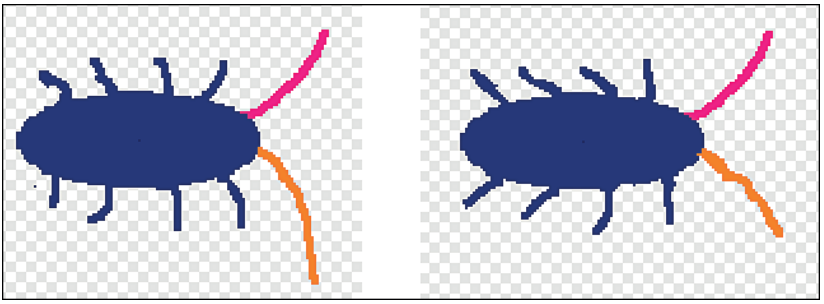


Рис. 3.50. *Костюмы для мокрицы*

На рис. 3.50 показан наполовину готовый скрипт для нашего насекомого. Используй блоки на правой стороне и установи их в нужное место.

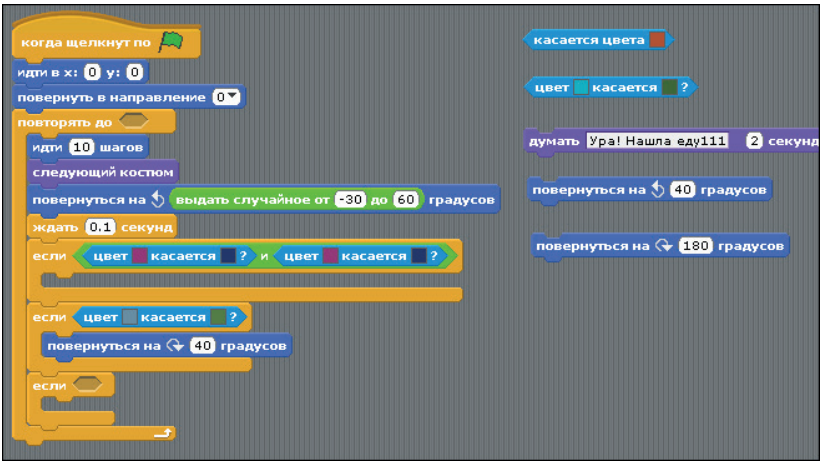


Рис. 3.51. *Из этих блоков ты можешь собрать скрипт для мокрицы*

Решение задач

Посадка на астероидах. Как это работает?

1. Если щёлкнуть мышью на зелёном флажке, космический корабль сначала вернётся в свою начальную позицию.

2. Корабль двигается вниз сам по себе (при этом его значение по координате **y** уменьшается). После каждого шага нужно проверить, коснулся он астероида или нет (астероид окрашен в жёлтый цвет).

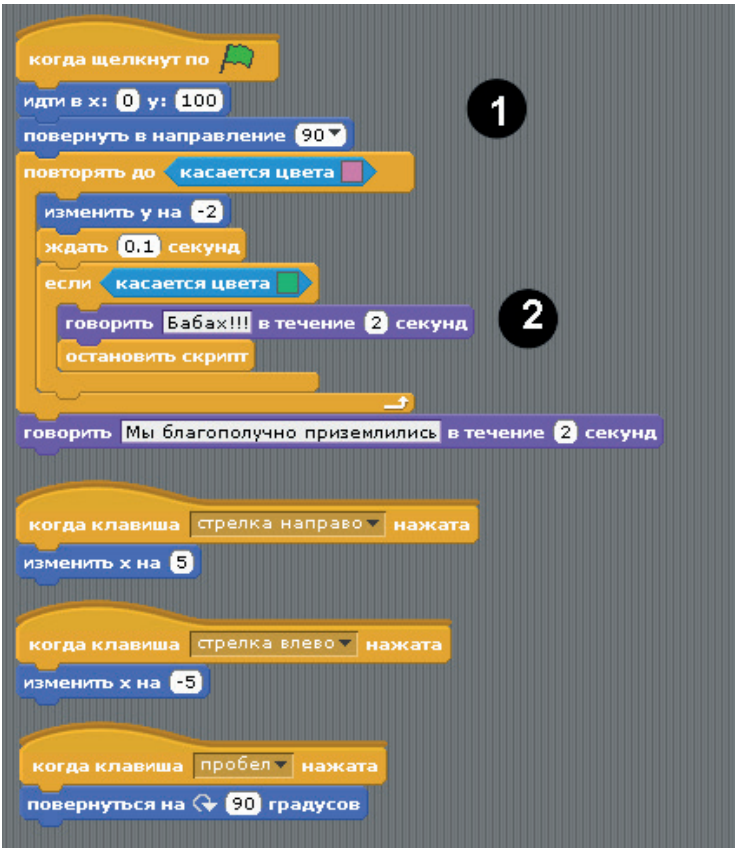


Рис. 3.52. *Это скрипты для управления космическим кораблём*

Мокрица в лабиринте. Как это работает?

1. Цикл повторяется, пока мокрица не коснётся своей пищи.
2. Мокрица поворачивается в произвольном порядке под углом между 30° влево и 30° вправо.
3. Когда оба усика коснутся цветной зоны, мокрица столкнётся со стеной лоб в лоб. После этого она повернёт обратно.
4. Если же она столкнулась левой стороной, то поворачивается вправо.

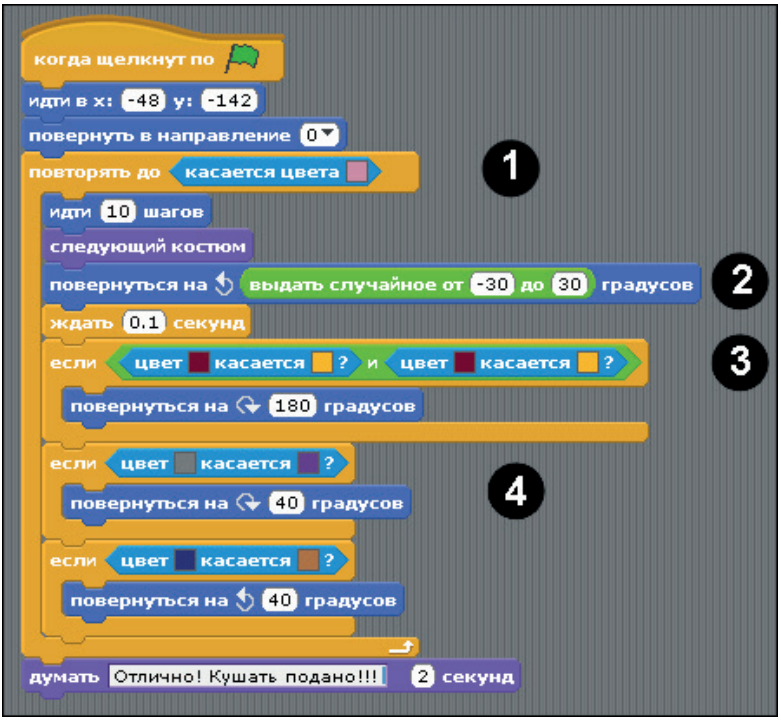


Рис. 3.53. *Скрипт для управления движениями мокрицы*

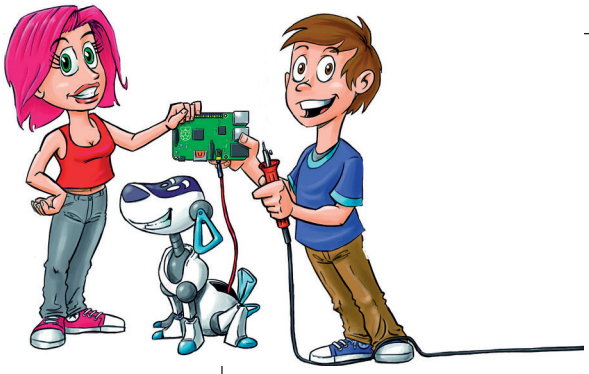
Ответы на вопросы

Автогонки

1. Для того чтобы выбрать команду **идти**, тебе понадобится большее количество шагов. Или просто вставь в команду **ждать** меньшее время.
2. Машина развернется передком к верхней границе сцены.

Метеориты

1. Для сцены в категории **движение** не существует никаких команд. И это вполне логично, ведь сцена у нас остаётся неподвижной. Коллекции команд **внешность**, **сенсоры** и **перо** уже содержат команды для сцены, которые отличаются от команд для объектов.
2. Если в ячейке **от** и **до** ты введёшь десятичное число, то случайные числа тоже станут десятичными, правда, с двумя цифрами после запятой, например 0,15.



4 Мультяшные истории

В этой главе речь пойдёт создании анимации, то есть мультиков, с использованием нескольких фигур и нескольких фонов. Ты научишься создавать диалог между двумя фигурами, мультипликационное видео с несколькими сценами, а также придумывать викторины с возможностью случайного выбора вопросов.

Проект 7. Шуточный мультфильм

На рис. 4.1 мы видим следующее. На сцену выходит собачка и что-то говорит. При этом она открывает и закрывает рот, а второй пёсик ей отвечает. То есть мы видим своеобразный диалог, в конце которого звучит какая-нибудь шутка. В чём особенность такого мультфильма? Мы видим двух персонажей, реагирующих друг на друга. Никто никого не перебивает, а ждёт своей очереди. Ответ появляется в нужное время и только тогда, когда прозвучал вопрос. Не раньше. Это называется синхронизацией.

Как происходит синхронизация в скриптах Scratch? К этому мы вернёмся чуть позже, а пока начнём с фигур.

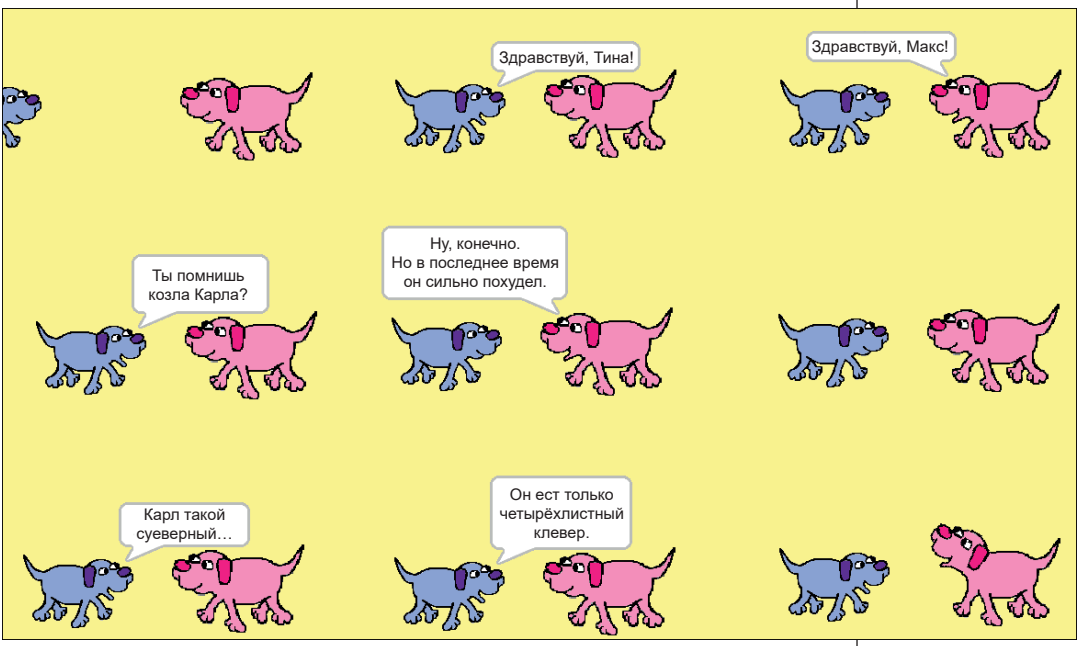


Рис. 4.1. *Мультяшный диалог между двумя персонажами*

Рисуем мультяшных героев

Конечно же, ты можешь нарисовать наших героев самостоятельно в графическом редакторе. Но не менее интересно воспользоваться готовым рисунком и его немного преобразовать.

В твоём Raspberry Pi есть готовый набор фигур. Если тебе там ничего не приглянулось, то поищи интересные картинки в интернете. Если ты планируешь опубликовать свой проект на сайте Scratch, при копировании картинок в интернете учти, что закон об авторских правах необходимо соблюдать.

Creative Commons

Если собираешься публиковать свой проект в интернете, то можешь использовать только общедоступные картинки или картинки, которые правообладатель использовать разрешил. Много изображений, находящихся в сети, имеют лицензию американской организации Creative Commons. Тебя могут заинтересовать следующие категории лицензий:

- `cc 0`. Картинка находится в свободном доступе, и ты можешь использовать её по своему усмотрению;

- cc by. При использовании необходимо указать автора (*by* в переводе с англ. «от»);
- cc by-sa. Кроме указания автора картинки, тебе необходимо открыть доступ другим пользователям к своему Scratch-проекту (пометка *sa* означает в англ. *share alike*, то есть «поделиться с другими»);
- cc by-nc. Нужно указать автора, при этом твой проект не может быть использован в коммерческих целях (*nc* означает *non commercial*, то есть «некоммерческое использование»);
- cc by-nc-sa. Эта категория лицензии объединяет в себе три последних: нужно указать автора изображения, открыть доступ к проекту другим пользователям и не пытаться заработать на своём проекте деньги, то есть не использовать его в коммерческих целях.

Как искать картинки

Картинки, имеющие лицензию Creative-Commons, ты можешь найти в расширенной версии поиска в Google. Делается это так:

- заходишь на сайт https://www.google.ru;
- в правом верхнем углу нажимаешь вкладку **Картинки;**
- вводишь поисковый запрос. В Google поиск лучше вести, вводя поисковые запросы на английском языке, потому что англоязычных сайтов больше, чем русскоязычных;
- в панели инструментов поисковика нажми кнопку **Инструменты**. Ниже появятся открывающиеся списки **Размер**, **Цвет**, **Права на использование**, **Тип**, **Время** и **Ещё**. Щёлкни мышью на заголовке открывающегося списка **Права на использование** и выбери строку **С лицензией на использование и изменение** (см. рис. 4.2).

Сохраняем картинку в папке проекта

Когда нужный тебе рисунок будет найден, сохрани его в папке проекта. Это делается так:

- щёлкни правой кнопкой мыши на найденной картинке. Рядом с курсором появится список команд, который называется *контекстное меню*;
- выбери из появившегося контекстного меню команду **Сохранить изображение как**;
- на экране появится диалоговое окно **Сохранение рисунка**. Здесь тебе нужно выбрать и открыть папку с проектом и сохранить в ней найденную картинку.

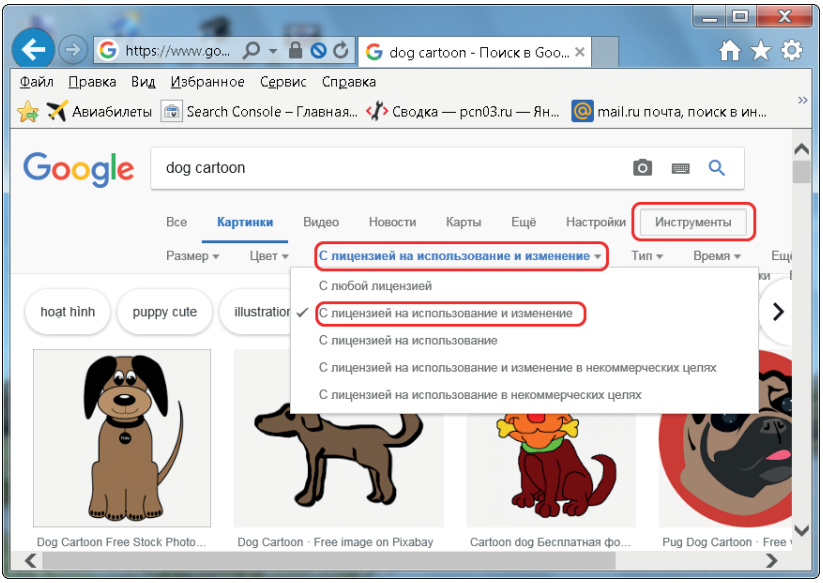


Рис. 4.2. *Поиск картинок с правом их использования и изменения*

Создаём Scratch-проект из картинки

В Scratch ты можешь создать новый объект, который будет носить костюм найденной картинки. Порядок действий следующий:

- создай новый проект Scratch и удали кота;
- назови вновь созданный проект, например, **Dogs** и сохрани его в отдельной папке, например в **Dogs project**.

Чтобы использовать в проекте картинку, найденную в интернете, выполни следующие действия:

- в панели инструментов правой нижней панели, расположенной под сценой, нажми кнопку (рис. 4.3);

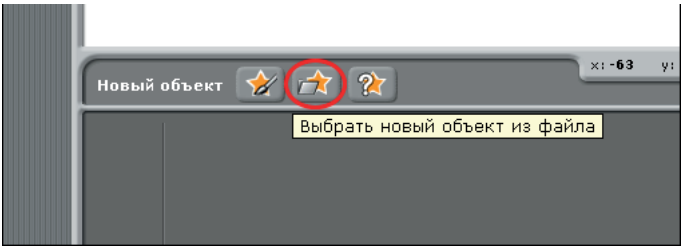


Рис. 4.3. *Открываем диалоговое окно* ***Новый объект***

- на экране появится диалоговое окно **Новый объект**, в котором ты увидишь шесть папок, где хранятся картинки, их ты сможешь использовать в своих проектах.

Каждая папка имеет название, соответствующее тем картинкам, которые в ней хранятся (рис. 4.4);

Рис. 4.4. *Диалоговое окно* ***Новый объект*** *открыто*

- в левой части диалогового окна нажми вторую сверху кнопку **pi** (эта кнопка открывает твой домашний каталог) и открой папку, в которой хранится твой проект;
- щёлкни мышью на найденном в интернете рисунке и нажми кнопку **OK**.

Для нашего проекта изображения собачек Тины и Макса загружать из интернета не придётся, так как их можно найти в папке **Animals** (Животные). Чтобы выбрать изображение собачки, нажми в левой части диалогового окна четвёртую сверху кнопку – **Костюмы**, дважды щёлкни мышью на папке **Animals** и, прокручивая вертикальную полосу прокрутки, найди нужное изображение. Щёлкни мышью на найденном изображении и нажми кнопку **OK**. Новый объект сразу же появится на сцене.

Как уже говорилось ранее, имя персонажа вводится в поле ввода в верхней части средней панели над ярлыками **скрипты**, **костюмы** и **звуки**. Имя вводится вместо предлагаемого имени по умолчанию **Спрайт1**. На рис. 4.5 это поле ввода обведено красным овалом. Обрати внимание: сейчас в средней панели активна вкладка **скрипты**. Цвет ярлыка этой вкладки отличается от цвета ярлыков **костюмы** и **звук**.

- Загрузи на сцену нужное изображение собаки и назови её Максом.

Костюм этого объекта ты сможешь видоизменить. Это должна быть маленькая собачка.

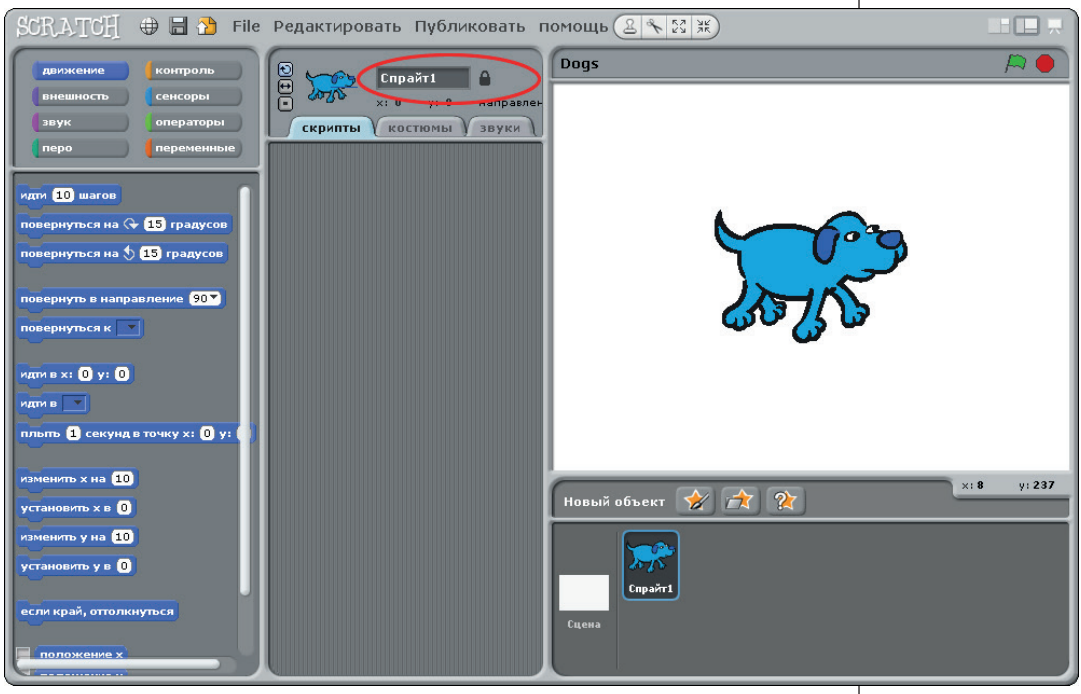


Рис. 4.5. *Поле, в которое нужно ввести имя объекта*

- Щёлкни мышью на ярлыке вкладки **костюмы**. Выбранная вкладка станет активной.
- Нажми кнопку **Редактировать**. Откроется графический редактор, в котором ты сможешь изменить объект.

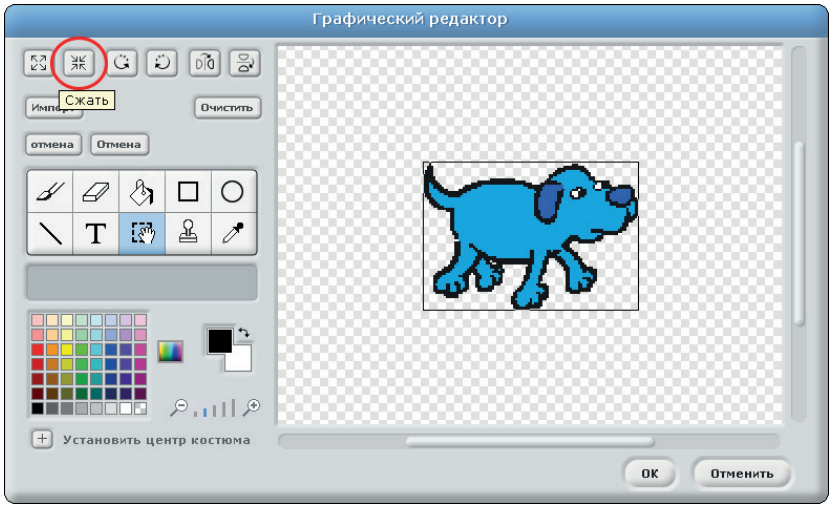


Рис. 4.6. *Сжимаем картинку*

Картинка на самом деле большая, и её нужно уменьшить примерно вдвое.

- Нажми три раза кнопку **Сжать**, чтобы уменьшить изображение (рис. 4.6).

Далее нам нужно вырезать изображение из фона. Но это нужно делать тогда, когда вокруг объекта действительно находится фон. Если вокруг объекта фон в виде бледного шахматного поля (как на рис. 4.6), ничего делать не нужно. Если же персонаж находится на цветном фоне, в этом случае фон нужно удалить. Для этого выполни следующие действия:

- щёлкни мышью на образце цвета в правом нижнем углу палитры (на рис. 4.7 этот образец цвета обведён красным кружочком № 1). Выбранный цвет ты увидишь в левом верхнем прямоугольнике справа от палитры цветов (обведён кружочком № 2);
- в панели инструментов графического редактора нажми кнопку **Заливка**;
- установи указатель мыши, принявший вид выливающегося ведра, на фоне (на рис. 4.7 красный кружочек № 3) и щёлкни мышью. Фон будет удалён (рис. 4.7);
- когда редактирование будет завершено, сохрани изменения. Для этого нажми в правом нижнем углу окна графического редактора кнопку **ОК**.

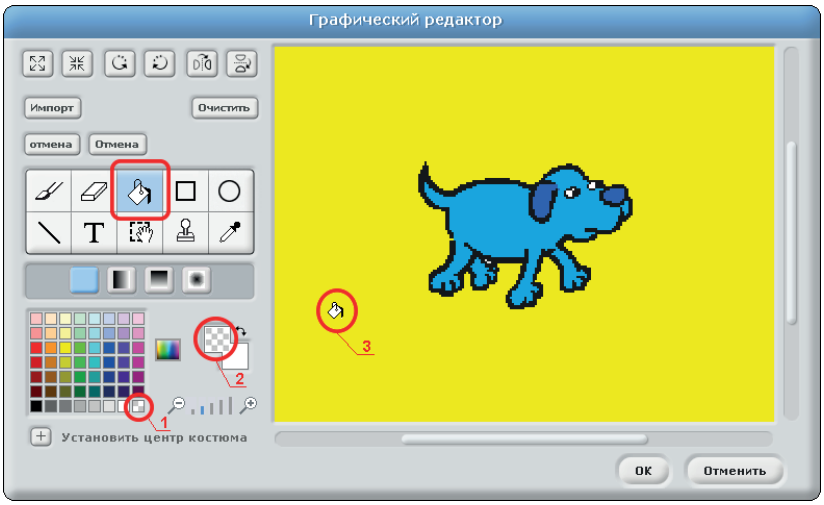


Рис. 4.7. *Удаляем фон*

Теперь для персонажа под именем Макс нужно нарисовать ещё один костюм, где у собачки рот открыт (Макс произ-

носит фразу). Для этого тебе нужно скопировать костюм, скопированный костюм переименовать, открыть его в графическом редакторе и нарисовать открытый рот:

- на вкладке **костюмы** нажми кнопку **Копировать**. Под первым костюмом появится его копия;
- переименуй оба костюма. Верхний костюм назови **Max1**, а его копию – **Max2**;
- под полем ввода имени второго костюма (Max2) нажми кнопку **Редактировать**. Костюм Max2 откроется в графическом редакторе;
- с помощью кнопки увеличь масштаб изображения так, чтобы тебе было удобно рисовать, и дорисуй собачке открытый рот.

Помни, лишние линии ты можешь удалить с помощью ластика (кнопка в панели инструментов). Подходящий размер ластика ты можешь выбрать из открывающегося меню под панелью инструментов. Подрисуй с помощью инструмента **Кисть** контур открытого рта. После чего выбери инструмент **Пипетка**, щелкни мышью в виде пипетки внутри контура персонажа (чтобы скопировать цвет), выбери инструмент **Заливка** и залей дорисованную нижнюю челюсть открытого рта. Можешь подрисовать язычок и изменить положение хвоста.

- Когда редактирование второго костюма закончишь, закрой графический редактор, нажав кнопку **ОК**. Все внесённые изменения автоматически будут сохранены.

В результате у тебя получится два костюма: один с открытым, другой с закрытым ртом.

- Переименуй первый костюм, назвав его **Max рот закрыт**, а второй костюм назови **Max рот открыт**.

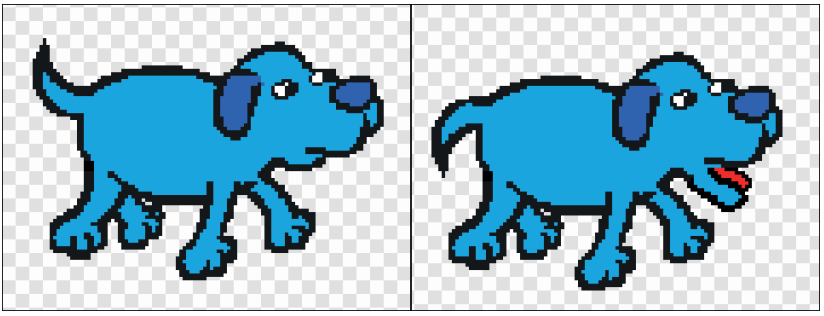


Рис. 4.8. *Два костюма для Макса*

Далее нужно сделать костюмы для Тины.

- Сделай копию Макса. Для этого в правой нижней панели под сценой щёлкни правой кнопкой мыши на ярлыке объекта **Max** и выбери команду **дублировать**.

Справа от ярлыка объекта **Max** появится его копия, но эта копия по умолчанию будет называться **Спрайт1**.

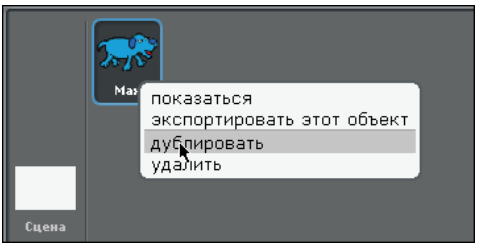


Рис. 4.9. *Копируем (дублируем) объект*

- Чтобы новый объект сделать активным, щёлкни на нём мышью. В верхней части средней панели появится имя нового объекта – **Спрайт1**.
- Если после этих действий откроется вкладка **скрипты**, выбери вкладку **костюмы**.
- Переименуй вновь созданный объект и назови его, например, **Тина**.

Для этого щёлкни мышью на поле ввода, расположенного вверху средней вкладки, введи нужное имя и нажми клавишу **Enter**. В правой нижней панели в нижней части ярлыка нового персонажа появится введённое имя.

- На вкладке **костюмы** для персонажа под именем Тина удали второй костюм.
- На вкладке **костюмы** введи в поле ввода имени первого костюма имя Тина и нажми клавишу **Enter**.
- Под полем ввода имени костюма нажми кнопку **Редактировать** (рис. 4.10). Откроется графический редактор, в окне которого ты увидишь изображение выбранного костюма.

Тина, в отличие от Макса, смотрит в другую сторону. Поэтому в первую очередь тебе нужно зеркально отразить рисунок костюма этого персонажа. Сделать это можно так:

- нажми в панели инструментов графического редактора кнопку **Выбор** и обведи изображение рамкой (рис. 4.11);
- нажми в панели инструментов графического редактора кнопку **Горизонтальное отражение** (рис. 4.11).

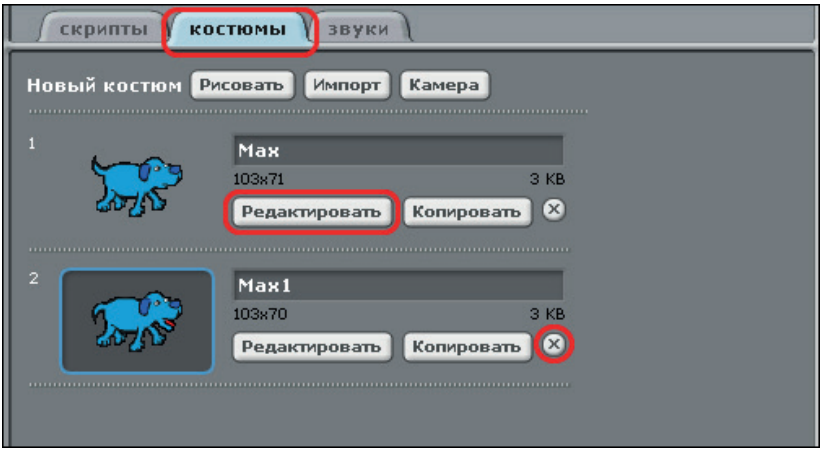


Рис. 4.10. *Второй костюм копии удаляем, первый – преображаем*

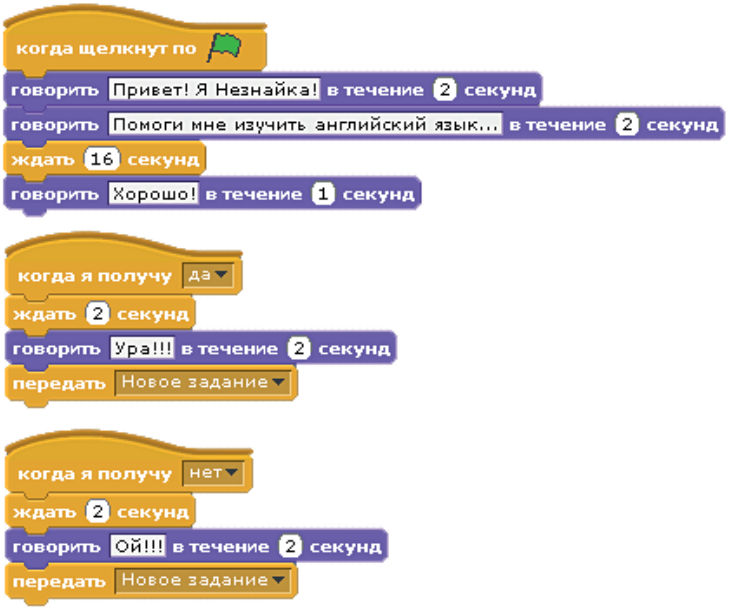


Рис. 4.11. *Зеркальное отражение изображения по горизонтали*

Далее нужно размер рисунка немного увеличить и перекрасить.

- Чтобы увеличить размер рисунка, нажми два раза кнопку **Расти** .
- Выбери инструмент **Заливка цветом** .

- Выбери в палитре цветов цвет для костюма и щёлкни мышью, принявшей вид , внутри контура костюма.
- Залей таким образом все детали костюма. Выбери другой цвет и залей выбранным цветом ухо и нос.
- Сохрани изменения, нажав кнопку **OK** в правом нижнем углу графического редактора.

Сделай второй костюм для персонажа Тина, в котором у собачки будет открыт рот. Для этого нажми расположенную под полем ввода имени персонажа вкладки **костюмы** кнопку **Копировать**. Ниже появится копия первого костюма. По умолчанию эта копия будет называться **Тина1**. Далее нажми кнопку **Редактировать** и нарисуй в графическом редакторе открытый рот, после чего сохрани внесённые изменения. После этого сделай для Тины ещё один костюм, в котором, например, приподними ей голову, как будто собачка смеётся. В результате у тебя получится три костюма. Назови эти костюмы, например, **Тина**, **Тина говорит** и **Тина смеётся**. Эти имена будут нужны при составлении скриптов.

Скрипты: синхронизация через ожидание (обслуживание)

Вообще-то, достаточно объяснить работу одного скрипта, например скрипта для персонажа по имени Max. Для остальных участников диалога скрипт работает по такому же принципу.

- Щёлкни мышью на ярлыке **Max**, который находится в правой нижней панели.
- Открой вкладку **скрипт** и собери скрипт для Макса, как показано на рис. 4.12 (верхний). Как работает этот скрипт, рассказано ниже, в разделе «Как это работает».
- Собери скрипт для Тины, как показано на рис. 4.12 (нижний).

Ты так же можешь нарисовать и фон. Для этого щёлкни мышью на ярлыке сцены в левой части нижней панели, нажми кнопку **Редактировать** и залей сцену, например, зелёным цветом. Можешь нарисовать цветочки.

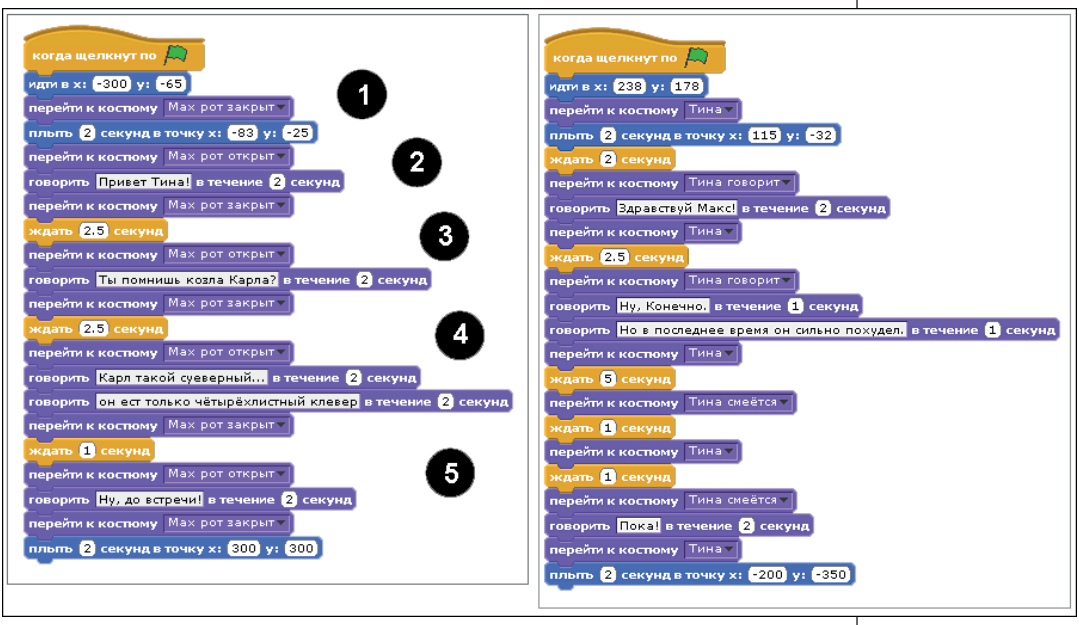


Рис. 4.12. *Скрипты для Макса (левый) и для Тины (правый)*

Как это работает

1. С помощью этой части скрипта Макс перемещается за пределы левой части экрана. Затем Макс в течение двух секунд перемещается в точку экрана с координатами X = –83, Y = –43. Изначально его рот закрыт.
2. Собака что-то произносит. Сначала она открывает рот, затем появляется облачко с текстом, который виден на экране 2 с. Потом рот у собачки закрывается снова.
3. Теперь 3 с говорит другой собеседник. Заметь, если речь собеседника будет длиться более 3 с, время в поле ввода командного блока **ждать** нужно увеличить и сделать это время на 1 с больше, чем время, которое занимает речь собеседника. Это называется синхронизацией через ожидание.
4. Длинный текст лучше разделить на два облачка. Это сделает текст не только читабельным, но и интересным. Ну а командный блок **перейти к костюму Max рот закрыт** отображает костюм с закрытым ртом.
5. Макс ждёт, пока Тина отсмеётся, говорит «До встречи» и уходит.

Проект 8. Интерактивная анимация – синхронизация через сообщения

В этом разделе мы будем с тобой развивать интерактивную историю с несколькими объектами и несколькими фонами. Это простая приключенческая игра со множеством сцен и одним финальным кадром. В каждой сцене зритель должен выбирать подсказки на картинке, а затем для продолжения игры совершать правильное действие.

Игра

В первой сцене мы увидим звёздное небо. Если нажать на нужную звёздочку, то она увеличится и в течение 2 с будет «думать». Затем сцена изменится.

Важно, что текст на картинке не полностью раскрывает информацию о том, что тебе нужно сделать с клавиатурой и мышью. Игрок должен сам догадаться, щёлкнув мышью на Полярную звезду.

Рис. 4.13. *Сцена первая: где Полярная звезда?*

Во второй сцене нужно сорвать кокосовый орех. Как это сделать? Очень просто. Если нажать на орех, он упадёт. После этого можно переходить к следующей сцене. Возможно, для тебя это чересчур легко. Ты, конечно, можешь всё настроить по-другому. Например, можешь нажать на ствол, пальма закачается и сбросит вниз орех. Но мы всё-таки остановимся на упрощённой версии.

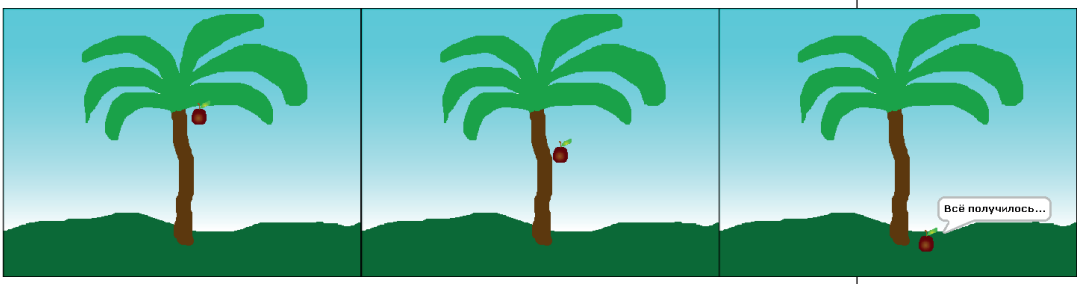


Рис. 4.14. *Сцена вторая: как сорвать кокос?*

Третья сцена более изысканная. Чтобы полить цветок, игроку нужно с помощью функции **Drag&Drop** переместить лейку. Как только лейка окажется над растением, она тут же наклонится, и польётся вода. При этом растение вырастает до размеров куста. Ну а после появляется финальный кадр.

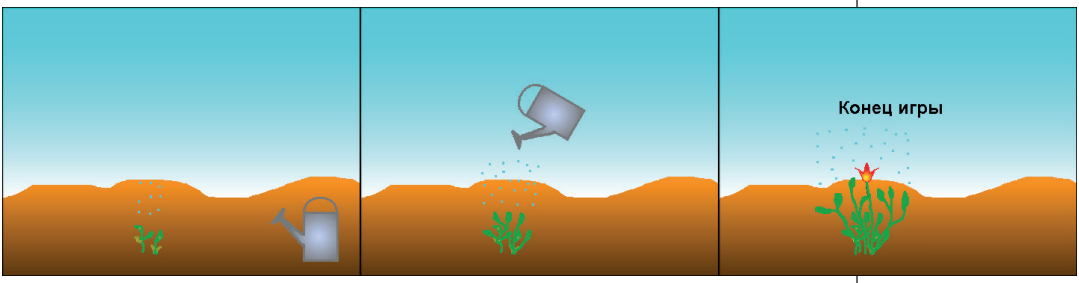


Рис. 4.15. *Сцены 3 и 4 (заключительные)*

Сцена

Будет лучше, если ты представишь себе всю анимацию в виде театральной постановки, состоящей из множества сцен. И каждой сцене требуется свой собственный фон. К тому же у нас есть четыре объекта, играющих как актёры на сцене. Это звёздочка, кокос, лейка и цветок. Рисунок 4.16 наглядно демонстрирует, как выглядит конец этой сказки.

К объектам мы ещё вернёмся, а пока займёмся сценой.

- Создай новый проект, удали кота и сохрани вновь созданный проект под именем, например, **Игра1**.
- Щёлкни мышью на значке сцены, который находится в левой части правой нижней панели.

  Вокруг этого значка появится голубая рамка, обозначающая, что данный элемент управления выбран. В поле ввода верхней части средней панели появится название **Сцена**, автоматически откроется вкладка **скрипты**.

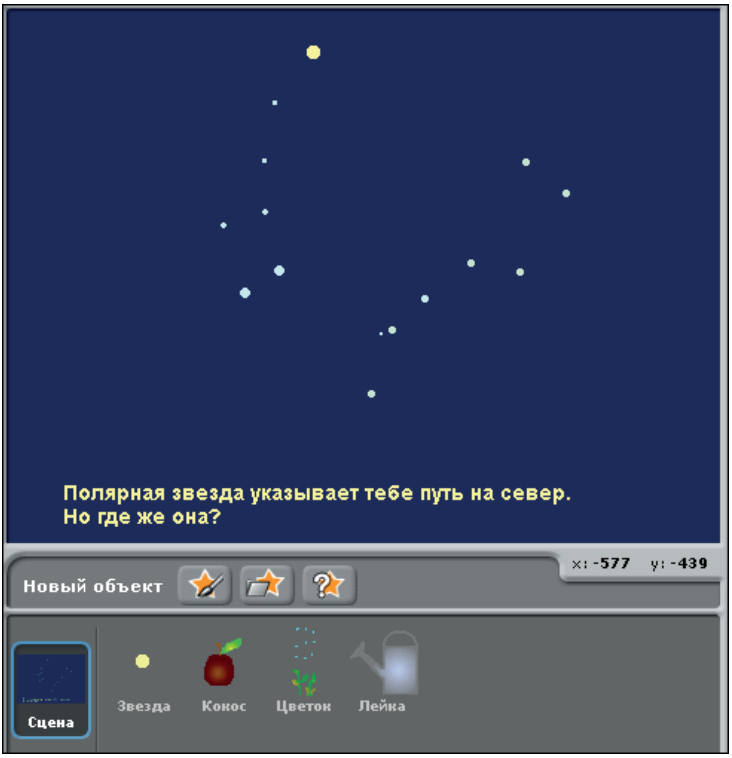


Рис. 4.16. *Первый фон сцены и символы четырёх персонажей*

- Щёлкни мышью на ярлыке **фоны**, чтобы открыть данную вкладку. Вначале на сцене ты увидишь пустой фон (белое поле).
- Нажми кнопку **Редактировать**. На экране появится графический редактор.

  На рис. 4.16 слева ты увидишь созвездие «Ковш Большой Медведицы» (его ещё называют просто Большой Медведицей) и другие звёзды. Слева и выше от Большой Медведицы находится созвездие под названием Малая Медведица. На рис. 4.16 справа показаны очертания этих созвездий.
- Нарисуй созвездия Большой и Малой Медведицы, как показано на рис. 4.16 слева, предварительно выбрав соответствующий цвет и размер для кисти.
- Сохрани первый фон под именем **Звёздное небо**.
- Запомни цвет и размер кисти, которой рисовались звёзды. В дальнейшем нам это пригодится.
- Сделай две копии нарисованного фона. Мы их сейчас отредактируем. На одном фоне мы нарисуем пальму,

а на другом – пустыню, и, соответственно, так и назовём: **Пальма** и **Пустыня**.

- Нажми два раза кнопку **Копировать**. Кнопка находится в средней панели под полем ввода имени первого фона **Звёздное небо**.
- Нажми кнопку **Редактировать** под полем ввода первой копии фона. Откроется окно графического редактора, в поле которого ты увидишь нарисованный ранее фон.
- Нажми кнопку **Очистить**.
- Нарисуй пальму, примерно такую, как на рис. 4.14. Но кокосовый орех рисовать не нужно.
- Нажми кнопку **Редактировать** под нижней копией фона звёздного неба, очисти рабочее поле (кнопка **Очистить**) и нарисуй пустыню как на рис. 4.15. Но без кустика и лейки. Можешь украсить рисунок тучей с дождиком, идущим вдалеке, и солнышком.
- Сделай ещё одну копию этого фона, выбери инструмент **Текст** T, напиши, например, **Конец игры** и сохрани этот фон под именем **Пустыня1** (рис. 4.17).

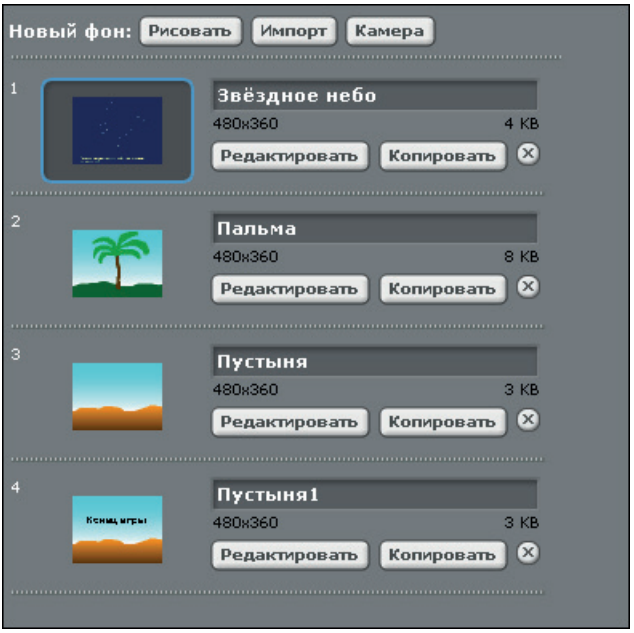


Рис. 4.17. *Фоны созданы*

Каждая сцена – это новый фон, или, если говорить языком театра, декорация. Здесь ни один элемент сцены не оснащён командой для смены фона, но некоторые элементы

сцены могут отправлять сообщения. На эти сообщения и реагируют скрипты.

- Открой вкладку **скрипты** и собери скрипт так, как показано на рис. 4.18.

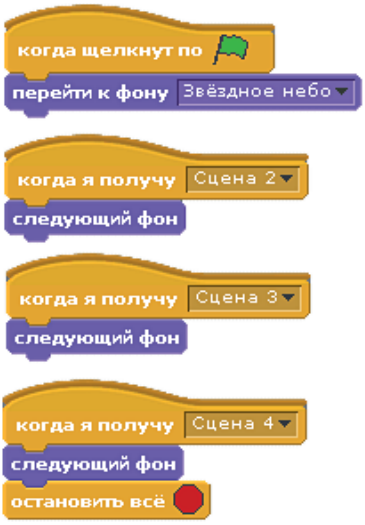


Рис. 4.18. *Скрипты сцены для изменения фона*

Первый скрипт отвечает за то, чтобы в начале игры была видна сцена под именем Звёздное небо. Следующие скрипты отображают на сцене следующие коды. Но тут есть одна проблема. Чтобы, когда это потребуется, отобразить на сцене следующий фон, командный блок **когда я получу...** должен получить соответствующее сообщение, на которое этот блок должен отреагировать и отобразить на сцене следующий фон. Но у нас есть три фона, и поэтому будет три сообщения. Командный блок **когда я получу...** должен отреагировать только на то сообщение, имя которого выбирается из открывающегося списка в правой части блока. Остальные сообщения он должен игнорировать. Поэтому все сообщения будут носить свои уникальные имена. Но, так как этих сообщений пока нет, имена сообщений также отсутствуют. Поэтому мы сначала внесём имена будущих сообщений в открывающиеся списки, а потом создадим и сами сообщения.

- Нажми на стрелку в правой части открывающегося поля ввода командного блока **когда я получу...** и выбери из открывшегося меню строку **новый...**. Появится поле ввода, в которое нужно ввести имя будущего сообщения.
- Введи имя первого сообщения **Сцена 2**.

- Также введи в следующих командных блоках имена **Сцена 3** и **Сцена 4**.

Рис. 4.19. *Чтобы ввести имя сообщения, выбери строку* ***Новый***

Полярная звезда

Итак, сцена готова, и на ней должны появиться актёры. Начнём с первой сцены и Полярной звезды.

- В панели инструментов правой нижней панели нажми кнопку . Нарисуй кистью круг такого же цвета и примерно того же размера, что и звёзды на фоне, после чего сохрани объект, нажав кнопку **OK**.
- Назови свой новый объект Звезда.

Поле, куда ты можешь ввести имя своего нового объекта, находится вверху средней панели, над вкладкой **костюмы**.

Передвинь Полярную звезду в нужное место на сцене. Найти правильную позицию для неё ты можешь следующим образом: обрати внимание на две верхние звезды в верхней части ковша Большой Медведицы (их называют *Мерак* и *Дубхе*). Мысленно нарисуй прямую линию через две эти звезды влево и вверх. На этой самой линии и находится Полярная звезда. Ну, скажем, расстояние между звёздами Мерак и Дубхе – это у нас *x*. Значит, Полярная звезда расположена на расстоянии приблизительно 5**x* на этой линии.

Рис. 4.20. *Скрипты для звёзд*

Как это работает

1. При запуске программы размер звезды будет таким, как ты её нарисовал. Звезда ждёт, пока ты щёлкнешь на ней мышью.
2. Когда ты Полярную звезду найдёшь и щёлкнешь на ней мышью, её размер увеличится в три раза, и на экране появится сообщение «Да, я Полярная звезда...».
3. Далее с помощью блока команд **передать** будет отправлено сообщение **Сцена 2**, адресованное всем, и на экране появится следующий фон. В нашем случае это будет пальма. В то же время благодаря командному блоку **спрятаться** звезда исчезнет со сцены.

Нажми кнопку  , которую ты найдешь справа вверху над сценой, и посмотри, как работает скрипт. Если ты все сделал правильно, на экране отобразится фон с пальмой.

Сообщения в скриптах сцены ты настроил раньше. Если ты хочешь отправить сообщение другой сцене, выбери название новой сцены из открывающегося списка **Передать**. Но в нашем случае сообщение отправляется для второго фона. Кстати, это послание предназначено не только для сцены, но и для актёра следующей сцены, которому также понадобится ключевое слово.

Кокосовый орех

Во второй сцене у нас запланирован выход на сцену кокосового ореха. Согласно скрипту он не должен показываться на первой сцене. Теперь настаёт очередь его выхода. В этой сцене главная роль будет принадлежать именно ему.

Если ты предыдущий скрипт не проверял и фон с пальмой на сцене отсутствует, отобразить вторую сцену можно следующим способом:

- в левой части правой нижней панели щёлкни мышью на ярлыке **Сцена**;
- открой вкладку **фоны**;
- щёлкни мышью на ярлыке фона **Сцены 2** (с пальмой).

Теперь нарисуем сам кокосовый орех:

- нажми кнопку  и нарисуй в графическом редакторе кокосовый орех;
- закрой графический редактор, нажав кнопку **ОК**;
- переименуй его и назови **Кокос**;
- перемести кокос в нужное место. Орех должен висеть на пальме;

- открой вкладку **скрипты**;
- собери скрипт, как показано на рис. 4.21.

Рис. 4.21. *Скрипты для кокоса*

Как это работает

1. После запуска программы кокос не виден. Он появляется на сцене только тогда, когда услышит ключевое слово. Командный блок **Идти в x: и y:** перемещает кокос в исходную точку. Обрати внимание: координаты из командного блока на картинке с твоими реальными координатами, скорее всего, не совпадут. Поэтому, когда ты установишь командный блок **Идти в x: и y:**, в полях ввода командного блока ты увидишь реальные координаты.
2. Когда кокос услышит, что **Сцена 2** началась, он появляется на сцене.
3. После того как ты щёлкнешь на кокосе мышью, он упадёт вниз.
4. Сцена подошла к концу. Кокос отправляет всем сообщение, что начинается **Сцена 3**, а сам скрывается. Кроме того, на экране появится фон с пустыней.

Цветок

В третьей сцене принимают участие два объекта – цветок и лейка. Цветок начинает расти, как только лейка касается его.

- Нажми кнопку и нарисуй в графическом редакторе маленькое растение.

Это будет первый костюм. У этого растения могут быть даже пожелтевшие ростки, потому что ему нужна вода.

- Кистью маленького размера нарисуй в верхней части растения парочку крошечных точек неприметного цвета (рис. 4.22 левый).

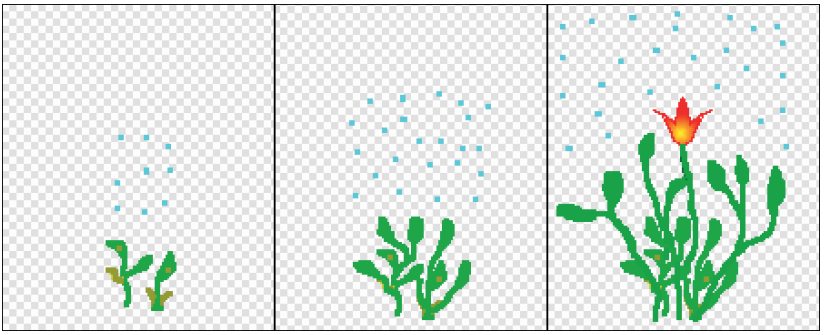


Рис. 4.22. *Костюмы показывают цветок в различных стадиях роста*

Ты уже догадался, зачем нужны цветку эти точки? При поливе цветочка лейку нужно держать именно над этими точками. Через них растение будет незаметно вытягиваться вверх, для того чтобы Scratch мог зафиксировать касание лейки к цветку.

- Сохрани первый костюм и назови этот персонаж, введя имя **Цветок** в верхнее поле ввода средней панели.
- Нажми под полем ввода **Костюм1** кнопку **Копировать**. Ниже появится копия первого костюма.
- Под полем ввода **Костюм2** нажми кнопку **Редактировать** и дорисуй веточки, как будто куст растёт.
- Сохрани внесённые изменения, нажми кнопку **Копировать** и отредактируй **Костюм3**, увеличив длину веточек и, возможно, дорисовав цветочек.
- Сохрани третий костюм.
- Если хочешь, переименуй костюмы, назвав их, например, **Цветок1**, **Цветок2** и **Цветок3** или **Маленькое растение**, **Растение подросло** и **Большое растение**. Эти имена нам будут нужны в дальнейшем.
- Установи первый костюм на сцене в нужное положение.
- Перейди на вкладку **скрипты** и собери скрипт для цветка, как показано на рис. 4.23.

Рис. 4.23. *Скрипты для растения*

Как это работает

1. После запуска скрипта цветок в костюме Цветок1 (переименован из **Костюм1**) появляется на сцене, но командный блок **спрятаться** скрывает его до появления **Сцены 3**, фон которой представляет собой пустыню.
2. После получения сообщения **Сцена 3** цветок (в первом костюме) становится видимым (за это отвечает командный блок **показаться**). Рост цветка начинается после получения от кувшина сообщения **Расти**. Так как этого сообщения ещё нет, щёлкни мышью на маленьком чёрном треугольнике, расположенном в правой части открывающегося списка командного блока **когда я получу**, выбери в появившемся контекстном меню строку **новый…** и введи в появившемся поле ввода имя сообщения **Расти**.
3. Когда цветок вырастет, мы переходим к следующей сцене. Когда очередь дойдёт до последнего костюма и цветок вырастет, можно переходить к следующей сцене. Проследи за тем, чтобы твоё растение не исчезло (никаких команд *прячься*).
4. Если цветок ещё не вырос, цикл повторяется, цветок надевает следующий костюм, и цикл повторяется вновь, пока цветок не наденет следующий костюм.

Лейка

У лейки всего два костюма. Первый, когда лейка стоит в вертикальном положении, а второй костюм – лейка наклонена и из неё течёт вода.

- В панели инструментов правой нижней панели нажми кнопку . Откроется графический редактор.
- Используя инструменты **Прямоугольник** , **Линия** и **Эллипс** , нарисуй лейку как на рис. 4.16.
- Сохрани новый персонаж. Изображение лейки появится в правой нижней панели рядом с изображениями звезды, кокоса и цветка.
- В средней панели создай копию этого костюма. Для этого нажми кнопку **Копировать**, расположенную под полем ввода **Костюм1**.
- Нажми кнопку **Редактировать**. Появится окно графического редактора с изображением лейки.
- В панели инструментов графического редактора нажми кнопку **Выбор** и выдели всю лейку, обведя её рамкой.
- Нажимая кнопку **Поворот против часовой стрелки** , поверни лейку так, как будто ты поливаешь.

Далее мы в скриптах добавим функцию Drag&Drop – перетаскивание, с помощью которой мы будем переносить лейку мышкой.

- Перейди на вкладку **скрипт** и собери скрипт для лейки, как показано на рис. 4.24.

Как это работает

1. Пока левая кнопка нажата, лейка перемещается за указателем мыши. Если левую кнопку мыши отпустить, лейка останется в том положении, в котором ты её оставил. Так работает командный блок всегда, когда мышка нажата. Это и есть Drag&Drop!
2. Если коснуться цветка, лейка наклонится и начнёт лить воду. К растению относятся также и маленькие спрятанные точки над зелёными усиками.
3. После небольшого ожидания сообщение **Расти** будет отправлено. Цветок прочитает его, после чего сменит свой костюм и немного вырастет.
4. В случае когда лейка *не* касается растения, она вернётся в вертикальное положение.

Важно

Функция **Drag&Drop** работает только в полноэкранном режиме. Поэтому перед тестированием своего проекта нажми в правом верхнем углу кнопку .

Рис. 4.24. *Скрипты для лейки*

Проект 9. Викторина

В этом разделе мы будем программировать викторину, и речь в ней пойдёт об изобретении. На рисунке изображён Томас Эдисон (Thomas Edison), известный изобретатель, которому мы благодарны за создание электрической лампочки. В проекте он называет предположительную дату какого-либо творения, например «Мне кажется, что замок-молния был изобретён до 1909 года». Игроку необходимо нажать на зелёный или красный палец, чтобы согласиться с этим утверждением или оспорить его.

Далее следует ответ Эдисона: «Ты совершенно прав!» или «Нет, здесь ты ошибся».

Если игрок прав, то он получает один балл. Если проиграл – балл минусуется.

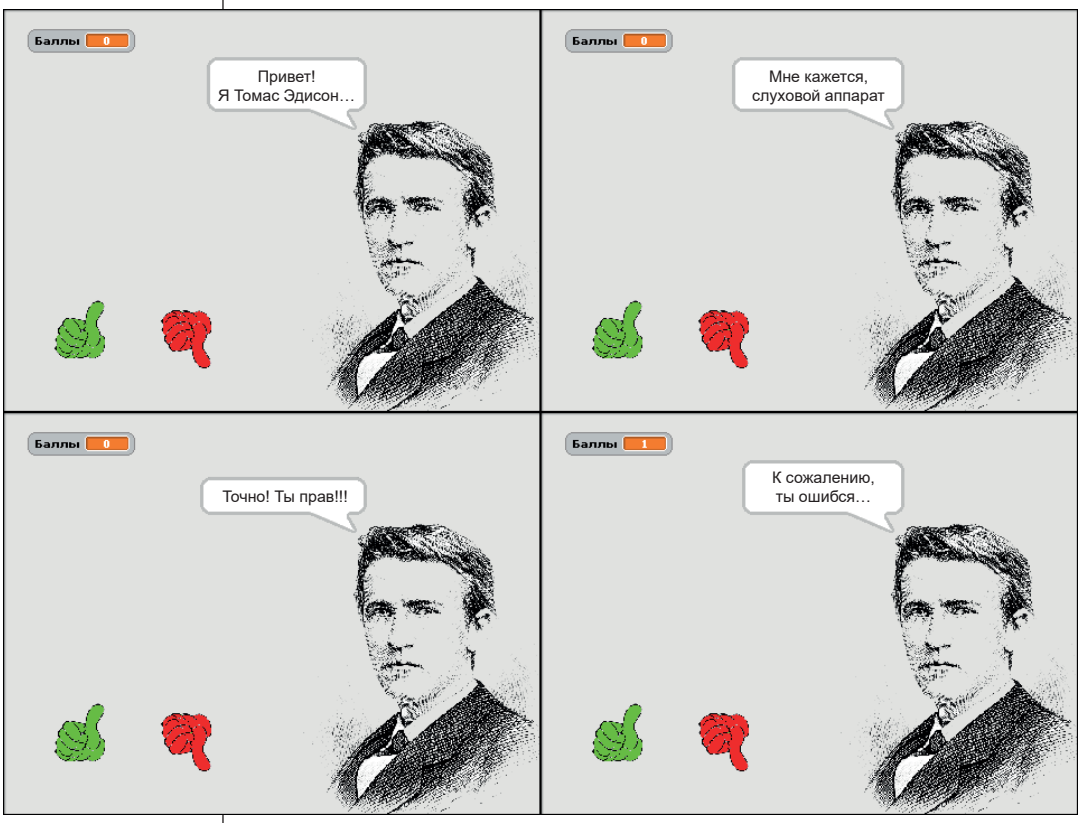


Рис. 4.25. *Вопрос викторины состоит из нескольких частей*

Списки

В этом проекте мы используем списки. Ты уже знаком с ними в обычной жизни: список покупок, список гостей для твоей следующей вечеринки или же список дел, которые ты хочешь выполнить в период летних каникул.

Списки в информатике – это *изменяемая последовательность элементов*. Изменяемая означает, что ты можешь к своему списку что-то добавить (т. е. сделать его длиннее) или же удалить какие-то элементы, заменив их другими.

Каждая строка (элемент) списка имеет свой *индекс*. Индекс – это целое число, указывающее на позицию данной строки в списке. Первый элемент (строка) содержит индекс 1, второй – индекс 2 и т. д.

При подготовке этого проекта тебе необходимо для начала написать список изобретений. Ниже ты увидишь пример такого списка. Правее каждого предмета в скобках указан год изобретения.

Что это за предметы, и когда они были изобретены?
Огнетушитель (1863)
Стиральная машина (1864)
Электрический утюг (1882)
Грампластинка (1887)
Замок-молния (1893)
Слуховой аппарат (1901)
Электрический тостер (1909)
Электрический холодильник (1913)
Светофор (1914)
Микроволновая печь (1947)
Застёжка-липучка (1948)
Кредитная карта (1950)

В проекте *Scratch* вышеприведённый список следует разделить на две части. Первый список мы назовём **Устройства.** Этот список будет состоять из названий изобретённых устройств. Второму списку мы присвоим название **Год**. Этот список содержит даты изобретений устройств.

Для создания списков выполни следующие действия:

- создай новый проект и сохрани его под именем **Викторина**;
- удали со сцены кота;
- в левой верхней панели нажми кнопку **Переменные**. В левой нижней панели появится поле блоков команд **переменные**. Но пока здесь нет ни одного командного блока, а только две кнопки: **Создать переменную** и **Создать список**;
- нажми кнопку **Создать список**. Появится диалоговое окно с полем ввода для имени создаваемого списка (рис. 4.26);
- введи в это поле ввода слово **Устройства**. Это будет имя списка;
- закрой диалоговое окно с полем ввода имени списка, нажав кнопку **ОК**.

После того как список **Устройства** будет создан, ты увидишь два изменения:

- на сцене появляется серое поле. Это заготовка для списка. Ты можешь изменять размер самого поля, вносить новые элементы (строки) и редактировать ранее внесённые элементы. Пока этот список пуст (**длина: 0**);
- в коллекции команд **переменные** ниже имени созданного списка появится блок команд, относящийся

к этому списку. Перед именем списка ты увидишь установленный флажок. Если снять этот флажок, список **Устройства**, который ты сейчас видишь на сцене, скроется. Чтобы вновь отобразить данный список, установи этот флажок.

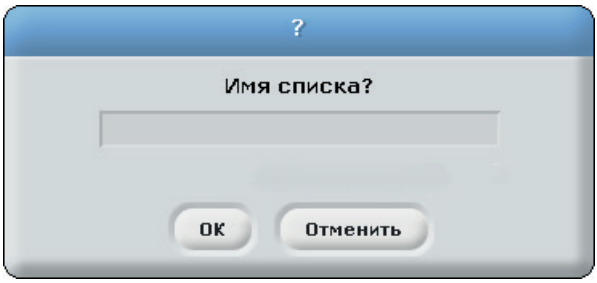


Рис. 4.26. *Диалоговое окно для ввода имени списка*

Пока список пуст, а размер его поля маленький. Сначала подготовь поле списка. Для этого выполни следующие действия:

- установи указатель мыши на названии заготовки для списка, нажми левую кнопку мыши и, не отпуская левую кнопку мыши, перемести заготовку в середину сцены;
- установи указатель мыши на правом нижнем углу заготовки, нажми левую кнопку мыши и, не отпуская её, перемести мышь вправо и вниз. Вслед за указателем мыши начнет увеличиваться размер поля списка (если ты будешь перемещать таким образом правый нижний угол в направлении левого верхнего угла, размер поля списка будет уменьшаться);
- щёлкни мышью на маленьком плюсике в левом нижнем углу заготовки, чтобы создать новый элемент (строку). В верхней части заготовки появится поле ввода с номером 1 (слева от поля ввода). Номер строки – это и есть **Индекс** данной строки;
- внеси в строку название первого устройства **огнетушитель** (рис. 4.27).

Первый элемент списка имеет индекс **1**. Это очень большая редкость, так как в большинстве языков программирования нумерация начинается с цифры **0**.

- Заполни таким образом весь список. Чем длиннее список, тем интереснее викторина.
- Таким же образом создай список **Год** с датами изобретений. Следи, чтобы дата в строке этого списка совпадала со строкой названия изобретённого в этот год

устройства из списка **Устройства** и чтобы номера строк совпадали (рис. 4.28). Имя и содержание списков показано в табл. 4.1.

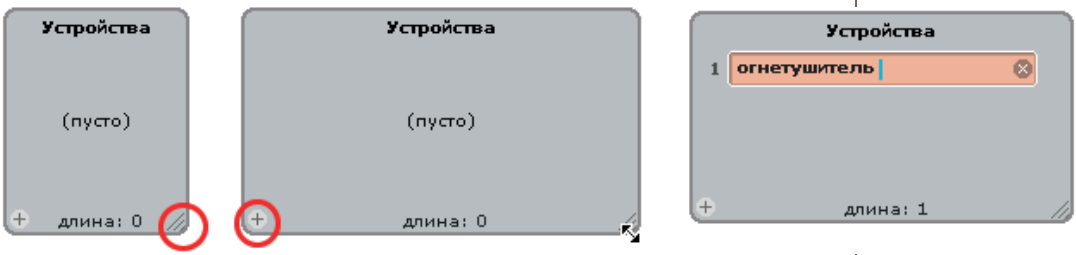


Рис. 4.27. *Измени размер списка и введи название устройства*

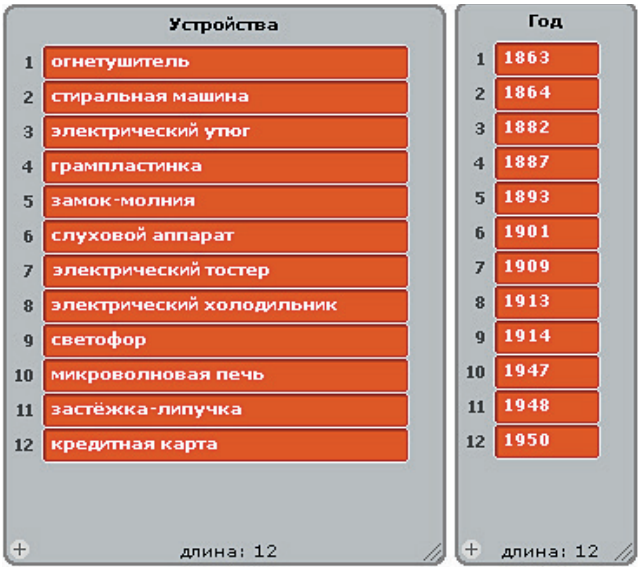


Рис. 4.28. *Списки созданы*

Обрати внимание! Это важно! При сохранении проекта все переменные также будут сохранены, и никакие данные утеряны не будут. Но тебе понадобятся и другие переменные. В приведённой ниже таблице ты увидишь, какие списки тебе нужно ещё создать.

Таблица 4.1. Имя и назначение списков

Имя списка	Назначение
Устройства	Список названий устройств
Год	Список, отображающий, в каком году было изобретено устройство (отображается цифрами)

Таблица 4.2. Имена и назначение переменных

Имя переменной	Назначение переменной
Баллы	Отображает количество заработанных игроком баллов
Дата	Год, наиболее близкий к году изобретения устройства
Номер	Случайное число, с помощью которого выбирается название устройства и год его изобретения

- Создай переменную **Баллы**. Для этого в левой панели с переменными нажми кнопку **Создать переменную**, в появившемся поле ввода введи имя переменной **Баллы** и нажми кнопку **OK**.
- Создай переменную **Номер** и сделай это поле ввода невидимым.
- Создай переменную **Дата** и сделай её невидимой.
- Сделай списки невидимыми. Для этого в левой панели сбрось флажки, расположенные слева от названия списка. Флажок левее переменной **Баллы** не сбрасывай.

Итак, списки созданы. Приступаем к следующему этапу – создадим ведущего викторины.

Ведущий викторины

Ведущий нашей викторины – Томас Эдисон. Поскольку фото уже достаточно старое, то является общедоступным, его можно свободно использовать, не нарушая авторских прав. Ты, конечно же, можешь найти фото и других личностей, которые будут задавать вопросы викторины. Возможно, даже захочешь использовать своё собственное фото. Если решишь опубликовать свой Scratch-проект, то обязательно убедись, что авторские права не будут нарушены или, если человек жив (например, это твой знакомый), он не будет возражать, что его изображение будет опубликовано. Этого требует «авторское право на изображение». Томас Эдисон умер уже очень давно и поэтому не может препятствовать использованию своего фото в этом проекте.

- Запусти поисковую машину, введи в строку поиска имя **Томас Эдисон**, выбери режим поиска **Картинки** и загрузи найденную фотографию в папку с проектом.
- Загрузи файл в проект. Для этого нажми кнопку **Выбрать новый объект из файла**  , открой папку, в которой ты сохранил портрет, щёлкни на нём мышью и нажми кнопку **OK**. Портрет появится на сцене и в правой нижней панели. Это и будет наш персонаж.

- Назови его **Томас Эдисон**. Для этого введи имя и фамилию в поле ввода в средней панели над ярлыками вкладок **скрипты**, **костюмы** и **звуки** (по умолчанию в этом поле ввода ты увидишь имя **Спрайт1**).
- Открой вкладку **костюмы** и нажми кнопку **Редактировать**. Портрет учёного откроется в графическом редакторе.
- Выбери инструмент **Заливка**  , выбери в палитре цветов правый нижний образец цвета (без заливки) и щелкни по фону рисунка.
- Если потребуется, уменьши размер изображения и закрой редактор.
- Установи портрет в правом нижнем углу сцены.

Ведущий викторины является умным персонажем. Он задаёт вопросы, реагирует на верные и неправильные ответы и определяет, сколько баллов ты заработал. Но информацию о том, правильный ответ или нет, ведущий получает из сообщения, всплывающего после нажатия на красный или зелёный палец.

- Щёлкни мышью на миниатюре **Сцена** (в левой части правой нижней панели). В средней панели откроется вкладка **фоны**.
- Нажми кнопку **Редактировать** и залей фон понравившимся цветом. Например, светло-серым. После чего закрой графический редактор.

Итак, сцена готова. Теперь необходимо собрать скрипт.

- В правой нижней панели щёлкни мышью на ярлыке персонажа, в средней панели щёлкни мышью на ярлыке **скрипты**.
- Собери скрипт, как показано на рис. 4.29.

Как это работает

1. Этот скрипт создаёт новое задание. Прежде всего мы получаем переменную. Это случайное число от 1 и до последнего номера элемента в списке **Устройства**. Так как в списке **Устройства** 12 элементов, случайное число выбирается из диапазона от 1 до 12. Если же ты увеличишь список до 14 элементов, автоматически диапазон изменится от 1 до 14. Это случайное число определяет, к какому устройству относится вопрос викторины. Например, будет выбрана цифра **3**. Значит, предмет, о котором идёт речь, – это **электрический утюг**.

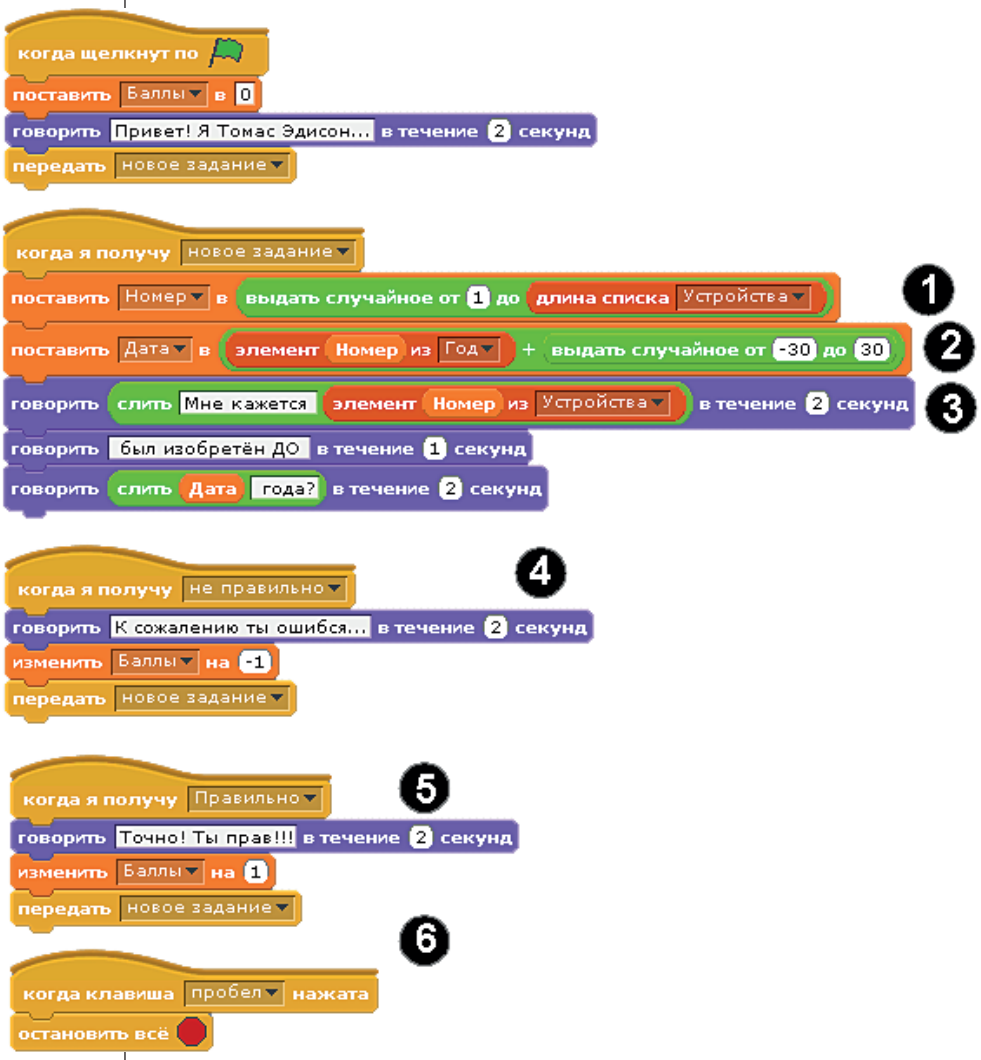


Рис. 4.29. *Скрипты для ведущего викторины*

2. Дата также выбирается случайным образом. Эта дата используется в тексте вопроса. Начальная точка – это год, когда было изобретено устройство. Год выбирается из списка **Год**. Номер строки выбирается с помощью переменной **Номер**. Затем мы получаем случайное число между –30 и +30. Это число прибавляется к году. В результате мы получаем дату, которая немного больше или немного меньше правильного ответа.
3. В этом скрипте формируется вопрос викторины. Вопрос начинается словами **Мне кажется**. Далее печатается название устройства, которое выбирается из

списка **Устройства**. Номер строки определяется той же переменной **Номер**, с помощью которой в прошлом командном блоке из списка **Год** выбиралась случайная дата. То есть если переменная **Номер** выдала значение **3**, в списке **Год** будет выбрана третья строка: **1882**. В списке **Устройства** тоже будет выбрана третья строка **Электрический утюг**. В результате мы получим начало фразы: «**Мне кажется, электрический утюг**». Далее мы переходим к следующему командному блоку. На экране появится продолжение фразы: «**был изобретён ДО**». После этого управление берёт на себя следующий командный блок, который формирует конец фразы: «***Дата* года?**», в которой дата – это переменная **Дата**, значение которой определяется командным блоком 2 (рис. 4.29).

4. Сообщение **Неверно** отправляется зелёным или красным пальцем, если ответ неправильный. Согласно этому сообщению баллы будут сниматься. После того как количество баллов уменьшится на 1, в работу вступает командный блок **передать**, который для всех отправляет сообщение **Новое задание**.
5. Если вы выбрали палец, соответствующий правильному ответу, всем будет отправлено сообщение **Правильно**. В этом случае количество баллов увеличится на 1, и с помощью командного блока **передать** будет всем послано сообщение **Новое задание**.
6. Этот скрипт при нажатии на клавишу **Пробел** полностью останавливает выполнение программы. Это сделано, чтобы ты смог закончить игру.

Создаём пальчики

Найди в интернете, скачай и сохрани в папке проекта картинку большого пальца, поднятого вверх. Щёлкни мышью на кнопке  , выбери изображение поднятого пальца и загрузи это изображение в проект. Это будет наш следующий персонаж. Открой вкладку **костюмы** и отредактируй этот костюм. Удали фон (для этого выбери инструмент  , выбери в палитре правый нижний образец цвета (прозрачный) и удали фон). Далее выбери зелёный цвет и залей этот костюм выбранным цветом. Сохрани изменения и назови этот костюм именем **Да**, потому что мы будем щёлкать по нему мышью, когда будем думать, что ответ правильный.

После чего в правой нижней панели щёлкни правой кнопкой мыши на ярлыке этого персонажа, выбери из контекстного меню команду **дублировать**. Будет создана копия этого пальца. Перейди на вкладку **костюмы**, нажми кнопку **Редактировать** и залей изображение красным цветом. Далее выбери инструмент **Выбор** и выдели изображение рамкой. После нажми кнопку **Вертикальное отражение**. Палец перевернётся и будет указывать вниз. Сохрани костюм под именем **Нет**. Имена **Да** и **Нет** вводятся в верхнем поле ввода средней панели.

- Собери для зелёного пальца скрипт, как показано на рис. 4.30.

Для этого в правой нижней панели щёлкни мышью на персонаже **Да** (зелёный палец), в средней панели выбери вкладку **скрипты** и собери скрипт.

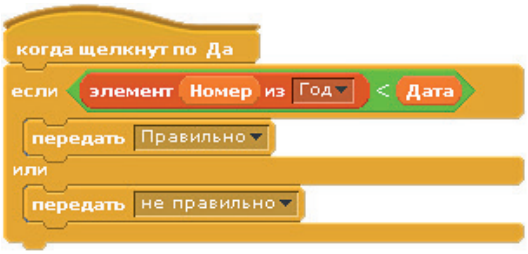


Рис. 4.30. *Скрипт для «зелёного пальца»*

Скрипт выполняется после того, как ты щёлкнешь мышью на руке зелёного цвета с поднятым вверх пальцем. Сначала объясним блок команд **элемент** со встроенным элементом **Номер**, который расположен левее символа **<**. Номер – это переменная, случайное число, используемое ведущим в задании. Год – это дата из списка **Год**. Это значение выбирается случайно, выбор определяется переменной **Номер**.

Если текущее значение **Год** меньше случайно выбранной даты, то будет отправлено сообщение **Правильно**. В остальных случаях отправляется сообщение **Неправильно**. Сообщение отправляется ведущему, и тогда он соответствующим образом реагирует.

В правой нижней панели щёлкни мышью на элементе **Нет**, открой вкладку **скрипт** и собери скрипт как на рис. 4.31.

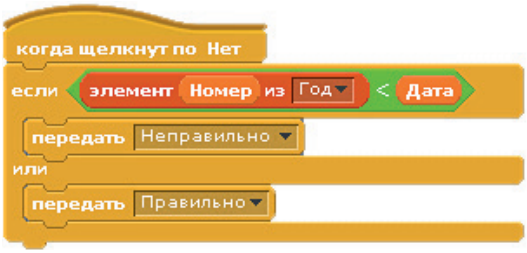


Рис. 4.31. *Скрипт для «красного пальца»*

Тестируем проект

В проекте с таким количеством списков и переменных легко допустить ошибку. Лучше всего протестировать программу следующим образом. Сделай все переменные и списки видимыми, установив флажки перед их названиями в коллекции команд **переменные** (рис. 4.32). Запусти программу и проконтролируй, все ли переменные имеют те значения, которые у них должны быть.

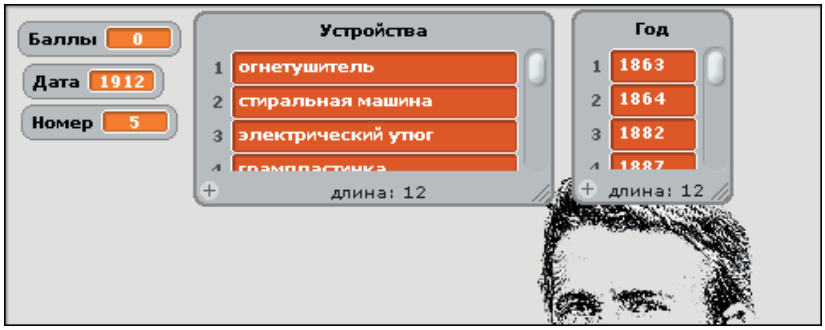


Рис. 4.32. *Необходимо все переменные и значения списков сделать видимыми*

Если всё идёт как надо, то эти флажки сбрось. Оставь только поле ввода **Баллы**.

Вопросы

1. В первом Scratch-проекте общались две фигуры. Как сделать так, чтобы они не перебивали друг друга, общаясь одновременно? Назови две техники.
2. Как можно удалить сообщение, которое было когда-то создано (например, с помощью блока **отправить...**)?

3. Какие значения имеют следующие выражения:

   а) 

;

   б) 

?

4. Добавь отсутствующие слова: … – это номер элемента в списке. У … списка нет ни одного элемента.

Задания

Задание 1. Автогонки с несколькими сценами

Во второй главе мы в программе Scratch создали программу автосимулятора. Однако в этом проекте есть один недостаток: размер гоночной трассы ограничен размером сцены. Устранить этот недостаток можно следующим образом: создать трассу размером в две сцены. Когда автомобиль касается правого края первой картинки, появляется новая картинка со второй частью трассы. Болид «перепрыгивает» на левую сторону сцены и едет дальше. То же самое происходит и в обратном порядке. Если автомобиль находится на правой части трассы и едет к левому краю, то картинка снова меняется.

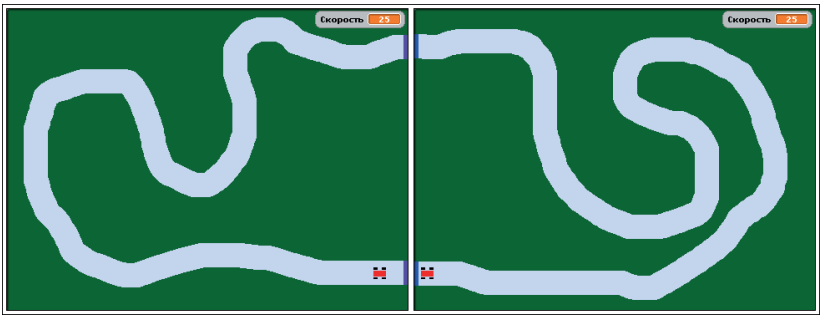


Рис. 4.33. *Размер гоночной трассы – два размера сцены*

Рекомендации

Нарисуй две фоновые картинки как на рис. 4.33. Обрати внимание вот на что. На правом краю первой картинки на гоночной трассе находится тонкая фиолетовая линия (цвет можешь выбрать любой). По этой линии объект-автомобиль может определить, что достиг правого края изображения. На правой грани второй картинки имеется линия голубого (или любого другого) цвета.

Сцена содержит два скрипта, и автомобиль тоже содержит два дополнительных скрипта (все остальные скрипты остаются неизменными).

Попробуй самостоятельно собрать дополнительные скрипты для перехода от первого фона ко второму из блоков команд, показанных на рис. 4.34. Ответ, как правильно собрать скрипты, ты найдёшь в конце главы.

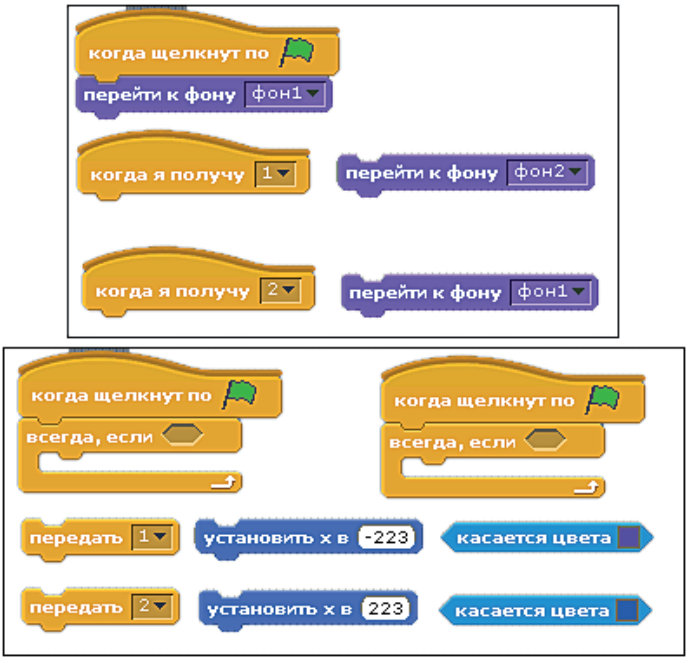


Рис. 4.34. *Блоки дополнительных скриптов для фона (вверху) и для автомобиля (внизу)*

И ещё, это очень важно. Чтобы скрипт работал корректно, тебе в начале скрипта необходимо будет внести изменения, как показано на рис. 4.35. Слева красной рамкой обведён командный блок **повторять до** (слева), который следует заменить на командный блок **всегда, если** (справа). Во встроенном командном блоке **касается цвета** необходимо выбрать цвет дороги (щёлкнуть мышью на образце цвета в командном блоке **касается цвета** (на рис. 4.35 справа обведён красным кружком) и щёлкнуть мышью в виде пипетки на нарисованной дороге). Почему мы так делаем? Командный блок **повторять до** повторяет цикл, пока машинка не коснётся травы (вылетит с дороги). Смысл работы командного блока **всегда, если** в том, что цикл повторяется, пока машинка будет на дороге (серый цвет). Через

секунду после того, как машинка окажется на траве, игра останавливается. Если этот командный блок не изменить, при смене фона машинка будет останавливаться, и с места её не сдвинуть.

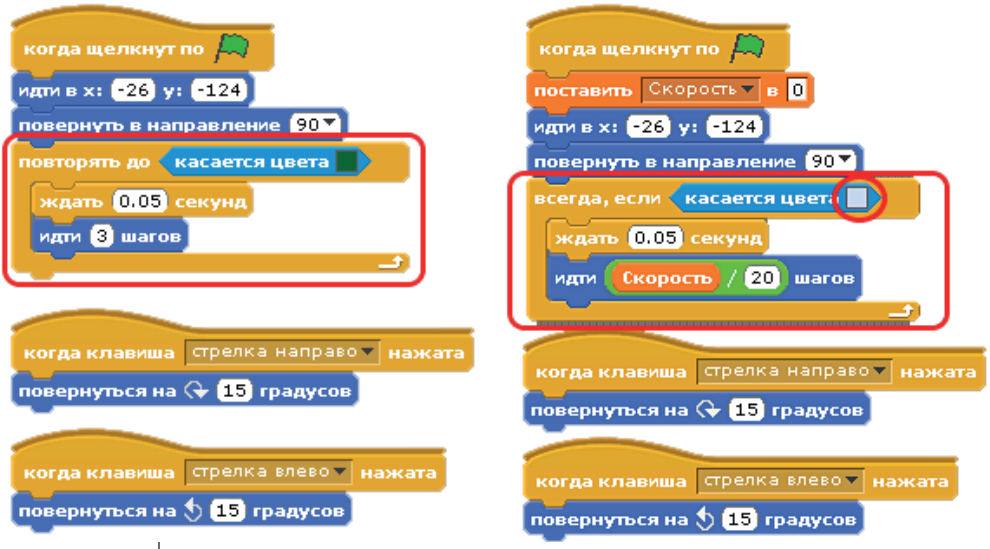


Рис. 4.35. *Изменение в скрипте для автомобиля*

Задание 2. Незнайка учит английский язык

Создай викторину для изучения английского языка. В этой игре участвуют пять персонажей: персонаж **Фото**, у которого будет 10 костюмов с разными фотографиями животных и предметов, персонаж **Незнайка**, персонаж **Знайка**, кнопка **Да** и кнопка **Нет**. Правила игры следующие. На экране появляется изображение. После того как ты щёлкнешь на изображении мышью, Знайка назовёт предмет. Тебе нужно нажать кнопку **Да**, если, по твоему мнению, Знайка прав, или кнопку **Нет**, если, по-твоему, Знайка не прав. Если кнопка **Да** была нажата правильно, Знайка тебя похвалит, если неправильно – поругает. Таково действие и кнопки **Нет**. Если, например, на экране ты видишь изображение автомобиля, Знайка говорит, что это, например, чашка, и ты нажимаешь кнопку **Нет**, ты даёшь правильный ответ. Знайка тебя хвалит. После этого на экране появляется следующая картинка, и цикл повторяется.

После нажатия кнопки происходит вступительный диалог между Незнайкой и Знайкой.

Сначала говорит Незнайка: «*Привет! Я Незнайка! Помоги мне изучить английский язык*».

После этого говорит Знайка: «*Привет! Я Знайка! Я помогу тебе!!! Щёлкни мышью на картинке. Я скажу название предмета или животного. Если я прав, нажми кнопку* ***ДА****. Если я ошибся, нажми кнопку* ***НЕТ***».

Незнайка отвечает: «*Хорошо!*»

Дальше, после того как ты щёлкнешь мышью на картинке, начнётся игра.

На рис. 4.36 показаны 8 кадров из этой игры. Нажатые кнопки обведены красными рамками. Далее ты увидишь советы по созданию игры.

Шаг первый: создаём костюмы для персонажа Фото

Создай новый проект под именем, например, **Незнайка**.

Скачай из интернета (нарисуй, сфотографируй и сохрани в папке *Scratch Projects*) изображения чашки, велосипеда, дерева, паровоза, ключа от замка, самолёта, автомобиля, лошади, кошки и удочки.

В панели инструментов нажми кнопку **Выбрать новый объект**, нажми в появившемся на экране окне графического редактора кнопку **Импорт**, открой папку **Scratch Project** (или папку, в которой были сохранены изображения), щёлкни мышью на нужном изображении и нажми кнопку **ОК**. Выбранное изображение будет загружено в графический редактор. Отредактируй изображение (убери фон; для этого выбери инструмент, выбери в палитре цветов правый нижний образец цвета (прозрачный) и щёлкни на фоне), если нужно, уменьши размер изображения (кнопка) и закрой редактор. Первый костюм персонажа создан. Щёлкни мышью на поле ввода в верхней части средней панели (над ярлыками **скрипты**, **костюмы** и **звуки**) и введи имя персонажа **Фото**.

В поле ввода имени костюма введи на английском языке название предмета или животного. Далее сделай копию костюма (кнопка **Копировать**), нажми под созданной копией кнопку **Редактировать**, с помощью кнопки **Импорт** загрузи в графический редактор следующее изображение. Отредактируй и сохрани костюм, назови костюм так, как называется изображённый предмет или животное. И так сохрани в проекте в виде костюмов для персонажа **Фото** все изображения. В результате у тебя получится 10 костюмов для персонажа **Фото** (рис. 4.37 слева).

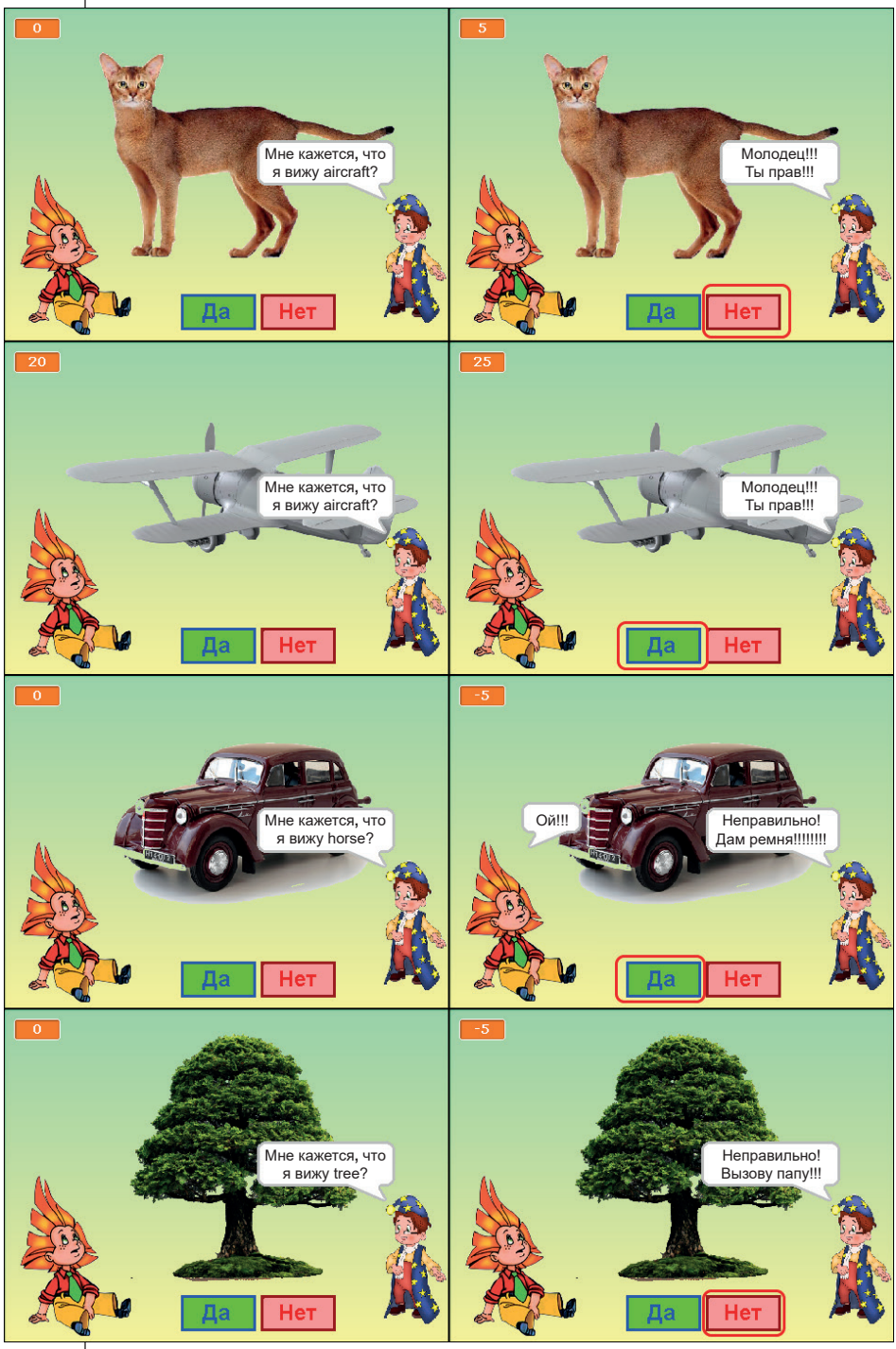


Рис. 4.36. *На четырёх верхних рисунках был дан правильный ответ, на четырёх нижних рисунках Незнайка ответил неправильно*

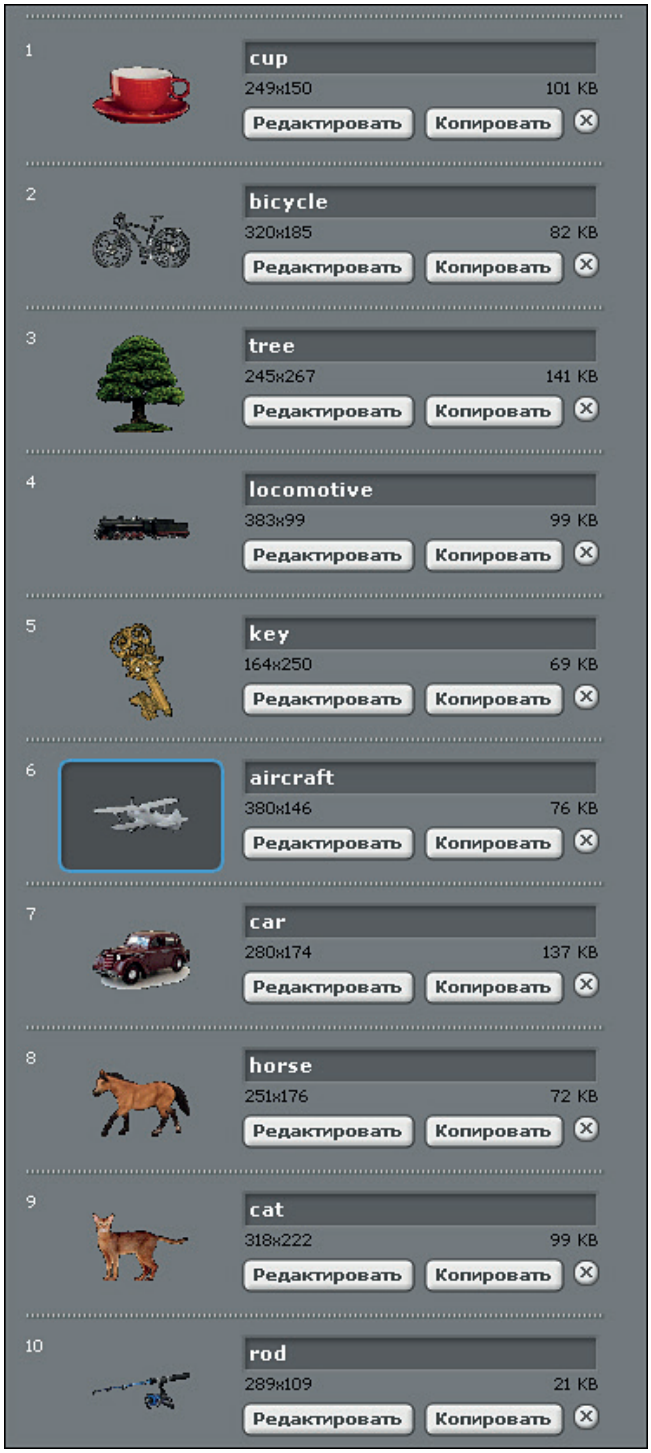


Рис. 4.37. *Костюмы для персонажа Фото*

Шаг второй: создаём персонажи Знайка, Незнайка, кнопку Да и кнопку Нет

Изображения персонажей **Незнайка** и **Знайка** загружаются из интернета. Кнопки **Да** и **Нет** нарисуй сам. В результате у тебя получится 5 персонажей: **Фото**, **Незнайка**, **Знайка**, кнопка **Да** и кнопка **Нет** (рис. 4.38).

Напомню, чтобы создать новый персонаж, нажми в панели управления правой нижней панели кнопку  , в графическом редакторе нажми кнопку **Импорт**, выбери папку, в которой ты сохранил изображение для персонажа, и загрузи изображение в графический редактор. Отредактируй изображение, закрой графический редактор. Щёлкни мышью в поле ввода в верхней части средней панели (над ярлыками **скрипты**, **костюмы** и **звуки**) и назови персонаж соответствующим ему именем. Чтобы ты не запутался в скриптах, назови персонажи так, как назвал их я.

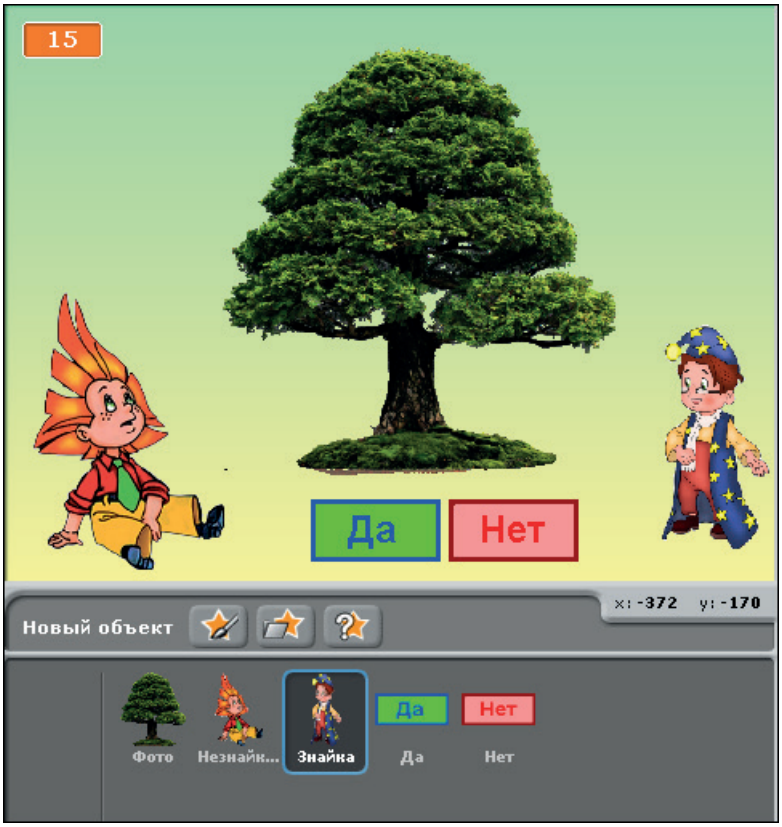


Рис. 4.38. *Пять персонажей создано*

Шаг третий: списки и переменные

Тебе нужно создать четыре списка и четыре переменные. Создай списки **Правильно**, **Соответствие**, **Накажу** и **Хвалю** (рис. 4.39).

Рис. 4.39. *Списки* ***Правильно***, ***Соответствие***, ***Накажу*** *и* ***Хвалю***

Обрати внимание: список **Правильно** предназначен для формирования фразы Знайки: «*Мне кажется, что я вижу…*». Продолжение этой фразы берётся из этого списка. Поэтому в этом списке строки с названиями предметов или животных должны соответствовать названиям костюмов персонажа **Фото** (рис. 4.37). Иначе на экране будет показываться один предмет, а Знайка его правильно назвать не сможет. Хотя он иногда умышленно будет называть предмет или животное неправильно.

Содержимое списка **Соответствие** позволит правильно сформировать переменную **А** (об этом чуть позже). С помощью этой переменной будет определяться, правильно ответил Незнайка или нет.

Списки **Накажу** и **Хвалю** помогают Знайке сформировать вторую часть фразы *Молодец!!!* или *Неправильно!* Здесь продолжение фраз ты можешь придумать сам. Да и количество строк тоже можешь увеличить. После того как списки будут созданы, скрой их.

Далее тебе нужно создать четыре переменные: **А**; **С**; **Баллы** и **Номер**.

Переменные **А** и **С** служат для правильной работы скриптов. Переменная **Номер** определяет номер костюма персонажа **Фото**, который будет отображён на экране для следующего вопроса. Кроме того, переменная **Номер** помогает определить номер строки из списка **Соответствие**.

Переменная **Баллы** показывает, сколько баллов ты заслужил. При каждом правильном ответе тебе будет прибавляться 5 баллов (это значение ты можешь изменить), при неправильном – 5 баллов у тебя будут вычитаться. Эту переменную следует отобразить на экране.

Переменная **С** позволяет Знайке сформировать вопрос. Если переменную **С** не вводить, Знайка будет называть предметы в соответствии с отображаемыми картинками. Гораздо интереснее, когда ты не знаешь, правильно Знайка назвал то, что изображено на экране, или нет.

Переменная **А** позволяет понять, правильно ответил Незнайка или нет.

Шаг четвёртый: скрипты

Объект **Фото** отвечает за смену картинки, формирование переменных **А** и **С** и завершение работы скрипта (конец игре) при нажатии клавиши **Пробел** (рис. 4.40).

Скрипт **Незнайка** управляет поведением и речами Незнайки (рис. 4.41).

Скрипт **Знайка** управляет поведением и речами Знайки (рис. 4.42).

Скрипты для кнопок **Да** и **Нет** передают нужные сообщения. Эти сообщения вырабатываются в зависимости от равенства значений переменных **А** и **С**. В скрипте для кнопки **Да**, если значение переменной **А** равно значению переменной **С**, передаётся сообщение **Да**. Если равенства нет, отправляется сообщение **Нет**. Скрипт для кнопки **Нет** действует наоборот: при равенстве значений **А** и **С** передаётся сообщение **Нет**; если **А** отличается от **С**, передаётся сообщение **Да**.

Щёлкни мышью в правой нижней панели на ярлыке персонажа **Фото**, открой вкладку **скрипты** и собери скрипт как на рис. 4.40. Щёлкни мышью на ярлыке персонажа **Незнайка** и собери скрипт как на рис. 4.41. Собери скрипты для Знайки, кнопки **Да** и кнопки **Нет**. Попробуй самостоятельно разобраться в работе скриптов. Подсказки в конце главы.

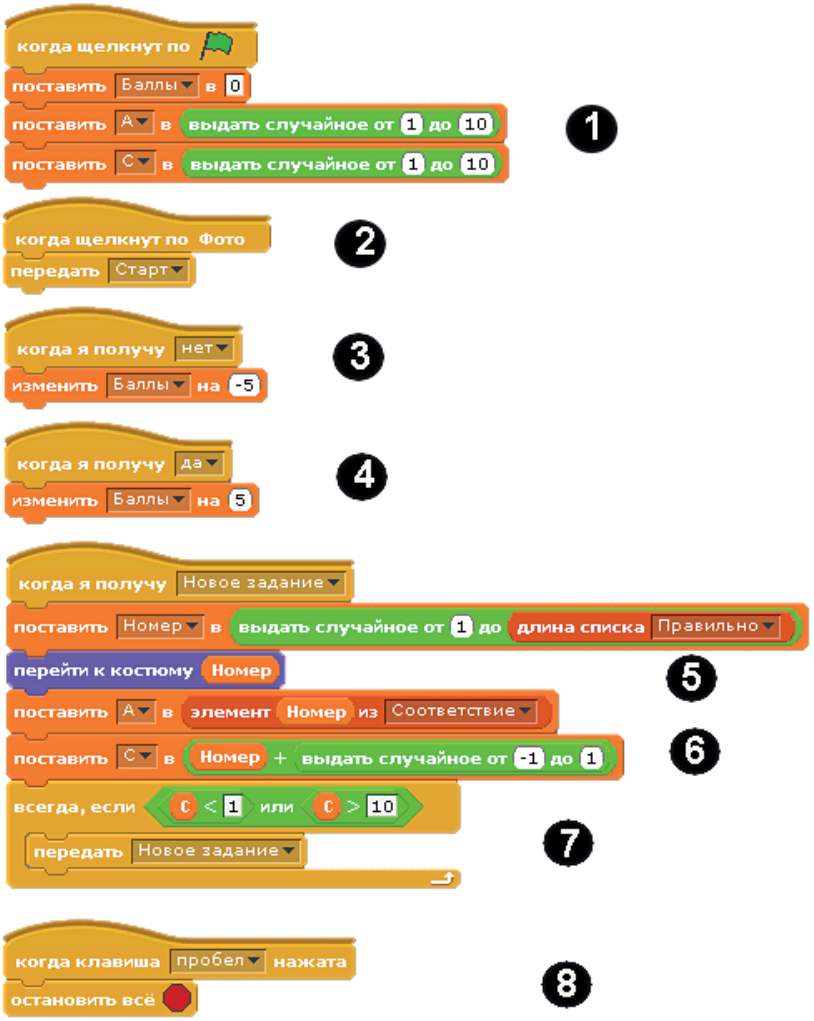


Рис. 4.40. *Скрипты для персонажа* ***Фото***

Рис. 4.41. *Скрипты для персонажа* ***Незнайка***

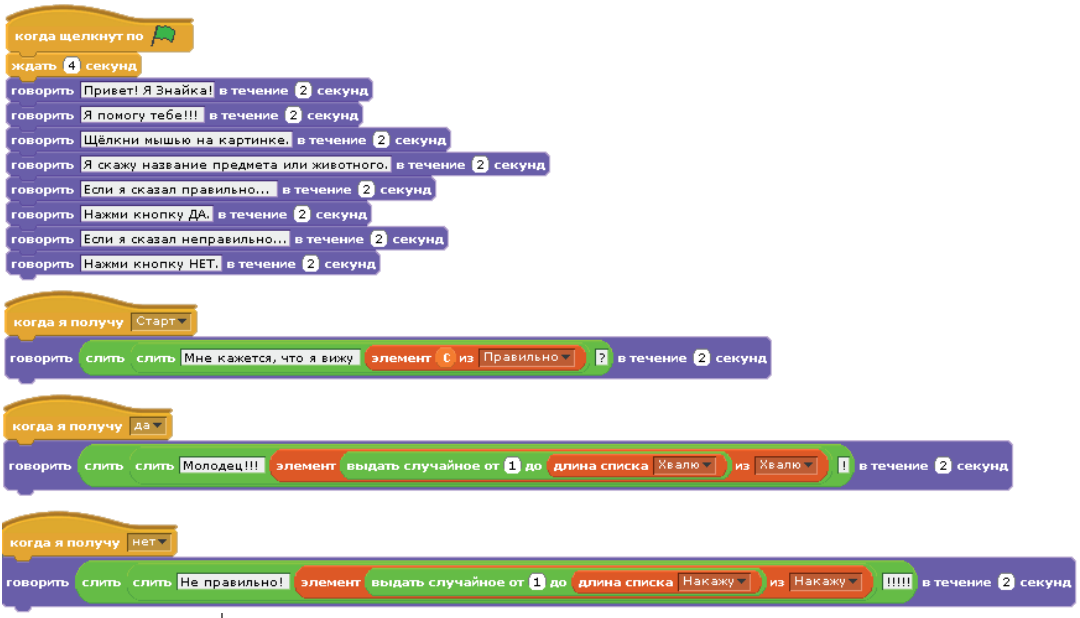


Рис. 4.42. *Скрипты для персонажа* ***Знайка***

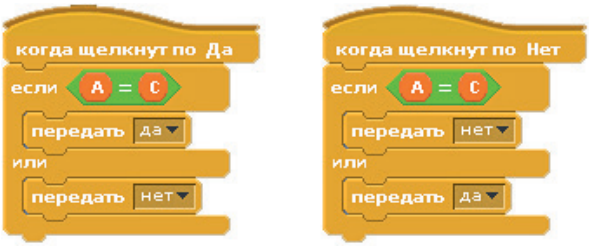


Рис. 4.43. *Скрипт для кнопки* ***Да*** *(слева) и* ***Нет*** *(справа)*

Ответы на вопросы

1. Ожидание и отправка сообщения.
2. Тебе нужно удалить все блоки, в которых содержится информация, что сообщение отправлено или принято.
3. а) Кто знает, где висит огнетушитель?
   б) Замок-молния.
4. Индекс, пусто.

Ответы на задания

Автогонки с несколькими сценами

Если автомобиль касается цветной линии, то на сцену поступает сообщение, и тогда сцена отвечает за смену фоновой картинки.

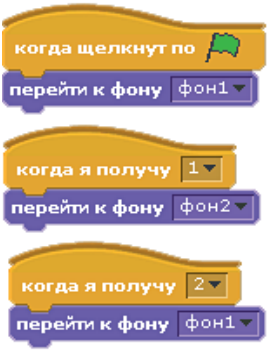


Рис. 4.44. *Скрипты для сцены*

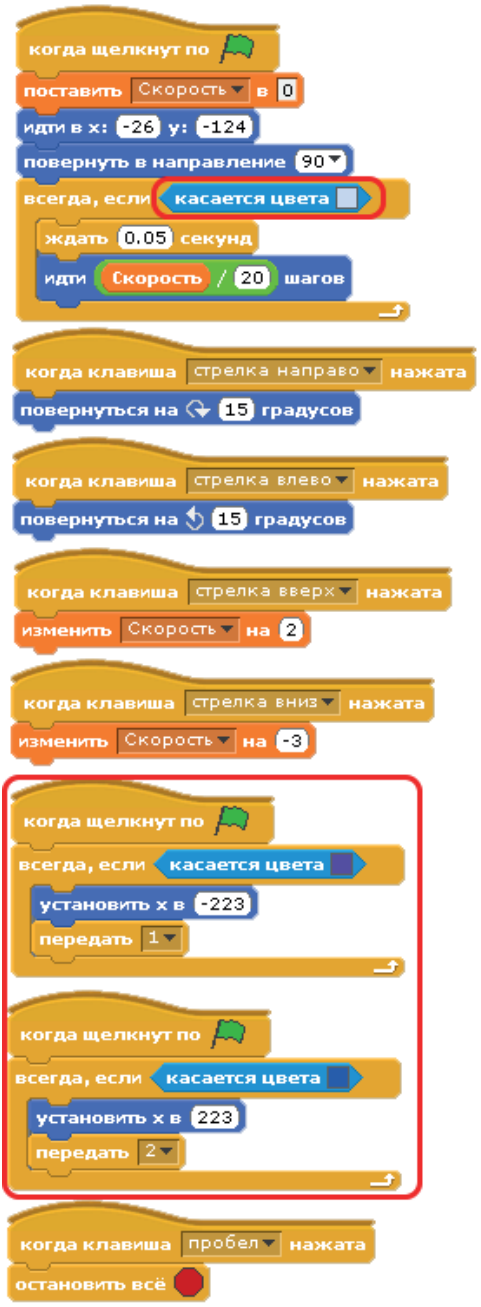


Рис. 4.45. *Дополнительные скрипты для объекта автомобиль*

Незнайка учит английский язык

Скрипт персонажа **Фото**.

1. Запуск программы, обнуление переменной **Баллы**, выбор случайных значений для переменных **A** и **C**. Эти переменные определяют конец фразы Знайки, задающего первый вопрос, и значение для получения правильного ответа.
2. Когда ты щёлкнешь мышью на изображении, будет сформировано сообщение **Старт**. Это сообщение получает скрипт Знайки, после чего Знайка задаёт вопрос.
3. При получении сообщения **Нет** (Незнайка ошибся) значение переменной **Баллы** уменьшается на 5 баллов. Сообщение **Нет** формирует скрипт кнопки **Да** или **Нет** (в зависимости от того, какая кнопка была тобой задействована).
4. При получении сообщения **Да** (Незнайка прав) значение переменной **Баллы** увеличивается на 5 баллов.
5. При получении сообщения **Новое задание** (генерируется скриптами персонажей Фото и Незнайка) формируется значение переменной **Номер**, далее с помощью этой переменой выбирается новый костюм (командный блок **перейти к костюму номер**). Изображения на экране должны появляться в случайном порядке. Поэтому значение переменной **Номер** формируется следующим образом: значение выбирается случайным образом в диапазоне от 1 до значения количества строк (длины списка) **Правильно** (рис. 4.46). Сейчас у нас в этом списке 10 строк. Поэтому диапазон получится от 1 до 10. Если длина списка **Правильно** изменится, автоматически изменится верхний предел диапазона значений, которые может принимать переменная **Номер**.

Рис. 4.46. *Командный блок для формирования значения переменной* ***Номер***

6. При получении сообщения **Новое задание** формируются значения переменных **A** и **C**, с помощью которых в скрипте Знайка при формировании вопроса выбирается строка из списка **Правильно**. Кроме того, сравнивая значения переменных **A** и **C**, скрипты кнопок определяют, правильный был дан ответ или нет. Обрати внимание, как формируется значение переменной **A** и **C**. Значение переменной **A** формируется так:

значение переменной **А** соответствует номеру строки, выбранной из списка **Соответствие** с помощью переменной **Номер** (рис. 4.47). То есть если из строки списка **Соответствие** переменная **Номер** определит третью строку, значение переменной **А** будет **3**. Это значение соответствует истинному номеру выбранного костюма. Далее эта переменная используется в скриптах кнопок.

Рис. 4.47. *Формирование значения переменной* ***А***

Переменная **С**, отвечающая при формировании вопроса в скрипте Знайка за выбор строки из списка **Правильно**, не должна всё время выбирать строку с настоящим названием отображаемого предмета или животного. Здесь должна быть загадка. Знайка в вопросе может правильно назвать отображаемый предмет, а может произнести название предмета из соседней строки (сверху или снизу). Поэтому нам нужно сделать этот выбор в некоторой степени случайным (рис. 4.48).

Рис. 4.48. *Формирование значения переменной С*

Для этого нам нужно изменить случайным образом значение переменной **Номер** в диапазоне от –1 до +1. То есть, например, мы получили значение переменной **Номер** равным **3**. На экране отобразится изображение дерева, и значение переменой **А** будет равно **3**. Вычисляем значение переменной **С**. Командный блок **выдать случайное**, допустим, выдает число **–1**. В результате значение переменной **С** примет значение **2**. В этом случае скрипт Знайки сформирует вопрос: «*Мне кажется, что я вижу bicycle?*» Знайка будет не прав, потому что на экране изображение дерева.

7. В то же время скрипты кнопок также получили значения **А = 3**, **С = 2**. Если ты нажмёшь кнопку **Да**, скрипт сравнит значения **А** и **С**, увидит, что **А** не равно **С**, и передаст сообщение **Нет**. Это сообщение получит скрипт Знайки и в соответствии с программой поругает тебя. Если нажать кнопку **Нет**, скрипт увидит, что значение переменной **А** не равно значению переменной **С**, и выдаст сообщение **Да**. Данные сообщения получит скрипт Незнайки и в соответствии с полученным сообщением с задержкой

в 2 секунды (ждет, пока Знайка произнесёт свою фразу) скажет «*Ура!*» или «*Ой!!!*» и передаст сообщение **Новое задание**. Сообщение **Новое задание** получит скрипт Фото, сгенерирует новое значение переменной **Номер**, в соответствии с этим сообщением перейдет к новому костюму с номером, соответствующим текущему значению переменной **Номер**, сформирует новые значения **А** и **С** и снова станет ждать сообщения **Новое задание**.

Кроме того, сообщение **Да** или **Нет** получит скрипт Знайка и в соответствии с полученным сообщением похвалит или поругает Незнайку.

Но у нас может случиться коллизия. Представь себе, что текущее значение переменной **Номер** будет 10, скрипт **выдать случайное** выдаёт значение 1 (рис. 4.48). После суммирования мы получаем значение для переменной **С**, равное 11. Но длина списка **Правильно** равна 10. Знайка произнесёт первую часть фразы «*Мне кажется, что я вижу…*» и больше ничего не произнесёт. Такой же результат будет, если значение переменной **Номер** будет 1, случайное –1. В результате значение переменной С получится 0. Но в списке нулевой строки нет! В этом случае нам следует в скрипт встроить командный блок **всегда, если** (рис. 4.49).

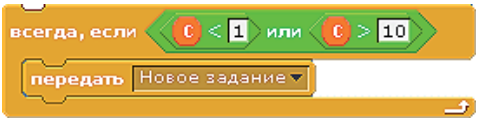
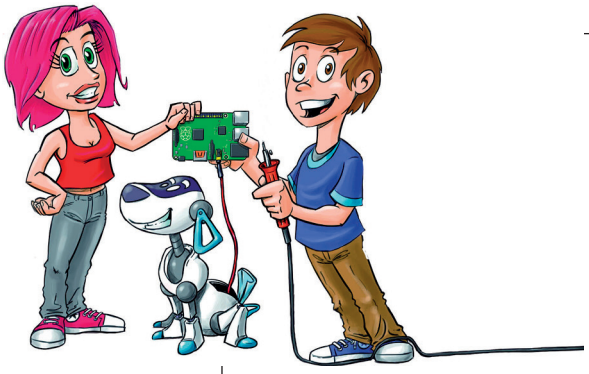


Рис. 4.49. *Командный блок* ***всегда, если***

Нам нужно сравнить значение **С** с 1 и со значением длины списка. Если значение **С** меньше 1, значит, нам нужно изменить значение переменной **Номер**. Это мы можем сделать, послав сообщение **Новое задание**. Такое же сообщение будет послано, если значение **С** будет больше значения длины списка **Правильно**. В нашем случае это 10. Для этого в командный блок **всегда, если** мы встроили оператор **ИЛИ** и два оператора, которые сравнивают значение переменной **С** с 1 и с 10. То есть если значение **С** меньше 1 **ИЛИ** больше 10, командный блок посылает сообщение **Новое задание**.

8. Этот командный блок после нажатия клавиши **Пробел** завершает работу всех скриптов программы.

Я думаю, что объяснять работу скриптов **Незнайка**, **Знайка**, кнопка **Да** и кнопка **Нет** не нужно. Ты и сам во всём сможешь разобраться.



5 Создание проектов с помощью Picoboard

Устройство Picoboard от компании «Спаркфан» (Sparkfun Electronics) – это небольшая плата красного, голубого или жёлтого цвета, подключаемая через USB-кабель к USB-порту. В эту плату встроены датчик освещения, микрофон, кнопка, ползунковый регулятор и четыре разъёма. Благодаря этому устройству твой Raspberry Pi получит «глаза» и «уши». Это устройство, прежнее название Picoboard-Scratchboard, разработано командой из компании «Lifelong Kindergarten» Массачусетского технологического института (MIT) в США и используется совместно с Scratch. В этой главе мы рассмотрим несколько проектов, с помощью которых ты познакомишься с техническими возможностями Picoboard. Учти, часто за её несколько шутливыми проектами стоит вполне серьёзная наука.

Плата Picoboard

Для создания проектов из этой главы тебе необходимо обзавестись платой с датчиками Picoboard компании «Спаркфан» (Sparkfun). Заказать её можно на сайте http://www.sparkfun.com. Стоимость этого устройства менее 3000 руб. Если же эту плату заказать на других сайтах, можно столкнуться с посредниками, и её стоимость возрастёт.

Если ты, следуя описаниям в этой книге, установил операционную систему Raspbian, то подключённый к компьютеру Picoboard заработает моментально. Просто подключи плату через переходник к USB-порту и принимайся за проекты (рис. 5.1 левый). Никакого другого программного обеспечения тебе больше устанавливать не понадобится. Проверить состояние датчиков тебе поможет группа команд **Ощущать** программы Scratch.

В комплект поставки Picoboard, кроме самой платы, входят четыре двужильных кабеля. На одном конце кабеля установлен разъём для подключения кабеля к плате, а на другом конце каждой жилы – пружинный зажим типа «крокодил». Обычно это разъёмы красного и чёрного цветов (рис. 5.1 правый нижний). Для подключения Picoboard к Raspberry Pi тебе понадобится ещё мини-кабель USB (рис. 5.1 правый верхний). Он может быть разной длины и разного цвета. В комплект поставки этот кабель не входит. Если дома такого кабеля не найдётся, его придётся купить.

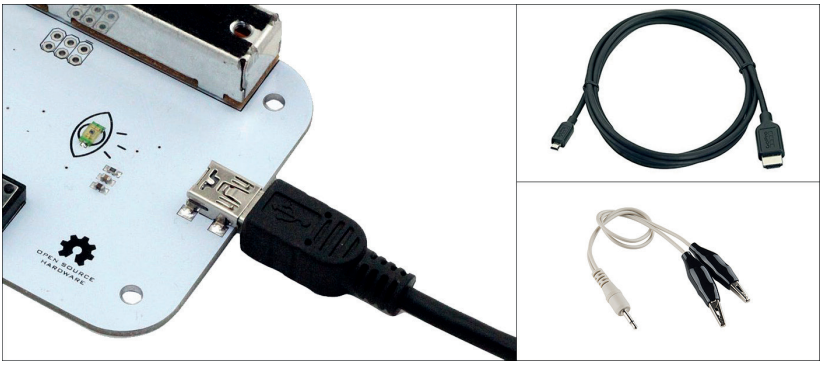


Рис. 5.1. *Так выглядит подключённый разъём мини-USB (слева), переходник мини-USB – USB (справа вверху) и кабель для измерения сопротивления (справа внизу)*

Сотрудничество с родителями

Для многих проектов в этой книге тебе, кроме Raspberry Pi, понадобятся и другие детали, список которых ты найдёшь ниже, но их не так-то просто найти в магазинах. Для заказа дополнительных элементов тебе, видимо, потребуется помощь родных. В приложении к книге есть глава с указаниями для родителей и воспитателей. Вероятно, тебе придётся показать им эту книгу, чтобы они сами прочитали её. Помощь родных и близких всегда полезна.

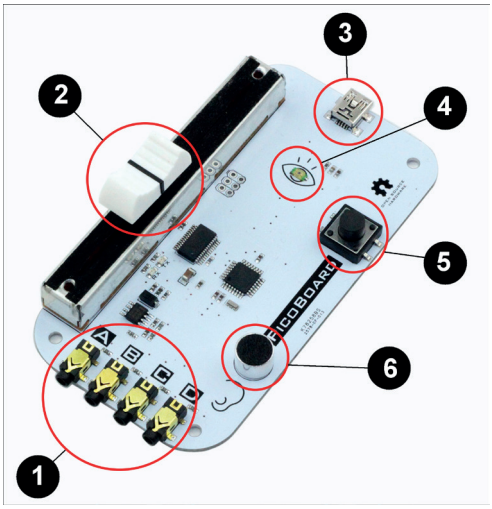


Рис. 5.2. *Датчики, разъёмы и элементы управления, смонтированные на плате Picoboard*

На плате Picoboard установлены следующие датчики, разъёмы и элементы управления:

1. четыре разъёма, предназначенных для разъёмов, помечены буквами A, B, C и D. Их называют резисторными входами, потому что через них измеряется подключённое электрическое сопротивление;
2. ползунковый регулятор (рычажок);
3. мини-разъём USB для соединения с компьютером;
4. датчик освещения в обрамлении нарисованного глаза;
5. кнопка;
6. микрофон, расположенный рядом с нарисованным ухом.

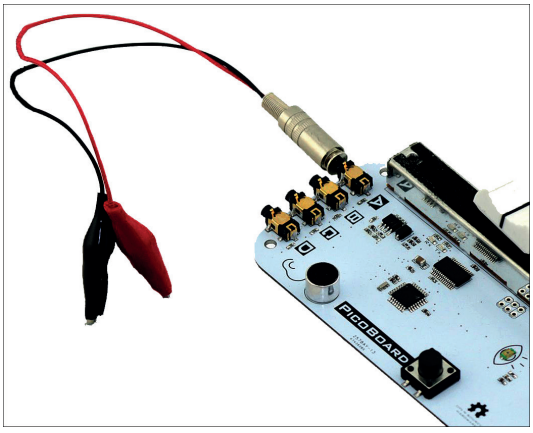


Рис. 5.3. *Устройство Picoboard с подключенным двужильным кабелем для измерения сопротивления*

Тестируем Picoboard

Сенсор (датчик) – это устройство, которое принимает физическую величину (например, свет или звук) и преобразовывает её в электрический сигнал. Picoboard создаёт из такого сигнала цифровое значение (код) в диапазоне от 0 до 100. Далее полученное значение передаётся Raspberry Pi. Как определить, отправляет ли сенсор код? Попробуй сделать следующее.

- Подключи Picoboard через USB-кабель к Raspberry Pi.
- Запусти Scratch.
- В левой верхней панели щёлкни мышью на кнопке **Сенсоры**. В левой нижней панели появится набор команд, входящих в данную группу.
- Щёлкни правой кнопкой мыши на блоке команд **значение сенсора** или **сенсор**. На экране появится контекстное меню (рис. 5.4 слева).
- Выбери в появившемся контекстном меню команду **показать данные ScratchBoard**. На сцене появится панель, в которой ты увидишь значения сенсоров, установленных на плате Picoboard.

Что обозначают эти значения? Значение **Рычажок** показывает положение движка регулятора. Если же, например, датчик света не освещён, его значение будет равно 0. В зависимости от интенсивности освещения будет изменяться значение в поле **свет**. Если взглянуть на значения разъёмов **A**–**C**, то увидим значение 100. Это значит, что значение сопротивления на этих разъёмах велико.

Рис. 5.4. *Проверка подключения платы Picoboard и выбор порта в случае, если плата не обнаружена*

Поэкспериментируй с датчиками и рычажком. Передвинь ползунок регулятора вверх или вниз. Значение в поле ввода **Рычажок** изменится (рис. 5.4 слева второй).

- Прикрой пальцем датчик освещения. Значение в поле ввода **свет** изменится и будет стремиться к 0.
- Произнеси любое слово. Значение в поле ввода **звук** также изменится.

Если же после подключения платы к компьютеру у тебя в полях ввода будут нули, значит, компьютер плату не увидел. В этом случае выполни следующие действия.

- Щёлкни правой кнопкой мыши на панели со значениями сенсоров. Появится контекстное меню.
- Выбери из появившегося контекстного меню команду **Выбрать серийный/usb порт**. Появится список COM-портов (рис. 5.4 правый).
- Выбери из появившегося списка нужный порт (возможно, тебе придётся последовательно перебрать каждый порт или попросить помощи у родителей).

Проект 10. Магические слова – распознавание речи

Сегодня управлять компьютерами и мобильными телефонами можно с помощью голоса. В этом проекте ты создашь очень простую и в то же время очень действенную функцию голосового управления. При этом будет использован датчик звука (микрофон) Picoboard.

Как это выглядит?

После того как ты запустишь программу распознавания, есть звук или нет, ты увидишь на сцене тёмный фон (рис. 5.5 слева). Произнеси по слогам слово **Темно**. Один слог нужно чётко отделить от другого. Если на экране тёмный фон, ничего не произойдёт. Скажи слово **Свет**. Фон сцены станет светлым (рис. 5.5 правый). Ты можешь повторять это много раз.

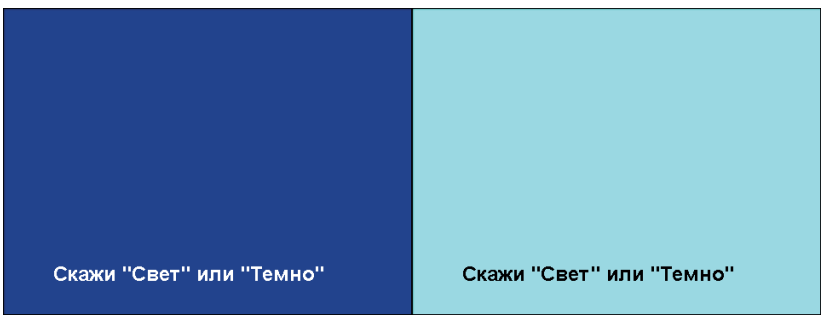


Рис. 5.5. *Так выглядит сцена при голосовом управлении*

Создание проекта

Создай новый Scratch-проект и удали кота. Кот в этом проекте нам не понадобится. В этом Scratch-проекте вообще нет никаких фигур (объектов), а есть лишь сцена с двумя фоновыми картинками.

Создание фона

Поскольку никаких других объектов у нас нет, то сцена выбирается автоматически.

- Выбери вкладку **фоны**, а затем нажми кнопку **Редактировать**. Откроется окно графического редактора.
- Выбери в палитре тёмный цвет, нажми кнопку [🪣] и залей фон выбранным цветом.
- Выбери инструмент [T] – **Текст**, выбери белый цвет и скажи *«Свет» или «Темно»*. Сохрани изменения, нажав кнопку **ОК**.

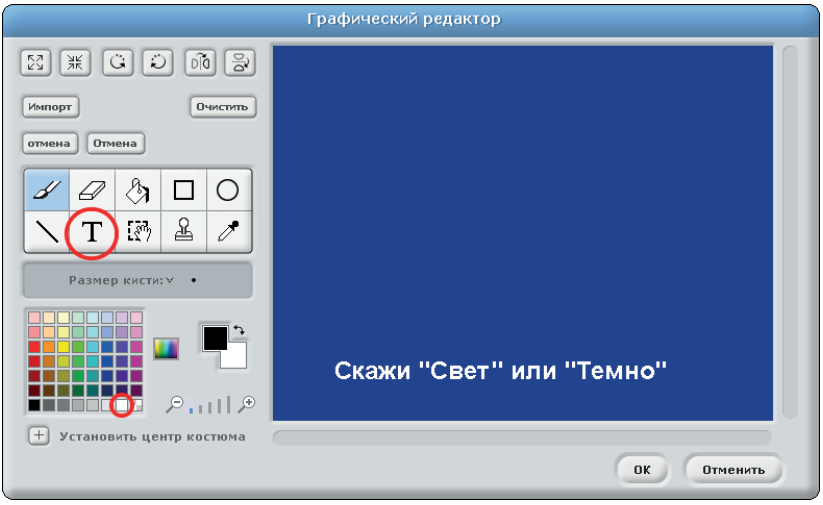


Рис. 5.6. *Создание фона с помощью графического редактора*

- Нажми кнопку **Копировать**, расположенную на вкладке **фоны** под полем ввода **фон1**. Ниже появится поле ввода с именем **фон2**.
- Нажми под полем ввода **фон2** кнопку **Редактировать**.
- Измени цвет фона на светлый. Для этого выбери инструмент [🪣], выбери в палитре цветов светлый фон и залей им фон сцены.

- Выбери инструмент T и щёлкни мышью в палитре цветов на темном образце цвета. Текст фона сразу окрасится в выбранный цвет.
- Сохрани внесённые изменения, нажав кнопку **OK**. Теперь у тебя будет два фона с текстом: светлый и тёмный (рис. 5.5).
- Переименуй фоны. Для этого на вкладке **фоны** введи в поле ввода **фон1** имя **Тёмный фон**, а в поле ввода **фон2 – Светлый фон**.

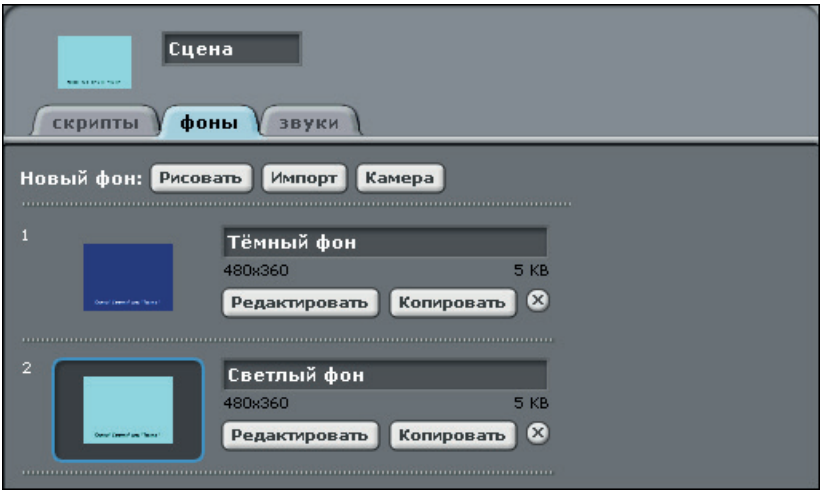


Рис. 5.7. *Переименованные фоны сцены*

Скрипт

Сцена управляется скриптом, показанным на рис. 5.9. Особенность этого скрипта в том, что в нём содержится несколько командных блоков с вложенными выражениями (рис. 5.9). Эти выражения на первый взгляд выглядят несколько замысловато (рис. 5.8) и собираются пошагово.

После того как строка с блоком команд будет собрана, введи в поле ввода1 значение **10**, а в поле ввода 2 – значение **1**.

А теперь вернёмся к скрипту. Суть идеи очень проста. Скрипт определяет, состоит ли произнесённое слово из одного слога (*свет*) или же из двух слогов (произнесенного по слогам *тем-но*). Компьютер проверяет, звук на протяжении секунды появится один или два раза.

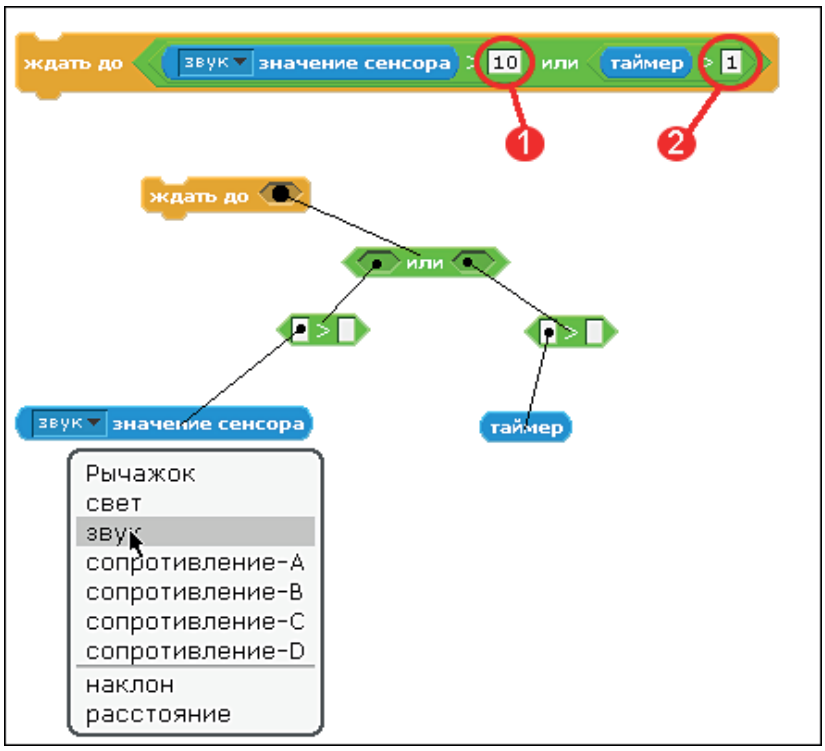


Рис. 5.8. *Создание команды с помощью вложенного выражения*

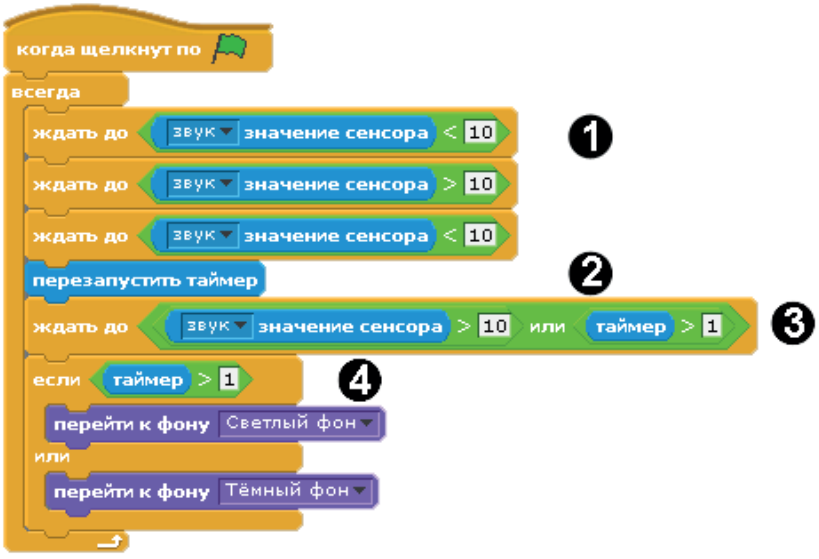


Рис. 5.9. *Скрипт для сцены*

Пояснения

Теперь более подробно рассмотрим внутреннюю часть замкнутого цикла (то есть пока работает программа бесконечно повторяющихся действий). Там есть четыре командных блока **ждать до**.

1. Вначале мы ждём, когда настанет тишина. Компьютеру требуется время, прежде чем он услышит какой-то звук, а именно звук произнесенного первого слога. Далее, когда первый слог уже прозвучал, компьютер ожидает новый звук.
2. Чтобы не пришлось ожидать целую вечность, компьютер использует таймер. В этой инструкции таймер перезапускается. При этом его значение должно оставаться на нулевом уровне.
3. С помощью этого командного блока компьютер ждёт новый звук. Когда первый звук был зафиксирован, ожидает, поступит ли на протяжении секунды ещё один звук. Время ожидания задаётся переменной. Таймер >1 означает, что его значение более единицы.
4. Если же прошло более секунды, а новый звук так и не появился, то компьютер фиксирует, что прозвучал *один* слог. В этом случае фон становится/остаётся светлым. Если это не так (зафиксировано более одного звука за секунду), фон остается/переключится на тёмный.

Проект 11. Создаём измеритель уровня звука

«А теперь пусть зрители решат, кто стал победителем», – объявляет ведущий на сцене. «Если первая команда вам понравилась больше, тогда поприветствуйте их сейчас аплодисментами, можно даже потопать ногами… Боже мой, вот это овации! Большое спасибо! А теперь следующий участник. Те, кто проголосовал за вторую команду, поддержите своих любимчиков как можно громче…»

Возможно, ты уже посещал ток-шоу, где публика с помощью аплодисментов выражала своё одобрение. В таких местах всегда присутствует измеритель уровня звука. Это устройство измеряет уровень шума. С помощью Picoboard и Scratch ты с лёгкостью научишься создавать его.

Как выглядит измеритель уровня звука?

Измеритель уровня звука – это индикатор в виде круга. Чем громче звук, тем больше размер круга на экране. Чем тише, тем размер круга меньше. Другими словами, круг демонстрирует текущий уровень громкости.

Значение **Звук** в правом верхнем углу показывает уровень громкости. Когда тихо, ты увидишь цифру 0. Чем громче, тем большее значение ты увидишь в этом поле ввода. Наибольшее значение – это 100. Это очень важная функция в определении решения зрителей. Чтобы запустить измеритель уровня звука, нажми клавишу **Пробел**. На экране появится заставка с более светлым фоном и надписью «Для остановки нажми клавишу Пробел». Снова на экране появится первая заставка с тёмным фоном, но в поле **звук** ты увидишь зафиксированный в прошлом измерении уровень громкости. Ты можешь записать это число, а потом выполнить ещё одно измерение.

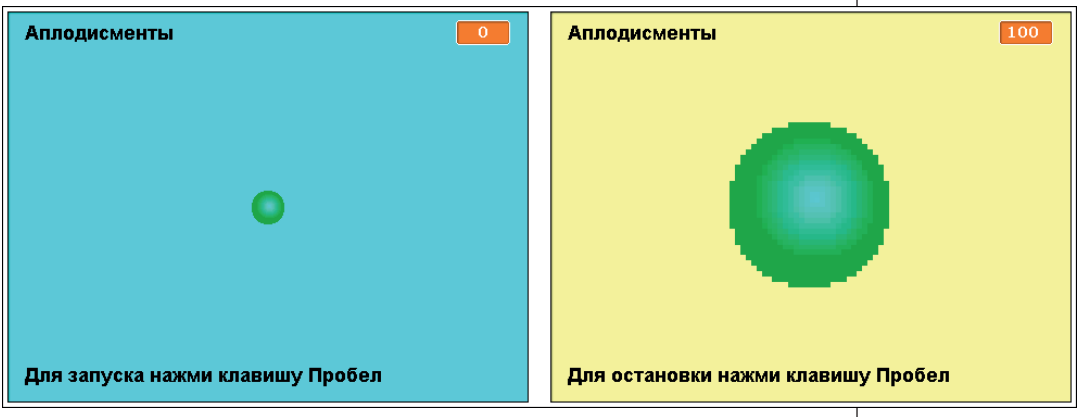


Рис. 5.10. *Площадь круга показывает текущий уровень громкости*

Как построить измеритель уровня звука?

Прежде всего создай переменную **Максимум**.

- В правой верхней панели нажми кнопку **Переменные**.
- В правой нижней панели нажми кнопку **Создать переменную**.
- Назови ее **Максимум**.
- На сцене появится поле новой переменной. Щёлкни правой кнопкой мыши на этом поле и выбери из появившегося контекстного меню команду **велик для чтения**.
- Перемести это поле в правый верхний угол сцены

Объект

Текущий уровень громкости представляется изменением размера объекта.

- Нажми кнопку , расположенную в верхней части правой нижней панели. Появится окно графического редактора.
- Нарисуй круг небольшого размера как на рис. 5.10.

Ты можешь и сам придумать, как будет выглядеть индикатор звука. Это может быть полоса, длина которой будет увеличиваться при увеличении уровня громкости и уменьшаться, когда звук становится тише. Может быть, это будут хлопающие ладошки, или ухо, или что-нибудь другое.

- Собери скрипт для объекта, как показано на рис. 5.11.

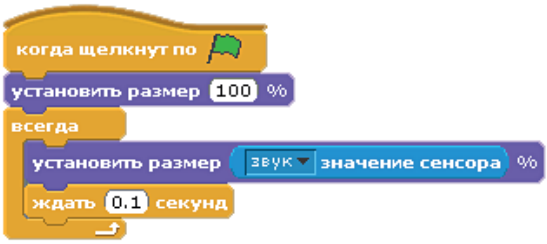


Рис. 5.11. *Скрипт объекта*

Скрипт содержит бесконечное количество повторений. Размер объекта всегда установлен на том значении, которое соответствует текущему уровню шума.

Сцена

Для сцены мы сделаем два фона.

- Щёлкни мышью на значке фона **сцена** в левой части правой нижней панели и в средней панели выбери вкладку **фон**.
- Нажми кнопку **Редактировать** и залей фон, например, светло-голубым цветом.
- Напечатай текст: «Аплодисменты Для запуска нажми клавишу Пробел».

Обрати внимание. Чтобы слово «Аплодисменты» расположилось в левом верхнем углу, а строка «Для запуска нажми клавишу Пробел», сделай следующее. Напечатай слово «Аплодисменты» и дальше нажимай клавишу ↵ **Enter**, пока текстовый курсор не окажется в левом нижнем углу сцены. После этого напечатай «Для запуска нажми клавишу Пробел».

- На вкладке **фоны** нажми кнопку **Копировать**. Ниже появится копия фона с именем **фон2**.
- Залей этот фон, например, светло-жёлтым цветом.
- В строке «Для запуска нажми клавишу Пробел» замени слово «запуска» на слово «остановки». Для этого выбери инструмент **Текст**, щелкни мышью за словом «запуска». Удали его, нажимая клавишу T, после чего напечатай слово «остановки».
- Переименуй **фон1** на **Пуск**, а **фон2** – на **Останов**.
- Теперь соберем для сцены два скрипта, как показано на рис. 5.12.

Рис. 5.12. *Два скрипта для сцены*

У сцены есть два скрипта. Первый (рис. 5.12.слева) запускается при нажатии на кнопку . Второй скрипт (рис. 5.12 справа) вводится в случае, если пользователь нажал клавишу **Пробел**. Четвёртый командный блок **ждать 0,5 секунд** (❶) – это команда, позволяющая компьютеру подождать, чтобы клавиша **Пробел** перешла в отжатое состояние.

В цикле (повтор действия) в переменной **Максимум** сохраняется значение наибольшего уровня звука. Если же уровень выше, чем **Максимум**, его нужно обновить (❷). На сцене это значение демонстрируется в поле оранжевого цвета, который мы ранее расположили в правом верхнем углу сцены.

Уровень громкости измеряется до тех пор, пока ты вновь не нажмёшь клавишу **Пробел**. После этого фон «прыгнет» на первую картинку, и работа скрипта завершится. Значение переменной **Максимум** более не изменится.

Измерение среднего уровня громкости

Ранее созданный измеритель уровня звука в большинстве случаев работает хорошо. Но есть один недостаток. Малейший шорох вблизи микрофона обеспечит максимальный уровень громкости, даже если вокруг совсем тихо. Громкость зрительских оваций прибор измеряет безупречно, когда совершается большое количество измерений, и он всегда показывает среднее значение. Размер круга изменяется медленно.

Переменные

Тебе понадобятся ещё две переменные – одну назовём **Сумма**, другую – **Среднее значение**. Только индикатор переменной **Среднее значение** должен отображаться на сцене. Переменная **Сумма** содержит в себе сумму измеряемого уровня громкости. Она нужна нам, чтобы рассчитать среднюю величину.

- Создай переменные **Сумма** и **Среднее значение**.

Сцена

Фоны для сцены мы оставим как есть.

Рис. 5.13. *Два скрипта для сцены*

Когда нажата клавиша **Пробел**, изменяется громкость.

Как это работает

1. Подожди немного, пока после включения снизится шум.
2. Для измерения среднего значения нам понадобится точное время. Поэтому показания таймера сбрасываются на 0.
3. Далее компьютер с периодом в 0,1 с совершает большое количество измерений.
4. Измеряемое значение прибавляется к значению переменной **Сумма**, и значение **Сумма** возрастает прямо пропорционально количеству проведённых измерений.
5. Теперь немного математики. Среднее значение – это сумма всех измерений, поделённое на количество измерений.

   Пример. При двух измерениях, равных 50 и 60 единицам, среднее значение будет 55 ((50 + 60) : 2 = 55). При трёх измерениях в 41, 40 и 42 единицы среднее значение равно 41 ((41 + 40 + 42) : 3 = 123 : 3 = 41). Здесь в каждой секунде содержится десять измерений. Поэтому ты должен поделить значение **Сумма** на значение таймера, умноженное на 10.

Проект 12. Игра «Пинг-понг»

В этом проекте ты будешь использовать рычажок Picoboard (движок регулятора), чтобы управлять ракеткой для игры в пинг-понг. Как ты уже понял, эта игра так и называется – «Пинг-понг». Это одна из старейших видеоигр. Идея очень проста. Через весь экран пролетает мячик. Он отскакивает от бокового и верхнего краёв экрана. Чёрный треугольник – это ракетка. Когда мячик касается её, то отскакивает вверх. Ты с помощью рычажка (ползунка регулятора) двигаешь ракетку вправо или влево и стараешься не допустить падения мячика вниз, на зелёную часть экрана. Если ты не успел отбить мячик и он всё-таки упал вниз – игра окончена.

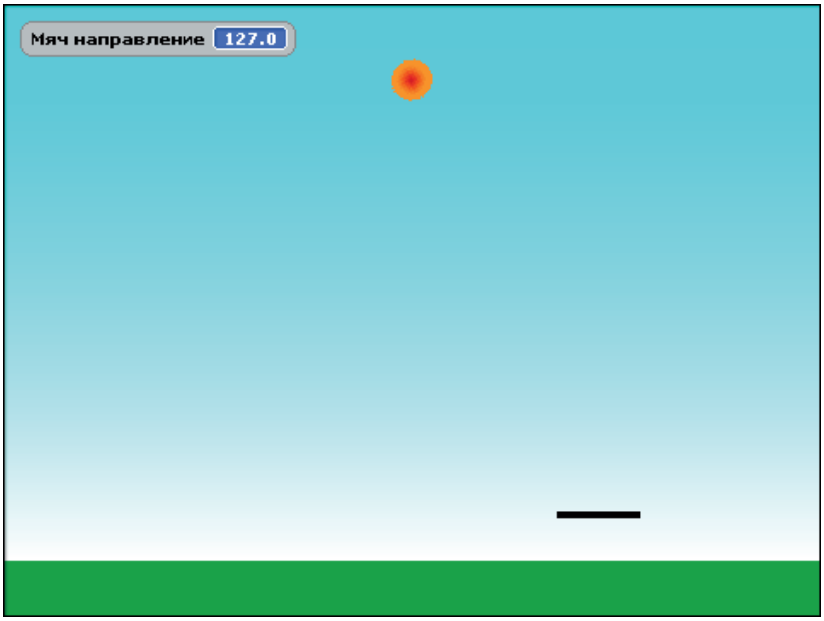


Рис. 5.14. *Фотография с экрана игры в пинг-понг*

Фон сцены имеет внизу зелёную ленту и только два подвижных объекта – ракетку и мячик.

Создай новый проект, удали кота и нарисуй фон для сцены (зелёный низ и бело-голубой фон неба) как на рис. 5.14. Чтобы нарисовать фон, перейди на вкладку **фоны**, нажми кнопку **Редактировать** и нарисуй в графическом редакторе фон. Зелёная трава рисуется с помощью инструмента  . Выбери этот инструмент, под панелью инструментов нажми кнопку  , выбери в палитре цветов нужный цвет и нарисуй сплошной зелёный прямоугольник внизу сцены. Чтобы нарисовать небо (горизонтальный бело-голубой фон), выбери инструмент  , выбери в палитре цветов голубой цвет. Этим цветом заполнится верхний левый прямоугольник в группе из двух прямоугольников  , расположенных правее палитры цветов. Нажми кнопку  и залей верхнюю части сцены. Закрой графический редактор, нажав кнопку **ОК**. Фон создан.

Летящий мяч

На рис. 5.15 наглядно представлено происходящее, а также имеется скрипт для мяча.

Нажми кнопку  в панели управления правой нижней панели и нарисуй в появившемся графическом редакторе

мячик. Далее нам необходимо собрать для мячика скрипт. Скрипт для мячика показан на рис. 5.15.

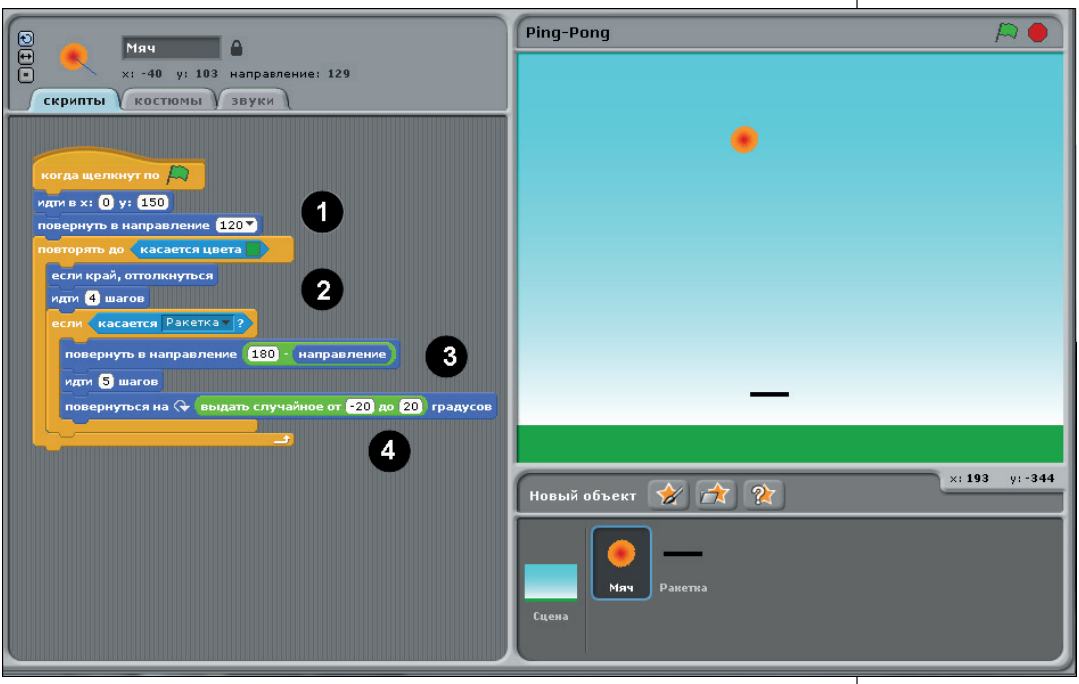


Рис. 5.15. *Скрипт для мячика*

Как это работает

1. Для запуска скрипта необходимо щёлкнуть мышью на флажке . Сначала мяч отправляется вверх под углом 120° и касается верхнего края экрана ближе к центру. После отражения от верхнего края экрана мяч движется вниз под углом 120°. В программе Scratch направление 0° указывает движение вверх, а положительный градус угла означает движение по часовой стрелке. Конечно же, ты можешь ввести и другое значение угла, например в 100°. Поэкспериментируй со значениями и посмотри, что у тебя получится.
2. Командный блок **если край, оттолкнуться** срабатывает только тогда, когда мяч касается правого, левого или верхнего края экрана.
3. Как только мяч коснулся края экрана, сразу меняется направление движения мяча, край отражает мяч по принципу «угол падения равен углу отражения». Схема изменения направления движения мяча показана на рис. 5.16. Уясни для себя, что угол направления движе-

ния мяча перед касанием (этот угол обозначен дугой серого цвета) и направление движения после касания (этот угол обозначен дугой оранжевого света) в сумме равны 180°.

4. Если мячик касается ракетки, он ещё раз меняет направление движения. Направление движения заранее неизвестно. Таким образом, в игру вводится случай. Так будет интереснее. Рисунок 5.17 демонстрирует процесс движения мяча.

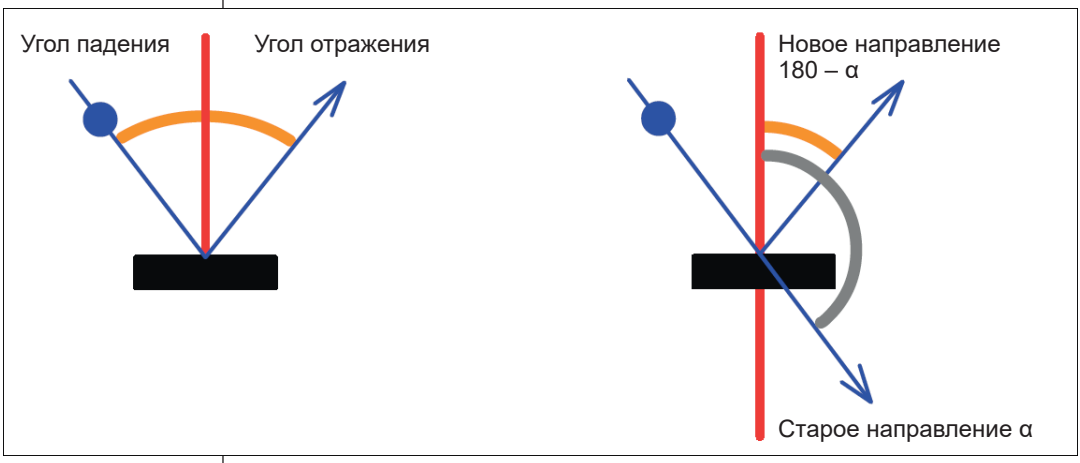


Рис. 5.16. *Угол направления полёта изменяется, если мяч касается ракетки*

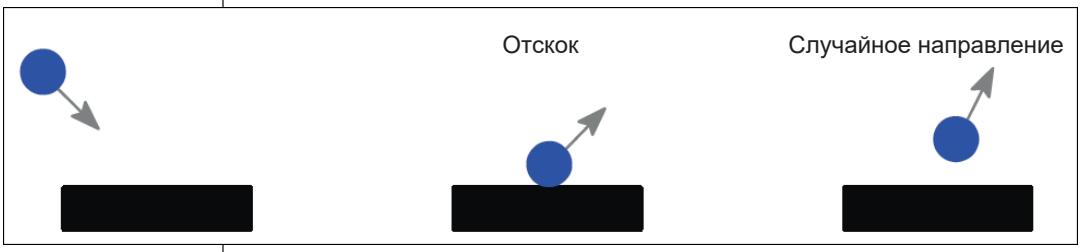


Рис. 5.17. *Процесс движения мяча во время касания ракетки*

Бесконечные расчёты с помощью углов кажутся довольно сложными, поэтому в заключение дам тебе маленький совет. Чтобы лучше понимать, как работает программа, ты можешь посмотреть текущий угол движения мяча. Для этого в коллекции командных блоков **движение** установи левее командного блока **направление** флажок.

Рис. 5.18. *Ты можешь смотреть текущий угол движения объекта*

Ракетка

Ракетка может двигаться в горизонтальном направлении с помощью рычажка Picoboard. Позиция *x* вычисляется из значения рычажка (движка регулятора).

- Щелкни в правой нижней панели на ярлыке ракетки, открой вкладку **Скрипты** и собери скрипт как на рис. 5.19.

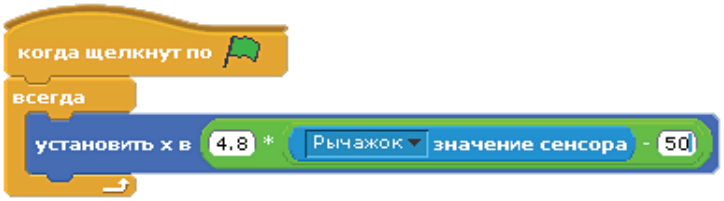


Рис. 5.19. *Скрипт для ракетки*

Формула выглядит следующим образом:

.

Как вычислить это значение? Как и остальные датчики, рычажок (ползунок регулятора) по умолчанию имеет значение в диапазоне от 0 до 100. Если ползунок находится в левом крайнем положении, то значение этого датчика будет равно 0. Если перевести ползунок регулятора в крайнее правое положение, его значение станет равным 100. Вспомни, насколько широкой является сцена. При крайнем левом положении ползунка регулятора координата X будет равна **–240** (минус двести сорок). При положении ползунка в крайнем правом положении координата X равна **240**. То есть нам необходимо рассчитать диапазон значений ползунка (рычажка) от 0 до 100 в координаты X от –240 до 240.

Вычитая от значения положения ползунка регулятора (рычажка) число 50, ты получаешь текущее значение положения рычажка –50 единиц (влево) или текущее значение положения рычажка +50 (вправо). После умножения полученного ранее значения на 4.8 получаем координату X –240 (полностью влево на сцене) или X +240 (полностью вправо на сцене).

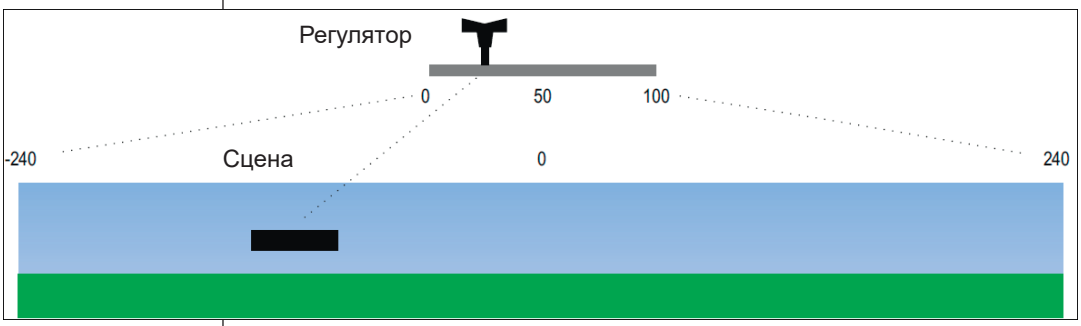


Рис. 5.20. *Связь между положением ползунка регулятора (рычажка) и X – координатой положения ракетки*

Поговорим о параллельных процессах

Мячом можно управлять с помощью трёх скриптов, представленных на рис. 5.21. Один длинный скрипт можно разделить на три маленьких. Многим кажется, что так легче понять работу программы. А тебе как кажется?

Первый скрипт гарантирует конец игры в случае касания земли.

Второй скрипт описывает поведение мяча после его касания с ракеткой. Когда мяч касается ракетки, то, согласно команде **касается Ракетка**, начинается выполнение трёх команд, находящихся внутри цикла **всегда, если**.

И наконец, третий скрипт полностью отвечает за полёт мяча.

Все три скрипта начинают работать, когда ты щёлкнешь мышью на кнопке . Все эти три блока команд работают одновременно и параллельно друг другу. Это называется параллельными процессами. Ты можешь сравнить параллельную работу этих трёх скриптов со следующей ситуацией. Три человека вместе плывут в вёсельной лодке. Один гребёт, второй следит за курсом и сообщает о прибытии. А третий занимается тем, что отталкивает лодку, когда она касается берега, причала или чего-нибудь ещё. Три человека в лодке – это одна команда, где каждый отвечает за порученное ему действие.

Рис. 5.21. *Три скрипта для управления мячом*

Проект 13. Устройство для приготовления лимонада

Эта проблема известна каждому. Сколько воды понадобится для приготовления фруктового сока (или сиропа), для того чтобы получить лимонад правильной концентрации да к тому же ещё и вкусный? Решить проблему нам поможет машина для приготовления лимонада. На рис. 5.22 наглядно показана конструкция такого устройства. Коробка с отверстием, в которое помещён стакан с лимонадом. Спереди имеется окошко, затянутое зелёной фольгой, в которую светит фонарик.

Как и в любом высокотехнологичном устройстве, на дисплее нашей машины имеется информация по её использованию. «Пожалуйста, поместите стакан с соком в отверстие». Хорошо, это мы сделали. Устройство зафиксировало стакан, и теперь на экране появляется следующее сообщение: «Сделать напиток более разбавленным? Добавить ...» Здесь тебе нужно ввести цифру. Затем появляется надпись: «Пожалуйста, добавьте в стакан воды». Ты наливаешь воду

в стакан. «Ещё!» – требует машина. В какой-то момент высвечивается сообщение: «Лимонад готов!»

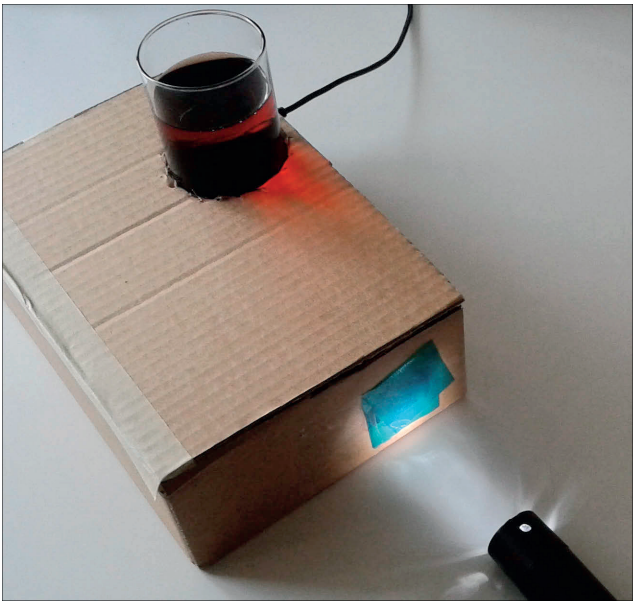


Рис. 5.22. *Так выглядит наше устройство по приготовлению лимонада*

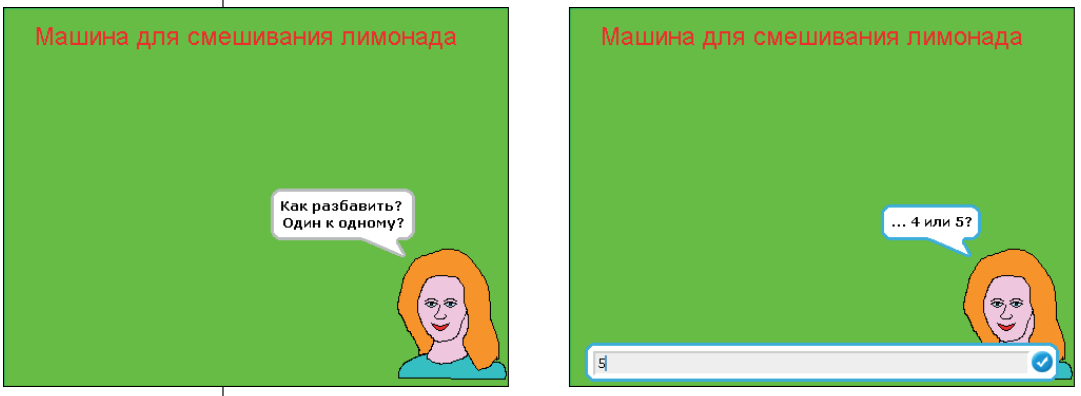


Рис. 5.23. *Девушка-ассистент даёт указания на дисплее*

Данная модель устройства по приготовлению лимонада работает только с напитками красного цвета, т. е. с соками из клубники, вишни, малины или красного винограда. Фонарик светит внутрь коробки через прозрачную зелёную фольгу, которой обтянуто окошко. Зелёный свет проникает через стакан с напитком и воздействует на датчик освещения Picoboard. Для чего нам нужен зелёный свет? Красная

жидкость поглощает зелёный свет и пропускает другие цвета. Поэтому жидкость нам видится красной. Говорят, что зелёный является комплементарным (дополнительным) цветом для красного. Чем сильнее мы разбавляем лимонад, тем бледнее выглядит красный и тем меньше поглощается зелёный цвет. Датчик освещения фиксирует, сколько зелёного света пропустила жидкость, и отправляет программе Scratch значение в диапазоне от 0 до 100. Чем больше эта цифра, тем сильнее разбавлен лимонад.

Устройство

Для изготовления нашей лимонадной чудо-машины тебе понадобятся следующие детали:

- программа Picoboard;
- картон (размером примерно 20×15×8 см) и дополнительно ещё немного бумаги;
- полоска клейкой бумаги;
- универсальный клей;
- алюминиевая фольга;
- зелёная прозрачная фольга (можно зелёную бумагу из-под леденцов);
- фонарик.

На рис. 5.22 показано, как может выглядеть наше устройство. Самым трудоёмким элементом является переключатель внутри коробки. Вырежи из толстой гофрированной бумаги (из картона) полоску размером 16×5 см, согни её и сделай в месте сгиба два разреза около 3 см в ширину. Затем прижми среднюю полоску внутри между этими двумя разрезами, как показано на рис. 5.24.

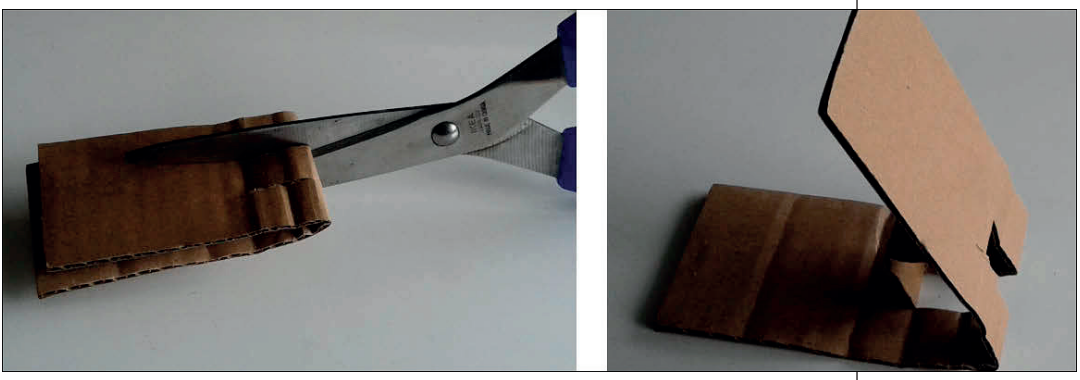


Рис. 5.24. *Полоску бумаги необходимо согнуть и разрезать в месте сгиба*

Теперь вырежи из фольги полоску и наклей её на картон, как показано на рис. 5.25 слева. Сформируй из фольги фигуру, напоминающую гвоздь, но с очень толстой шляпкой (рис. 5.25 слева, A).

С помощью ножниц проткни небольшое отверстие в детали B, в той её части, которая расположена по центру, и вставь туда наш алюминиевый «гвоздь». Согни его за картоном и склей всё кусочками клеящей ленты. Шляпка «гвоздя» станет вторым контактом переключателя. Всё! Теперь переключатель готов!

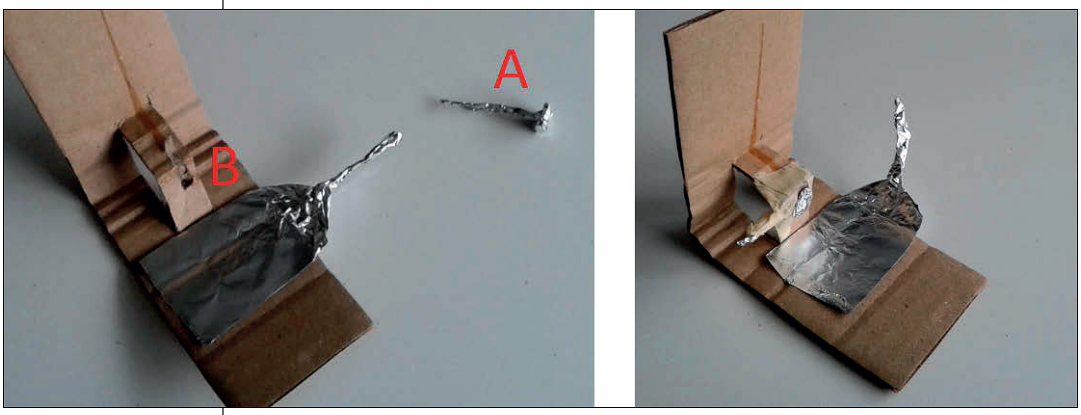


Рис. 5.25. *Так выглядят два наших алюминиевых контакта*

Подключи один из двужильных кабелей с помощью зажимов типа «крокодил» к обоим контактам. Подключи штекер этого двужильного кабеля к разъёму A платы Picoboard. Зафиксируй плату с помощью куска картона и клеящей ленты внутри маленькой коробки. Следи за тем, где расположен датчик освещения (на плате он обозначается нарисованным глазом). Перед датчиком находится самодельный переключатель (рис. 5.26).

В завершение вырежи в крышке коробки одно круглое отверстие для стакана и ещё одно, поменьше, на лицевой стороне коробки точно напротив датчика освещения. Наклей на второе отверстие кусок зелёной фольги. Чтобы усилить действие фильтра, ты можешь наклеить две полоски зелёной фольги друг на друга.

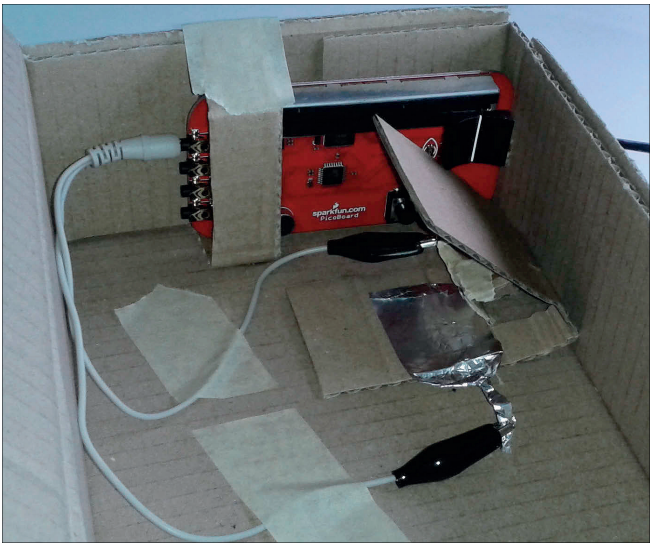


Рис. 5.26. *Расположение Picoboard и переключателя внутри коробки*

Дополнительные (комплементарные) цвета

Как мы уже упоминали ранее, красная жидкость поглощает зелёный свет и пропускает красный. Эти два цвета являются комплементарными, или дополнительными. Встряхни красный напиток в зелёном стакане и посмотри со стороны сквозь бокал. Его содержимое тебе покажется чёрным. Такой эффект получается потому, что сквозь стекло проникает лишь зелёный свет, но красный его полностью поглощает, так что никакого цвета не остаётся и содержимое выглядит тёмным.

Программа

Прежде чем программировать эту процедуру, тебе необходимо выяснить, сколько света падает на датчик освещения при различной концентрации разведённого напитка.

- Смешай в стакане фруктовый сок (или сироп) с водой в соотношении 1:4 и поставь стакан в специально отведённое для него отверстие в коробке.
- Включи фонарик и положи его так, чтобы он светил в предназначенное для света отверстие.

Чтобы проверить работу датчика освещения, нужно выполнить следующие действия.

- Создай новый Scratch-проект и сохрани его, назвав, например, **Лимонад**. Если ты в качестве ассистента не хочешь использовать кота, удали его.

- В коллекции блоков команд **Сенсоры** щёлкни правой кнопкой мыши внизу на командном блоке **значения сенсора** и выбери из появившегося контекстного меню команду **показать Данные ScratchBoard**. На сцене появится поле **выключить** с текущими значениями сенсоров.
- Выпиши значения датчика освещения. Это значение будет в диапазоне от 0 (нет света) до 100 (яркий свет).
- Разведи сок в стакане в пропорции (1:5) и выполни те же измерения, что и для сока в пропорции 1:4. Эти цифры понадобятся при настройке программы.

В простой версии сцена всегда показывает один и тот же фон. Там есть лишь один-единственный объект со своим костюмом, на котором можно видеть лицо ассистентки, объясняющей пользователю, что нужно делать.

- Создай фон для сцены, залив его понравившимся тебе цветом, и сверху сделай надпись **Машина для смешивания лимонада**.
- Создай нового персонажа. Для этого или нарисуй девушку-ассистента, или загрузи из интернета понравившийся тебе рисунок. Можешь вставить фотографию. Здесь раскрывается свобода творчества. Назови этого персонажа **Ассистент**.
- Создай в коллекции команд **Переменные** новую переменную и назови её **Свет**. Сделай эту переменную видимой, увеличь ее размер (щёлкни правой кнопкой мыши и выбери из появившегося контекстного меню строку **велик для чтения**). После чего расположи это поле ввода в левом нижнем углу сцены.
- Собери скрипт для объекта **Ассистент**.

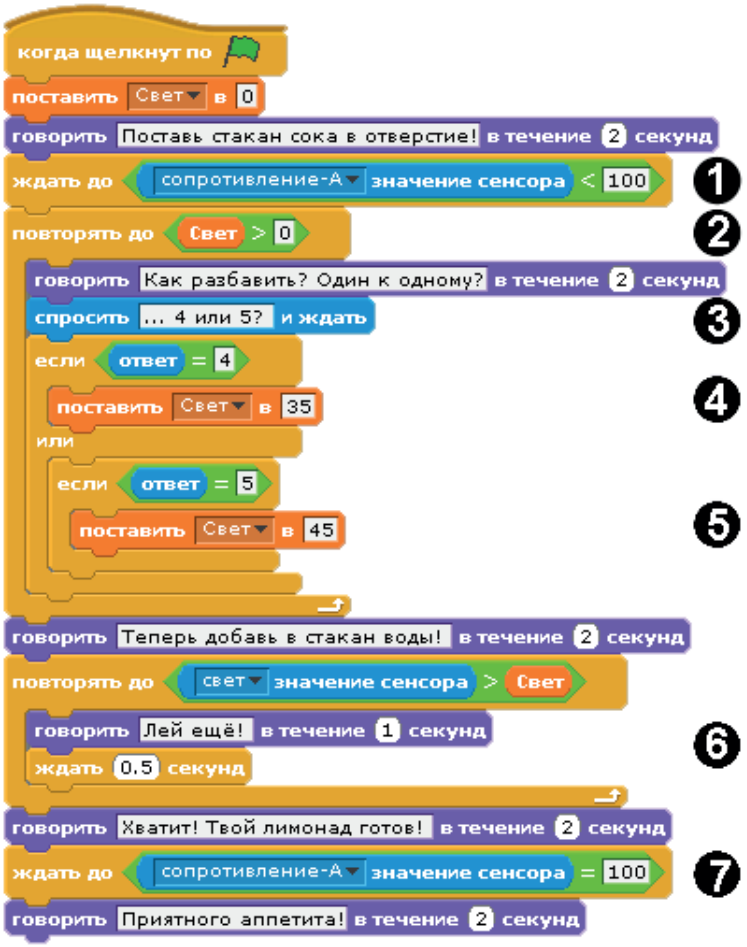


Рис. 5.27. *Скрипт для объекта* **Ассистент**

Пояснения

1. После запуска скрипта компьютер ждёт, когда ты поставишь стакан в предназначенное для него место. Когда стакан будет установлен в гнездо, он своим весом замкнёт контакты датчика **сопротивление – А**, и сопротивление этого контакта станет меньше 100.
2. Цикл будет повторяться до тех пор, пока в поле ввода не будет введена цифра 4 или 5. Только после этого переменная **Свет**, имеющая вначале значение 0, получит новое значение.
3. В нижней части экрана появится поле для внесения ответа, как показано на рис. 5.23.

4. Если будет введена цифра 4, переменная **Свет** получит значение сенсора 35, что соответствует пропорции 1:4.
5. Если пользователь ввёл цифру 5, то переменная **Свет** получает значение сенсора 45, что соответствует пропорции 1:5.
6. Ассистентка на экране произносит фразу «Лей ещё» до тех пор, пока текущее значение датчика освещения не станет больше значения переменной **Свет**.
7. Программа ждёт, пока стакан не будет извлечён из своего гнезда (когда стакан будет извлечён, контакты разомкнутся и сопротивление контактов **A** станет равным 100).

Расширения

В программе у ассистентки на экране всегда одно и то же выражение лица. Будет гораздо интереснее, когда во время произнесения фраз у неё будет открываться рот. Для этого тебе понадобится второй, дополнительный костюм.

- Сделай копию первого костюма и измени форму рта.
- Присвой первому костюму имя **Открыт**, второму – **Закрыт**.
- Чтобы при разговоре рот ассистентки открывался, измени скрипт, вставив перед каждым командным блоком **говорить** командный блок **перейти к костюму Открыт**, а после командного блока **говорить** – командный блок **перейти к костюму Закрыт** (рис. 5.28 справа).

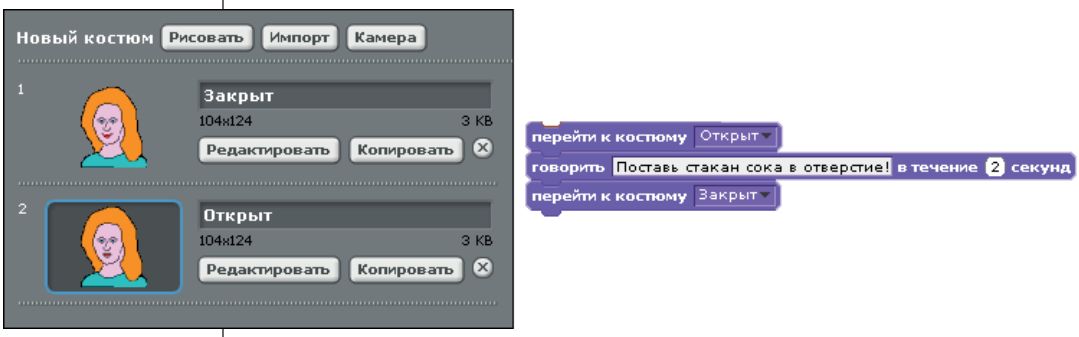


Рис. 5.28. *Создаём мимику рта*

Вопросы

1. Какой диапазон значений яркости и громкости получает компьютер от платы Picoboard?
2. Какая электрическая величина измеряется с помощью двужильного кабеля и штекера?
3. Как отобразить на экране значения всех датчиков платы Picoboard?
4. Какой датчик опрашивается с помощью этого командного блока 

? Какие ещё датчики ты можешь выбрать из этого открывающегося списка? Какие датчики, названные в открывающемся списке, не имеют отношения к плате Picoboard?
5. Что ты будешь делать, если Picoboard подключён, но компьютер датчиков не видит?

Задания. Экспозиметр

Создай экспозиметр, определяющий яркость с помощью стрелки.

Экспозиметр – это приспособление, используемое при фотографировании для измерения яркости или освещенности объекта. В нашем случае датчик освещения будет показывать уровень освещённости самого датчика с помощью отклонения стрелки нарисованного прибора.

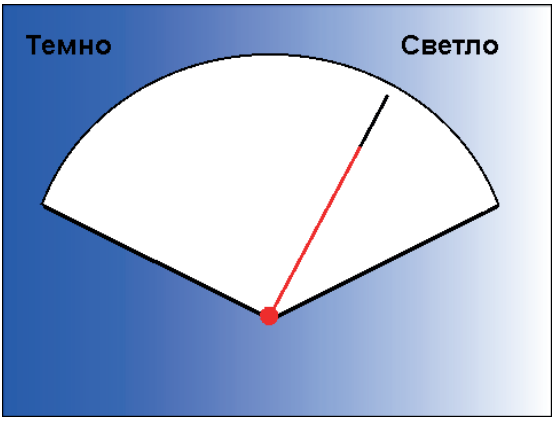


Рис. 5.29. *Так выглядит экспозиметр со стрелкой*

Совет. Сначала нарисуй фон, на котором будет показан прибор (рис. 5.29). Потом создай в графическом редакторе объект – стрелку. Задай центр вращения этой стрелки. Для этого используй в графическом редакторе функцию **Установить центр костюма**. Для создания скрипта для стрелки используй командный блок 

, который находится в коллекции команд **Движение**.

Ответы на вопросы

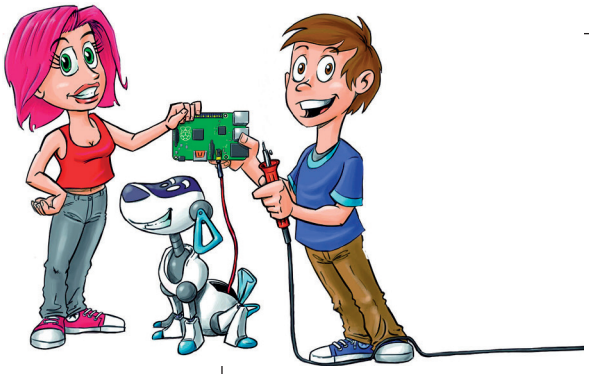
1. Диапазон значений яркости и громкости, которые выдают датчики Picoboard, – от 0 до 100. Это условные единицы измерения. Например, яркость свечения лампы измеряется в *люменах*. Единицей уровня громкости является *децибел*. Но если датчик звука выдаёт 40, это не значит, что уровень громкости равен 40 децибелам. В нашем датчике это будет 40 условных единиц.
2. С помощью двужильного кабеля и штекера измеряется электрическое сопротивление. Заметь, значения, выдаваемые датчиками A, B, C и D, – это не настоящие значения сопротивления подключённых к датчику устройств, а условные единицы в диапазоне от 0 до 100. Сопротивление настоящего электрического проводника (например, провода) измеряется в омах.
3. В правой верхней панели щёлкни мышью на кнопке **Сенсоры**, щёлкни правой кнопкой мыши на командном блоке **значение сенсора** или **сенсор** и выбери из появившегося меню команду **Показать данные ScratchBoard**.
4. Датчики **Наклон** и **Расстояние** не относятся к датчикам платы Picoboard.
5. В коллекции команд **Сенсоры** щёлкни правой кнопкой мыши на командном блоке **значение сенсора** или **сенсор** и выбери из появившегося меню команду **Показать данные ScratchBoard**. На сцене появится панель, в которой показываются значения датчиков. Щёлкни правой кнопкой мыши на этой панели и выбери из появившегося контекстного меню команду **Выбрать серийный/usb порт** (рис. 5.4).

Ответы на задания

Объект, представляющий собой стрелку с центром фигуры в нижней части. Скрипт для объекта **Стрелка** показан на рис. 5.30.

Рис. 5.30. *Скрипт для управления стрелкой*

Совет. Когда будешь рисовать стрелку, направь её вверх. В этом случае в нулевом положении стрелка будет отклонена влево.



6
Интерактивные игры и симуляторы

Если у тебя есть плата Picoboard, значит, ты можешь создавать интерактивные игры и симуляторы с самодельным управлением. К примеру, подбодрить утку громким возгласом, который заставит её летать выше и преодолевать препятствия. С помощью картона, жестяной банки или карандашного рисунка ты сконструируешь блок ввода для игры «Убей комара». С помощью мухобойки, изготовленной из фольги, нужно постараться поймать насекомое. Кроме того, ты узнаешь, как построить автомобильный симулятор, имеющий руль и педаль газа.

Для работы со всеми проектами тебе понадобится плата Picoboard. Но если даже у тебя её нет, большую часть проектов из этой главы ты всё равно сможешь создать. Просто вместо датчиков Picoboard тебе понадобится компьютерная клавиатура. В этой главе ты найдёшь советы, как с ней работать.

Проект 14. «Помоги утке!»

Утка (точнее сказать, селезень – это самец утки) летит над местностью, а её крылья движутся по траектории вверх и вниз. Птица должна следить за тем, чтобы не столкнуться с облаками или деревьями. К сожалению, утка уже устала, и её то и дело нужно подбадривать. Если окрик будет громким, птица взмывает ввысь. Если окрик будет тихим, силы утку покидают, и она опускается вниз. С помощью громкого звука можно помочь утке безопасно пересечь местность. Если же птица с чем-нибудь столкнётся, она крякнет «Ой!» и рухнет без чувств. В этом случае игра считается оконченной.

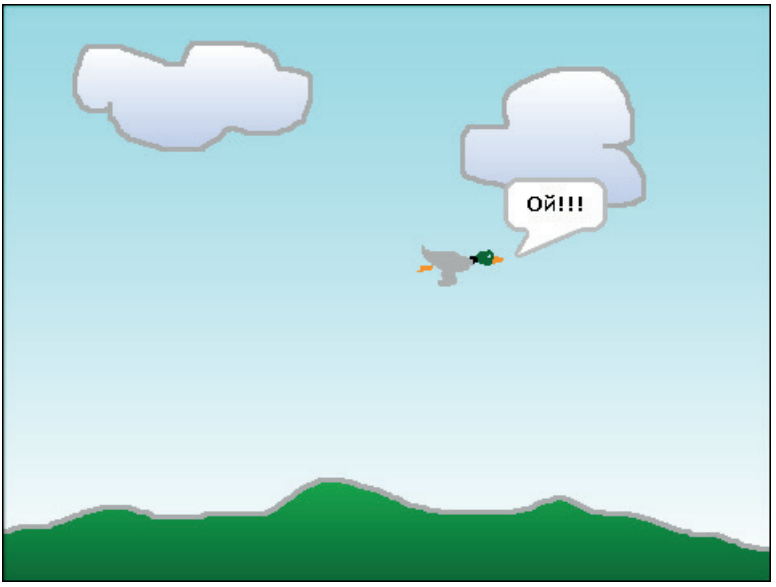


Рис. 6.1. *Конец игры – это столкновение утки с облаком, деревом, домиком, горой или землёй*

Утка переходит на следующий фон, если без столкновений с облаками и деревьями или землёй достигает правой стороны экрана.

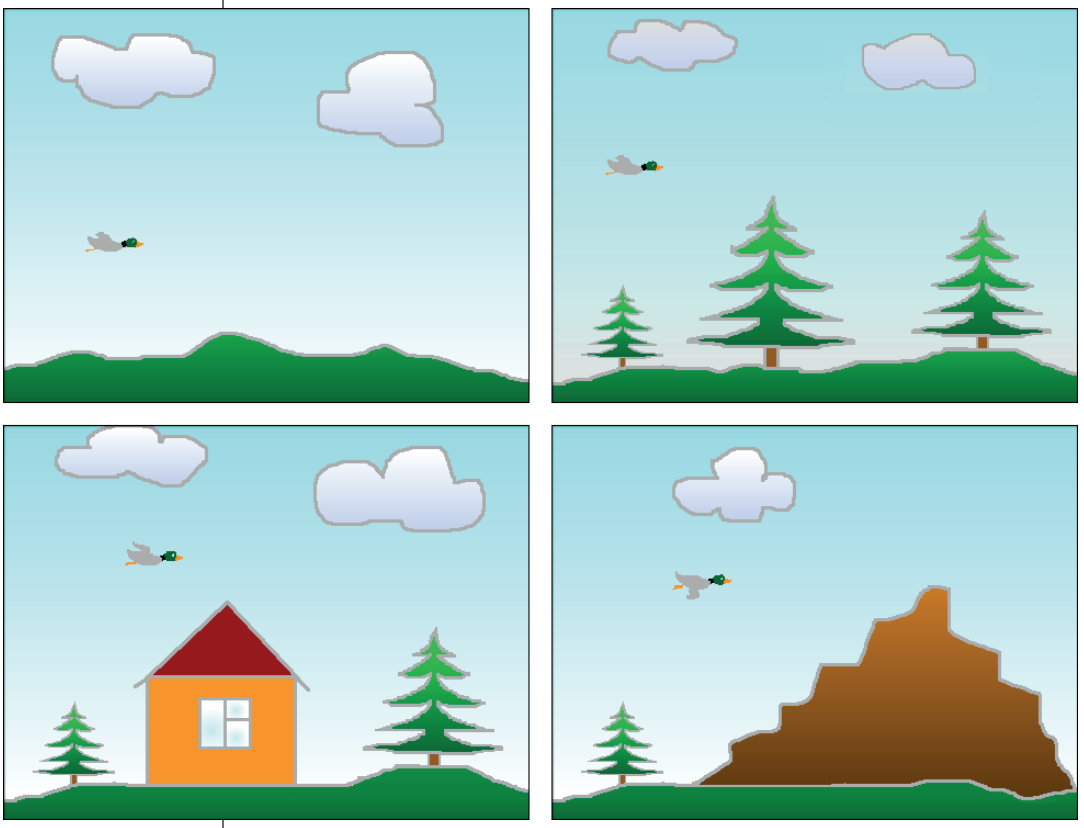
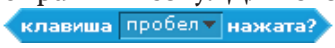


Рис. 6.2. *Фоны меняются, когда утка достигнет правого края экрана*

Без использования Picoboard

Если у тебя нет платы с датчиками Picoboard, то управлять летящими утками ты можешь с помощью клавиши **Пробел**. При нажатии этой клавиши утка будет набирать высоту. Для этого используется командный блок 

.

Сцена

Для сцены потребуется несколько фоновых картинок. Для того чтобы летящая утка легко могла распознать препятствия, воспользуйся следующим советом. Облака, деревья, горы и что ещё там будет встречаться на пути очерти серым цветом. Это всегда должен быть один и тот же оттенок. Потом ты сможешь проверить, касается ли объект **Утка** этого цвета.

Сцене также понадобятся два скрипта. Первый скрипт показывает фон в начале игры, а второй скрипт меняет картинку. Конечно же, сообщение **Следующий фон** тебе придётся создать, потому что пока этого сообщения нет.

- Создай новый проект и удали кота.
- Сохрани этот проект под именем, например, **Утка**.
- Нарисуй четыре фоновые картинки примерно как на рис. 6.2.

Конечно, ты можешь обойтись и двумя картинками, а можешь нарисовать шесть и больше. Чем больше картинок, тем игра интереснее.

- Открой вкладку **скрипты**.
- Собери скрипт как на рис. 6.3.

Обрати внимание: тебе в командном блоке **когда я получу** (имеется в виду, что команда получит сообщение) нужно выбрать строку **Следующее сообщение**. Но, поскольку этого сообщения ещё нет, сделай так: щёлкни мышью на маленьком чёрном треугольнике в поле открывающегося списка. Правее указателя мыши появится команда **Новый**. Щёлкни мышью на этой команде, введи в поле появившегося на экране диалога **Название сообщения «Следующий фон»** и нажми кнопку **ОК**. Это сообщение будет посылать скрипт объекта **Утка**.

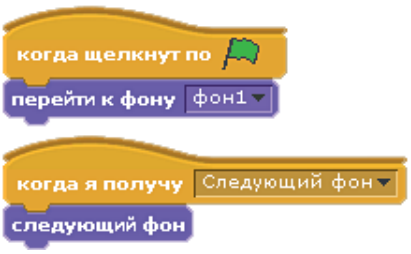


Рис. 6.3. *Скрипты для сцены*

Утка

Тебе понадобится объект, изображающий утку. У этого объекта есть несколько костюмов, представляющих различные фазы движения. Некоторые из них могут выглядеть очень похоже.

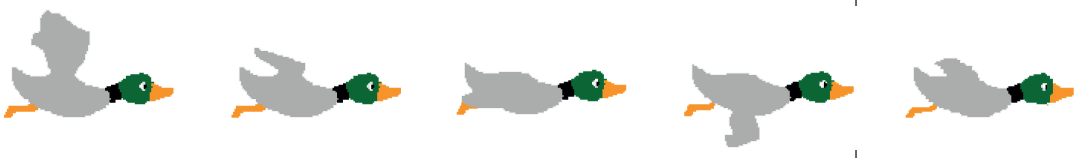


Рис. 6.4. *Фазы движений летящей утки*

При работе скрипта изображения будут поочерёдно сменяться и воспроизводиться как фильм.

- Нарисуй пять костюмов летящей утки как на рис. 6.4.

Для этого в верхней части правой нижней панели нажми кнопку , нарисуй первый костюм утки. Сохрани его. На вкладке **костюмы** нажми кнопку **Копировать**. Будет сделана копия первого костюма. Нажми кнопку **Редактировать** и измени костюм утки. И так для всех пяти костюмов.

- Переименуй объект, введя название **Утка** в самое верхнее поле ввода средней панели.

Следующим этапом нужно собрать скрипты. У нас есть пять скриптов, которые после нажатия на кнопку запускаются одновременно и работают параллельно. Все вместе они управляют полётом утки. Каждый скрипт выполняет свою определённую задачу.

Собери скрипты, как показано на рис. 6.5.

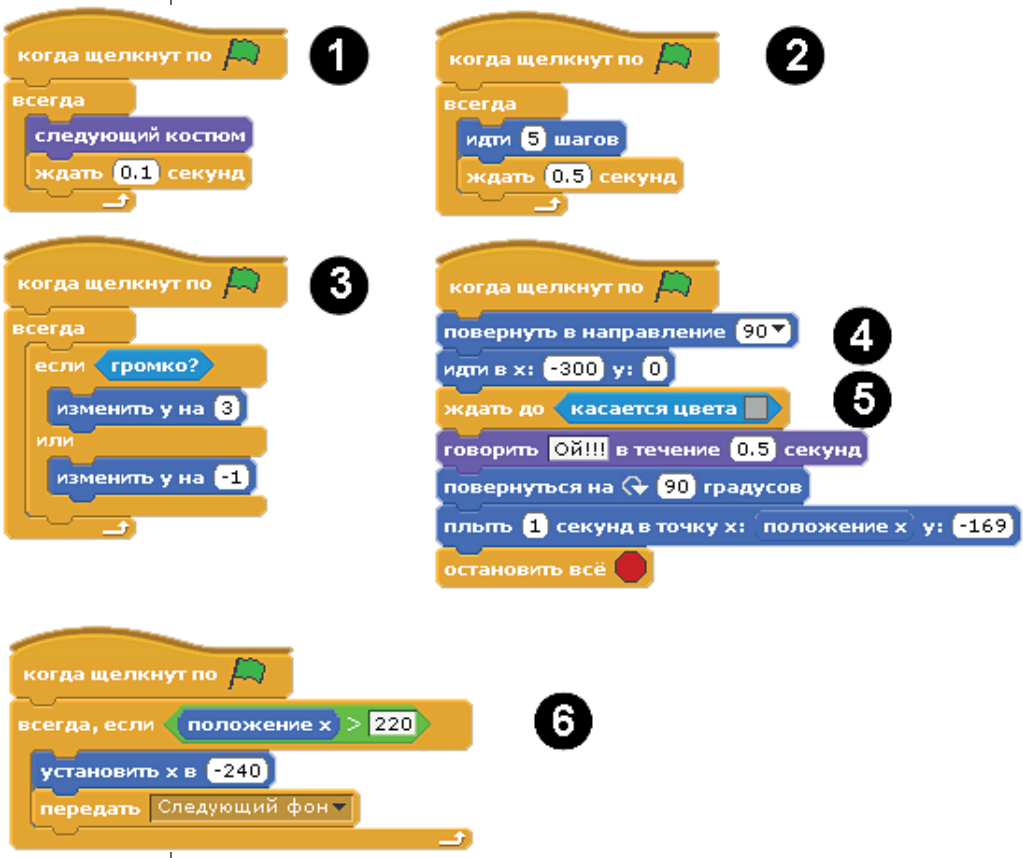


Рис. 6.5. *Скрипты для управления уткой*

Как это работает

1. Фазы движения утки, когда утка машет крыльями, повторяются по циклу. Время смены костюма – 0,1 с.
2. Каждые 0,5 с утка продвигается вперёд на 5 шагов. Если ты хочешь, чтобы утка летела быстрее, введи другое значение.
3. «Услышав» громкий звук, утка поднимается ввысь на 3 шага. Если звука нет, опускается вниз на 1 шаг.
4. Команды **повернуть в направление 900** разворачивает утку после падения в горизонтальное положение, а команда **идти в x** устанавливает утку в начальную позицию.
5. Эта команда ожидает, когда утка коснётся выбранного образца цвета. Этим цветом обведены все контуры облаков, деревьев, домика, горы и земли. После столкновения с препятствием утка удивляется и вертикально пикирует вниз. Игра окончена.
6. Этот скрипт контролирует, когда утка достигнет правого края экрана. Если это произошло, выполняется смена фона на следующий, а птица вновь оказывается у левого края. Смена фона происходит после получения сообщения **Следующий фон**.

Если у тебя нет платы Picoboard, замени командный блок **громко?** на командный блок 

.

Проект 15. «Поймай комара»

На экране ты видишь четыре печенья. Периодически появляется комар, который на несколько мгновений садится на сладкое, а затем вновь исчезает. Тебе необходимо в течение 30 с убить с помощью мухобойки как можно больше насекомых. Перед монитором лежит кусок бумаги или картона с четырьмя контактными площадками, изготовленными из проводящего ток материала, например фольги или четырёх банок из жести.

Сделай мухобойку из палочки или шеста, изготовленного из фольги. Касаясь мухобойкой контактной площадки из проводящей электричество поверхности, обозначающей печенье, ты ловишь комара, севшего на печенье, связанное с этой контактной площадкой (рис. 6.7). Если тебе удастся попасть в нужную контактную площадку, пока комар ещё там, то фоновая картинка ненадолго окрасится в красный

цвет, а значение количества обезвреженных комаров в поле на сцене увеличится на единицу.

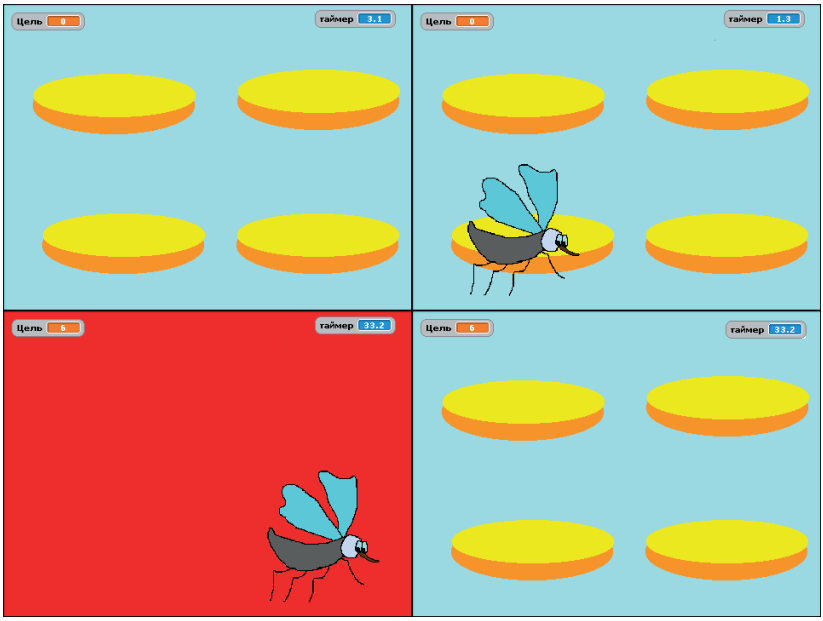


Рис. 6.6. *Комар садится на печенье в левом нижнем углу*

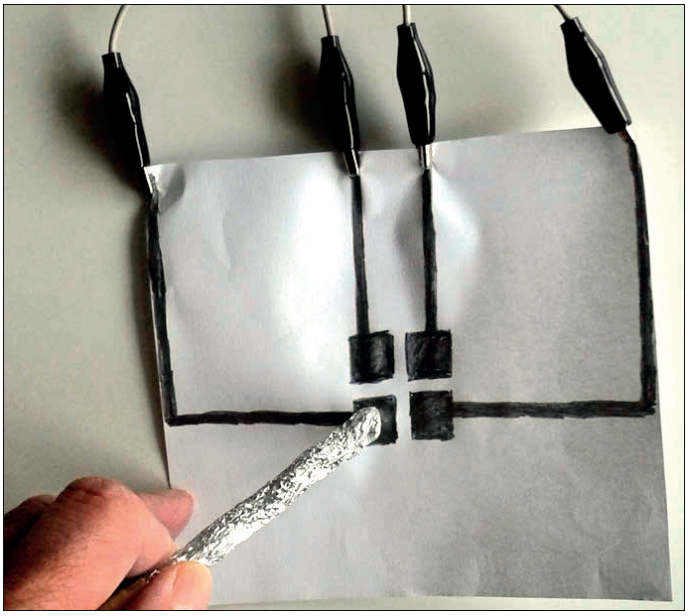
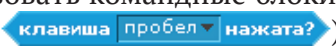


Рис. 6.7. *Мухобойка из фольги прижата к расположенной в нижнем левом углу контактной площадке, обозначающей печенье*

Без использования Picoboard

В отсутствие программы Picoboard ты также можешь ловить комаров с помощью клавиатуры. Каждое изображение лакомства (например, A, B, C, D) будет связано с определённой клавишей. Правое верхнее печенье связано с клавишей **A,** правое нижнее изображение – с клавишей **B** и т. д. Скрипт показан на рис. 6.12 и 6.13 и описан в разделе «Программное обеспечение». Главное отличие от приведённого скрипта в том, что вместо командных блоков, прослушивающих сенсоры (например 

), следует использовать командные блоки, следящие за состоянием клавиш (

).

Аппаратное обеспечение

А ты знал о том, что линия, нарисованная простым графитовым карандашом, проводит электрический ток? Нет? А ты проверь! Нарисуй простым карандашом толстую линию и с помощью двужильного провода, подключённого к разъёму **A** платы Picoboard, измерь сопротивление. Для этого щёлкни правой кнопкой мыши на командном блоке **значение сенсора** или **сенсор** (коллекция команд **Сенсоры**), выбери из появившегося контекстного меню команду **Показать данные ScratchBoard** и прикоснись контактами подключённого к плате двужильного провода концов нарисованной линии. В поле **A** ты увидишь условное значение измеряемого сопротивления, отличное как от 0, так и от 100. Значит, нарисованная линия (дорожка) проводит электрический ток. Это объясняется тем, что грифель карандаша состоит из графита. А графит проводит электричество не так хорошо, как, скажем, металл. Но, как и всякий проводник, нарисованная дорожка имеет сопротивление (сопротивление движению электрического тока). Если сопротивление проводника очень велико, значит, это диэлектрик. Если мало – проводник. Графит имеет не очень большое сопротивление и является проводником. Поэтому нарисованные таким карандашом дорожки (небольшой длины и достаточно широкие) также будут проводниками электрического тока, и мы их можем использовать в нашей программе.

Возьми самый мягкий (какой у тебя найдётся) простой карандаш. Да. Карандаши тоже бывают твёрдыми и мягкими. Понятие «Твёрдый» или «Мягкий» не обозначает его физические свойства на ощупь. «Твёрдый» карандаш рисует еле заметно. Им намечаются контуры. А мягкий карандаш даёт

насыщенную линию. Им контуры жирно наводятся. О том, что взятый карандаш мягкий, говорит буква М на его корпусе. Самый мягкий карандаш (содержащий наибольшее количество графита) обозначается как 6М. Если твой карандаш изготовлен в Европе, о его мягкости говорит буква В. Самый мягкий карандаш обозначается 9В.

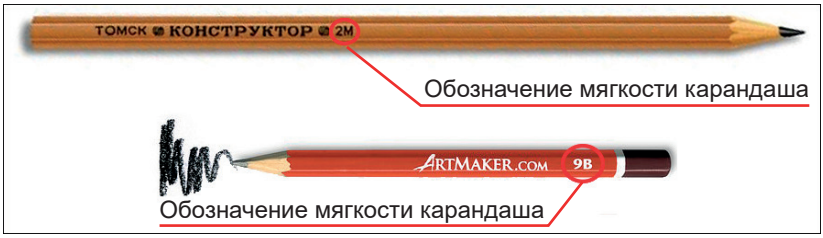


Рис. 6.8. *Обозначение мягкости карандашей*

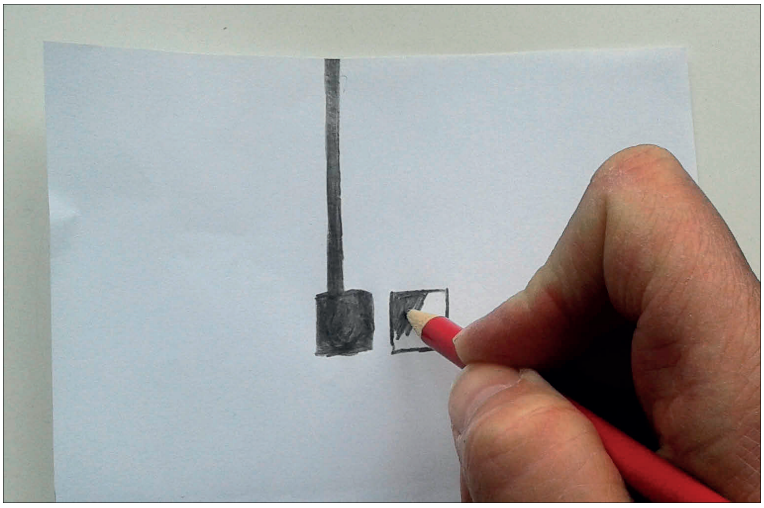


Рис. 6.9. *Контактные площадки и дорожки-проводники рисуются мягким простым карандашом*

- Нарисуй на бумаге четыре поля с токопроводящими дорожками, ведущими к краю (рис. 6.9).
- Сделай из фольги «мухобойку» и подключи к ней один конец каждого двужильного кабеля (рис. 6.10). Вторые концы кабеля с помощью зажима типа «крокодил» подключи к нарисованным дорожкам.

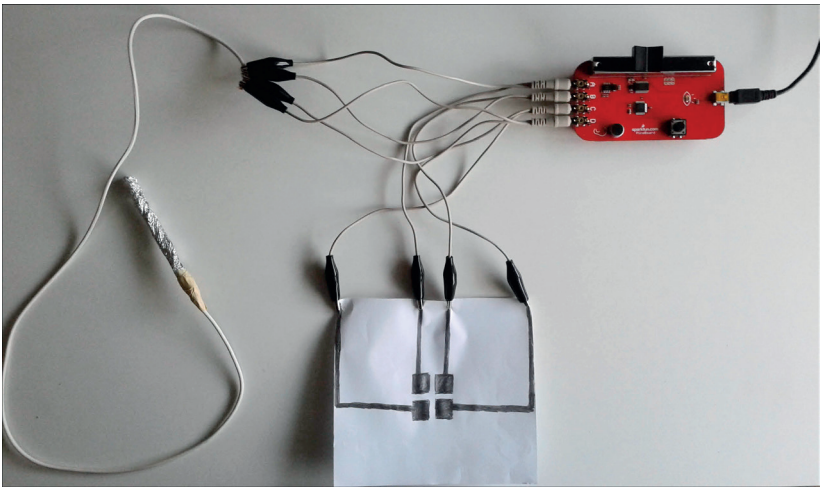


Рис. 6.10. *Так выглядит готовая конструкция. При подключении зажимов «крокодил» соблюдай полярность*

Обрати внимание: при подключении зажимов «крокодил» тебе необходимо соблюдать полярность. Для этого следуй нижеприведённой инструкции.

- Убедись, что Picoboard подключён к Raspberry Pi.
- Запусти программу Scratch.
- Щёлкни правой кнопкой мыши на командном блоке **значение сенсора** или **сенсор** (коллекция команд **Сенсоры**) и выбери из появившегося контекстного меню команду **Показать данные ScratchBoard**.
- Подключи двужильный кабель ко входу **A** в Scratchboard. Один из зажимов присоедини к проводу мухобойки, «укусив» его «крокодилом». Второй «крокодил» этого кабеля подключи к графитовой токопроводящей дорожке. Коснись мухобойкой контактной площадки, токопроводящая дорожка которой подключена к плате через разъём «крокодил». В диалоге **ScratchBoard**, в поле данных входа **A**, ты увидишь значение ниже **100**.

Со вторым двойным кабелем ты должен быть более внимательным.

- Подключи второй двойной кабель ко входу **B** платы Picoboard. Коснись одним из двух «крокодилов» контактной площадки, к которой подключен провод от контакта **A** платы Picoboard. Следи за показаниями диалога **ScratchBoard**. Если значения входа **B** (к которому подключён второй двужильный кабель входа **A**) будут равны **0**, значит, полярность кабеля **B** выбрана непра-

вильно. Возьми второй «крокодил» этого кабеля и снова коснись контактной площадки, к которой подключен «крокодил» провода входа **A**. Если значение в поле **B** изменится и станет больше **0**, значит, это правильный контакт. Подключи его с помощью «крокодила» к токопроводящей дорожке второго контактного поля. Второй контакт двужильного провода подключи к проводу мухобойки.

- Таким же образом подключи двужильные провода ко входам **C** и **D**, определи полярность каждого провода и подключи провода к дорожкам контактных площадок **C** и **D**, а другие концы – к мухобойке.

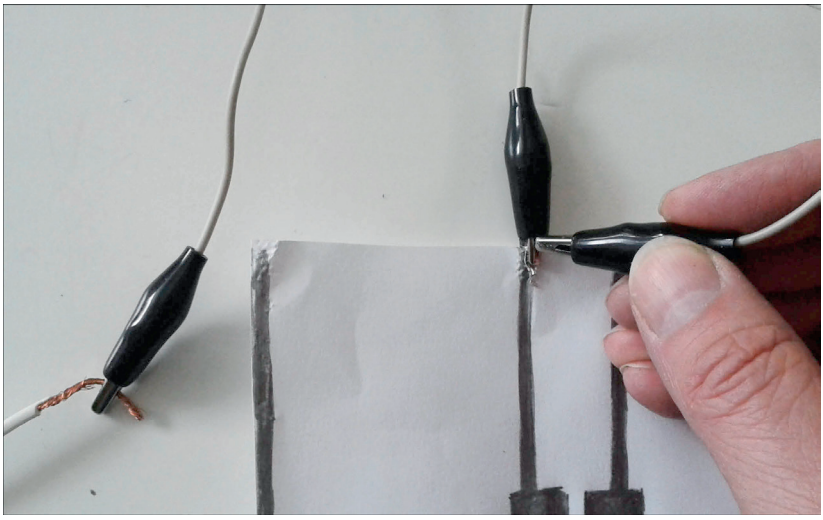


Рис. 6.11. *Перед подключением убедись в отсутствии замыкания с первым зажимом «крокодил»*

Программное обеспечение

Переменные

Создай две переменные. Переменная **Поле** должна содержать номер того поля, на которое садится комар. № 1 означает верхнее левое поле, № 2 – правое верхнее поле, № 3 – левое нижнее, и № 4 – это номер правого нижнего поля. Эта переменная не отображается на сцене. Убедись, что флажок рядом с именем этой переменной сброшен.

Переменная **Цель** должна отображаться на экране. В ней будет показано количество нейтрализованных комаров.

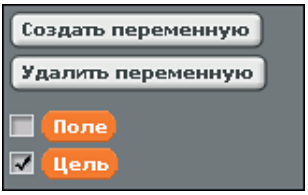


Рис. 6.12. *Переменная* ***Цель*** *отображается, а переменная* ***Поле*** *– нет*

Сцена

Сцена имеет две фоновые картинки.

- Первая фоновая картинка изображает четыре печенья, на которые время от времени садится комар.
- Вторая фоновая картинка полностью залита красным цветом. Она появляется недолго и только тогда, когда комар был нейтрализован.

Приступим к созданию фона сцены.

- Нарисуй два фона как на рис. 6.6. То есть один фон залей светло-серым цветом и нарисуй четыре «печенья». Второй фон просто залей красным цветом.

Теперь нужно собрать скрипты. В общей сложности здесь пять скриптов, четыре из которых очень похожи между собой. Все они запускаются вначале и затем работают параллельно друг другу.

Собери скрипты, как показано на рис. 6.13. У тебя получится пять скриптов. Первый перезапускает таймер, устанавливает значение **Поле**, равное 0, и контролирует время. Когда пройдёт 30 с, игра останавливается. Остальные четыре скрипта контролируют каждый свой контакт и поле (печенье) на сцене. Обрати внимание: четыре скрипта для управления сценой похожи друг на друга. На рис. 6.13 показан первый общий для всех скрипт, скрипт для левого верхнего контакта (на сцене это левое верхнее печенье) и частично показан скрипт для второго контакта (верхнего правого печенья).

Скрипт для верхней левой контактной площадки (и левого верхнего печенья) «караулит» левую верхнюю контактную (графитовую) площадку, которая соотносится с полем в виде печенья в левом верхнем углу. Когда мухобойка касается этого контакта, ток проходит через графит и сопротивление на входе **A** платы Picoboard понижается. Учитывая, что при замыкании контактов значение сопротивления в поле **A** будет равным **100**, при касании графитового поля значение сопротивления будет близко

к **100**. Если комар сел именно на это печенье, то касание мухобойкой именно этого контакта обозначает попадание в цель. Количество попаданий возрастает, что отразится в поле **Цель**, а фоновая картинка на 0,5 с окрасится в красный цвет. Если ты коснулся мухобойкой другой контактной площадки, не той, на которую «сел» комар, попадание в цель не считается.

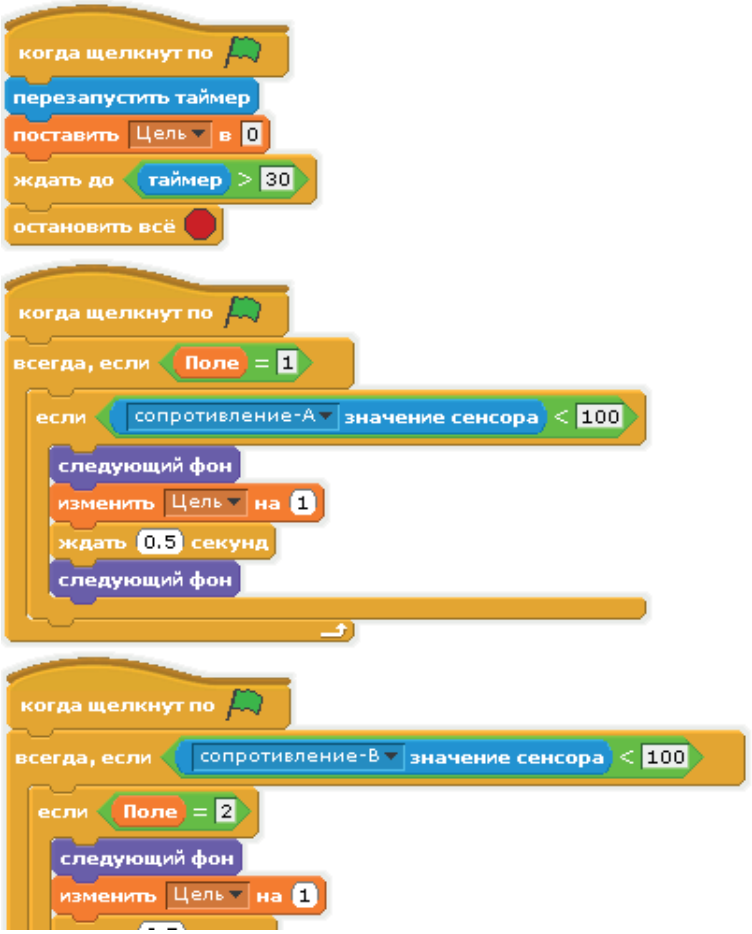


Рис. 6.13. *Скрипты для сцены*

Как уже говорилось ранее, на рис. 6.13 показан только один скрипт для первого поля, расположенного в левом верхнем углу. Три скрипта для второго, третьего и четвёртого полей (печенья) абсолютно похожи на скрипт для первого поля. Отличаются скрипты только номером поля и выбранными контактами на плате Picoboard. То есть для поля в правом верхнем углу (печенья) следует ввести значение **поставить**

Поле = 2, а **значение сенсора – сопротивление B**. Ещё раз обрати внимание: для каждого поля ввода собран свой скрипт, который наблюдает за сопротивлением своего контакта и запускает действие, если это сопротивление снижается до 100. Ты можешь скрипт для первого поля три раза дублировать, после чего ввести нужные значения.

Комар

Нарисуй объект, который будет изображать комара, как на рис. 6.6. У этого объекта будет всего один-единственный костюм и один скрипт. Хоть этот скрипт и кажется длинным, собирается он легко. Обрати внимание, это важно. Печенье, на которое должен «сесть» комар, выбирается случайным образом.

Рис. 6.14. *Скрипт для управления комаром*

Как это работает

1. Комар появляется неожиданно. Сначала его не видно, но после небольшого ожидания длительностью до трёх секунд он становится видимым. Но где же он появится?
2. У нас есть четыре печенья, которым мы присвоили цифры от 1 до 4. Номер каждого печенья выбран случайно и сохраняется в переменной **Поле**. Комар появляется на сцене над печеньем, номер которого выбирается случайным образом.
3. Время, которое проведёт насекомое на выбранном печенье, также выбирается случайным образом в диапазоне от 0,2 до 1 с.
4. Если значение **Поля** равно 0, значит, комар исчез и его нет ни на одном печенье.

Теперь сделаем так, чтобы игрой можно было управлять, не только касаясь «мухобойкой» контактной площадки, но и нажимая одну из четырёх выбранных клавиш. Для этого открой скрипт, управляющий фоном, и внеси изменения как на рис. 6.15. В левой части ты увидишь скрипт до внесения изменений, справа – после изменений. В нижней части рис. 6.15 показано, как собрать изменение в скрипте. Мы используем командный блок **или**, в левую часть которого вставляем командный блок **клавиша … нажата?**, а в правую часть – командный блок <, в который вставлен командный блок **значение сенсора**.

Контроллер MaKey MaKey

Если тебе нравится мастерить оригинальные коммутаторы из сподручных материалов, то, возможно, тебя заинтересует проект MaKey MaKey (скачать можно по ссылке http://www.makey-makey.com/). Это плата, похожая на Picoboard, которая подключается через USB-разъём. Его входы чувствительны к очень низкому напряжению. Таким образом, ты можешь сконструировать тачпад и выключатель, реагирующий на касания. Стоимость такой игрушки в России порядка 4250 рублей.

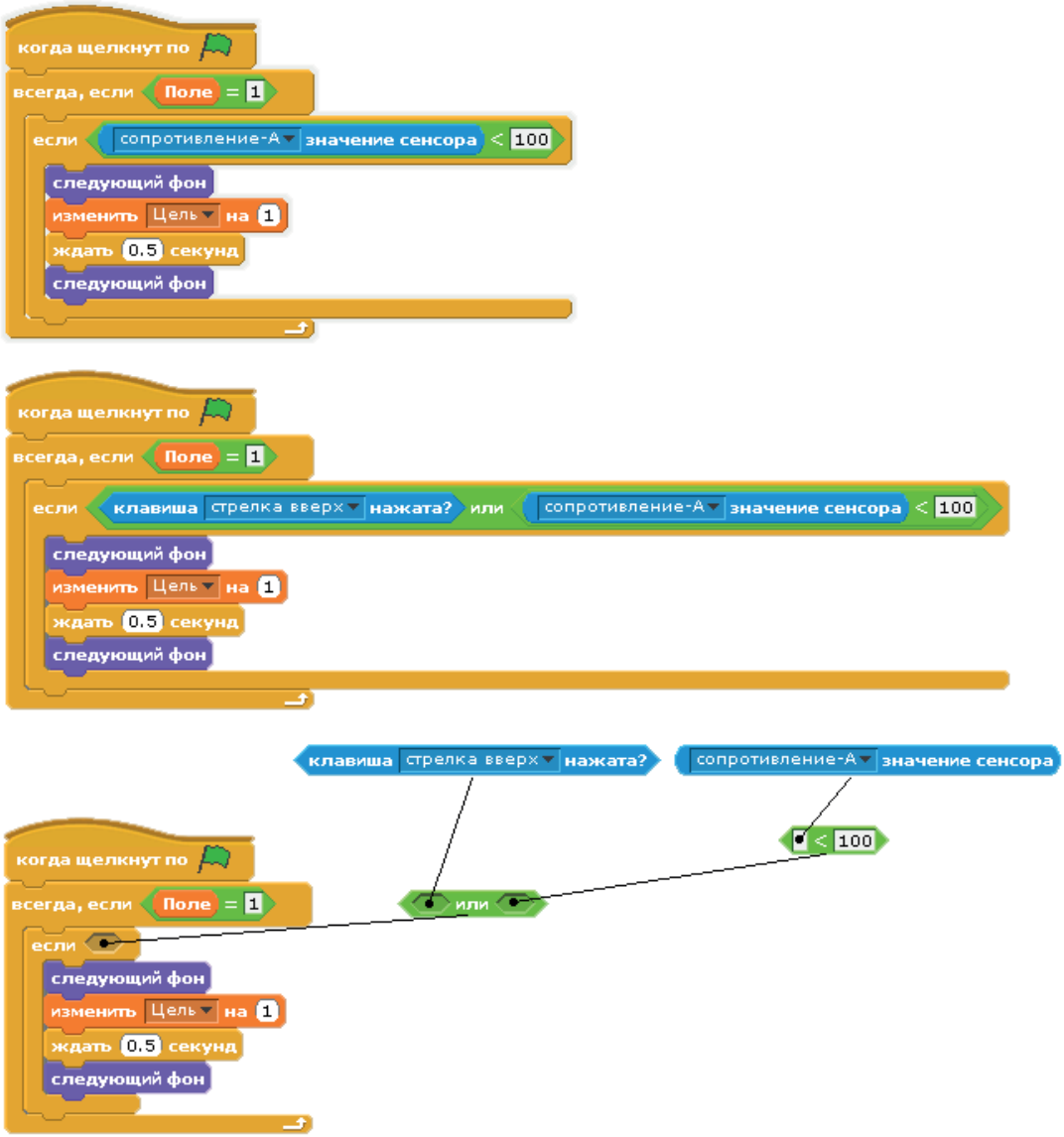


Рис. 6.15. *Изменение в скрипте, чтобы можно было управлять игрой не только с помощью мухобойки, но и клавишами*

Проект 16. Формула 1

Во второй главе мы уже занимались проектом автогонки, где наши болиды управлялись с помощью клавиш со стрелками. В этом разделе мы продолжим начатый проект. Мы построим полноценную приборную панель с рулём управ-

ления и педалью газа. Таким образом, автомобиль получит управление, почти как на профессиональном тренажёре.

Аппаратное обеспечение

Для создания гоночного проекта тебе понадобятся следующие материалы:

- одна маленькая картонная коробка и толстый кусок гофрированной бумаги;
- полоска клеящейся бумаги и клей;
- алюминиевая фольга;
- кабель;
- потенциометр (резистор) с сопротивлением в 10 кОм.

Создаём педаль газа

Из толстой бумаги сделай переключатель, в точности как при создании машины для смешивания лимонада, только более крупного размера и из толстой бумаги. К обоим контактам подключи двухжильный кабель (примерно 1,20 м длиной).

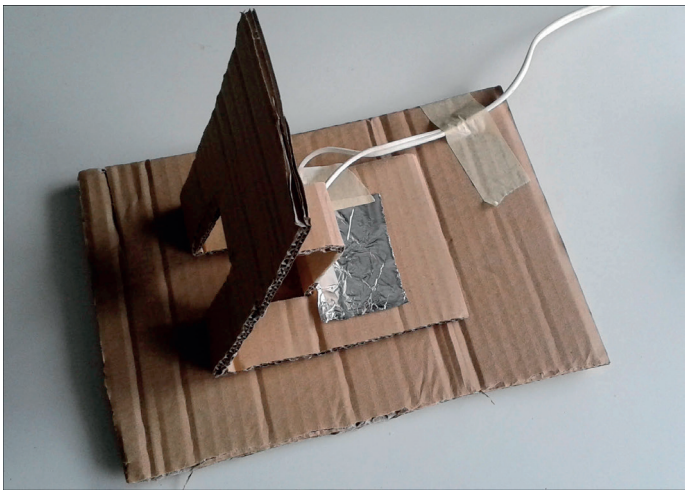


Рис. 6.16. *Строим педаль газа из бумаги и фольги*

Руль управления

Для создания руля ты будешь использовать потенциометр. Это деталь, дорожка которой состоит из нанесённого резистивного материала. *Resist* в переводе с английского – сопротивление. Дорожка сделана в виде дуги, в центре которой установлена ось, на которой закреплён скользящий

контакт. К концам покрытой резистивным слоем дорожки присоединены электрические контакты. Третий контакт скользящий. Перемещаясь по дорожке с резистивным покрытием, мы изменяем сопротивление между двумя неподвижными контактами и подвижным. Обрати внимание, как уже было сказано ранее, подвижный контакт соединен с осью.

Значение сопротивления потенциометра – это сопротивление дорожки с резистивным контактом. Тебе понадобится потенциометр сопротивлением в 10 кОм. Лучше всего Picoboard обрабатывает значение сопротивления от 0 до 10 кОм. Важен следующий момент. Сопротивление между средним и крайними контактами изменяется и регулируется путём вращения вала потенциометра. Позже на этот вал мы установим руль. Вращая этот руль по часовой или против часовой стрелки, ты будешь менять сопротивление между подвижным и неподвижными контактами.

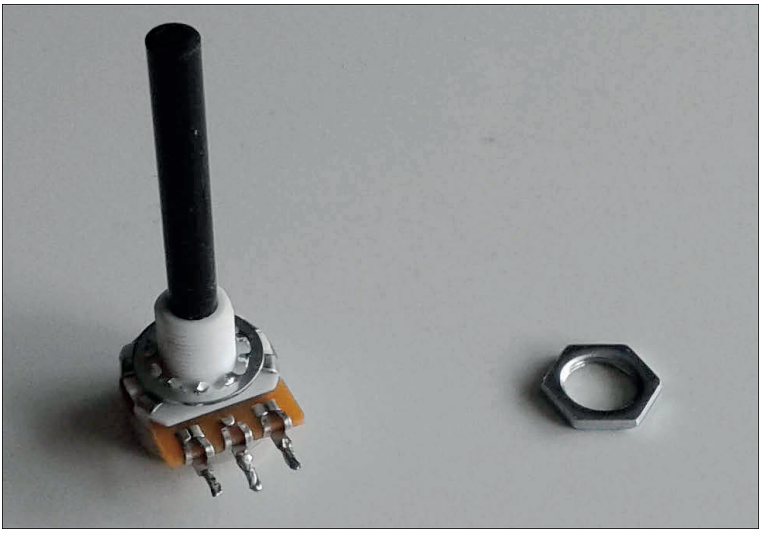


Рис. 6.17. *Так выглядит потенциометр с гайкой для крепления*

- Открути крепёжную гайку от потенциометра, вставь её в отверстие картонной коробки и закрепи потенциометр гайкой.
- В коробку помести Picoboard.
- Соедини вход **A** с внешним и средним контактами потенциометра (см. рис. 6.18).
- Открытый конец кабеля педали газа протяни через отверстие в коробке. Подключи вход **B** к двум проводам кабеля.

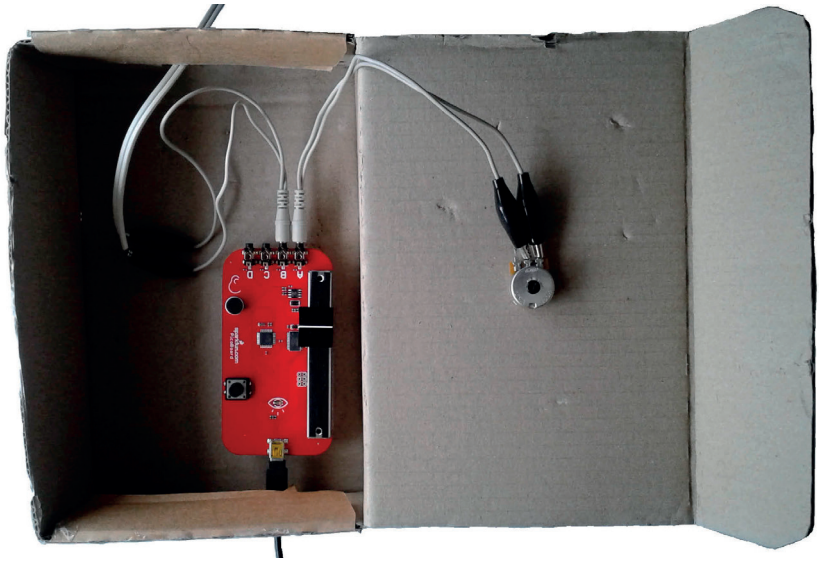


Рис. 6.18. *Консоль гоночного болида изготовлена из картонной коробки*

- К валу потенциометра приклей немного клеящей ленты, для того чтобы лучше закрепить следующий блок бумаги.
- Приклей тонкий лист бумаги (размером 2×10 см) к валу и согни его наружу (см. рис. 6.19, слева).
- На противоположную сторону приклей вторую полоску бумаги (см. рис. 6.19, в центре).
- Вырежи из толстой бумаги руль болида Формулы 1 и приклей его к бумажным полоскам вала. Кстати, бумажный руль своей формой напоминает серебряную стрелку мерседеса. Следи за тем, чтобы поворотный вал потенциометра находился в центральной позиции, когда руль расположен горизонтально (прямая позиция).

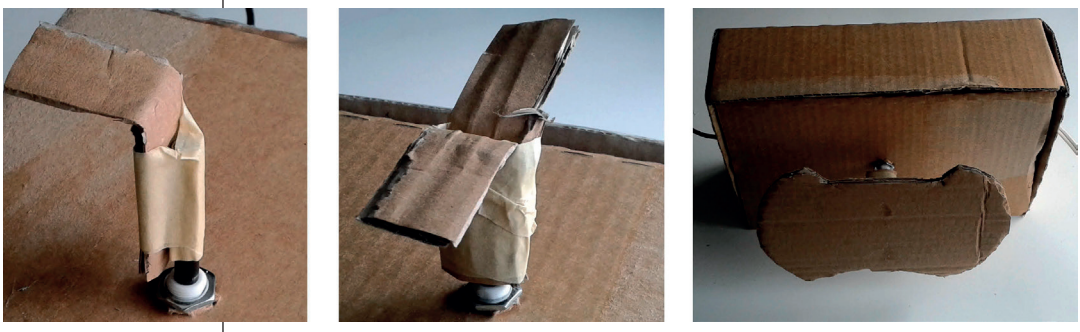


Рис. 6.19. *Установка руля управления*

Программа

Открой свой старый гоночный проект и сохрани его под новым именем. Затем выбери команду **Файл ⇨ Сохранить как**. Появится диалоговое окно с надписью **Сохранить проект**. Внизу в поле рядом с **Имя файла** внеси новое имя файла.

Ты можешь использовать все рисунки старого проекта. Изменяется только скрипт гоночного болида или, что лучше всего, собирается заново. Обрати внимание, если ты используешь проект с двумя фоновыми изображениями, на которых нарисована трасса, скрипт проекта нужно только доработать. Скрипты, в которых описано управление направлением и скоростью с помощью клавиш со стрелками, нужно оставить. Дорабатывается скрипт (лучше собрать новый скрипт), управляющий автомобилем с помощью педали и руля. В этом случае ты сможешь управлять автомобилем не только с помощью клавиш, но и рулём и педалью. Эти скрипты будут работать параллельно друг другу. Полный скрипт для управления автомобилем показан на рис. 6.22. Скрипт для фона не меняется.

Тестируем руль управления

Когда Picoboard подключён к Raspberry Pi, ты должен сначала проверить, какие значения на входе **A** отображаются при вращении руля. Для этого включи показания Picoboard. В коллекции команд **Сенсоры** щёлкни правой кнопкой мыши на команде **Значение сенсора** и выбери в появившемся меню команду **Показать данные ScratchBoard**.

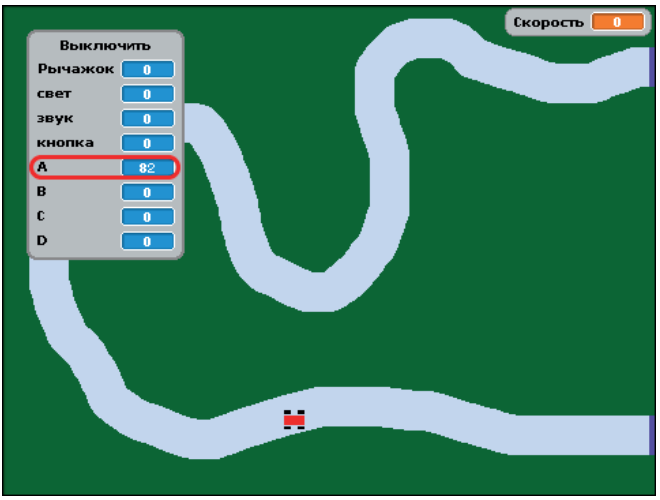


Рис. 6.20. *Так выглядит интерфейс симулятора гонок с показаниями Picoboard*

Прежде всего выясни следующее:

- значение входа **A**, когда руль находится в центральной позиции;
- значение входа **A**, когда руль повёрнут вправо до упора;
- значение входа **A**, когда руль повёрнут влево до упора.

Объект-автомобиль

У нашего автомобиля всего один скрипт, содержащий цикл, в течение которого постоянно опрашиваются входы **A** и **B** в Picoboard.

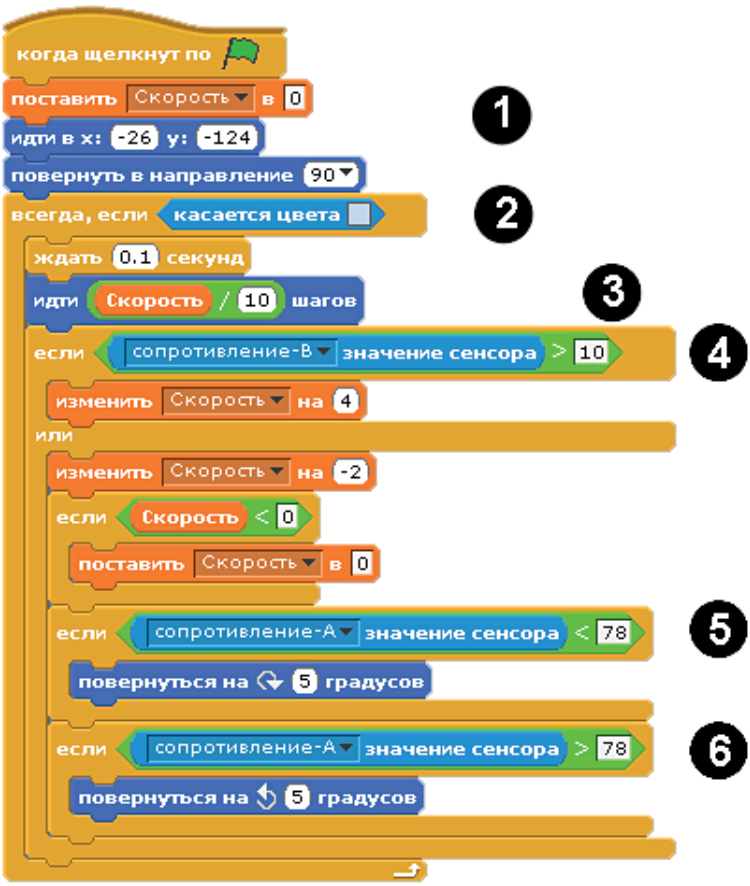


Рис. 6.21. *Скрипт управления гоночным болидом с помощью педали и руля*

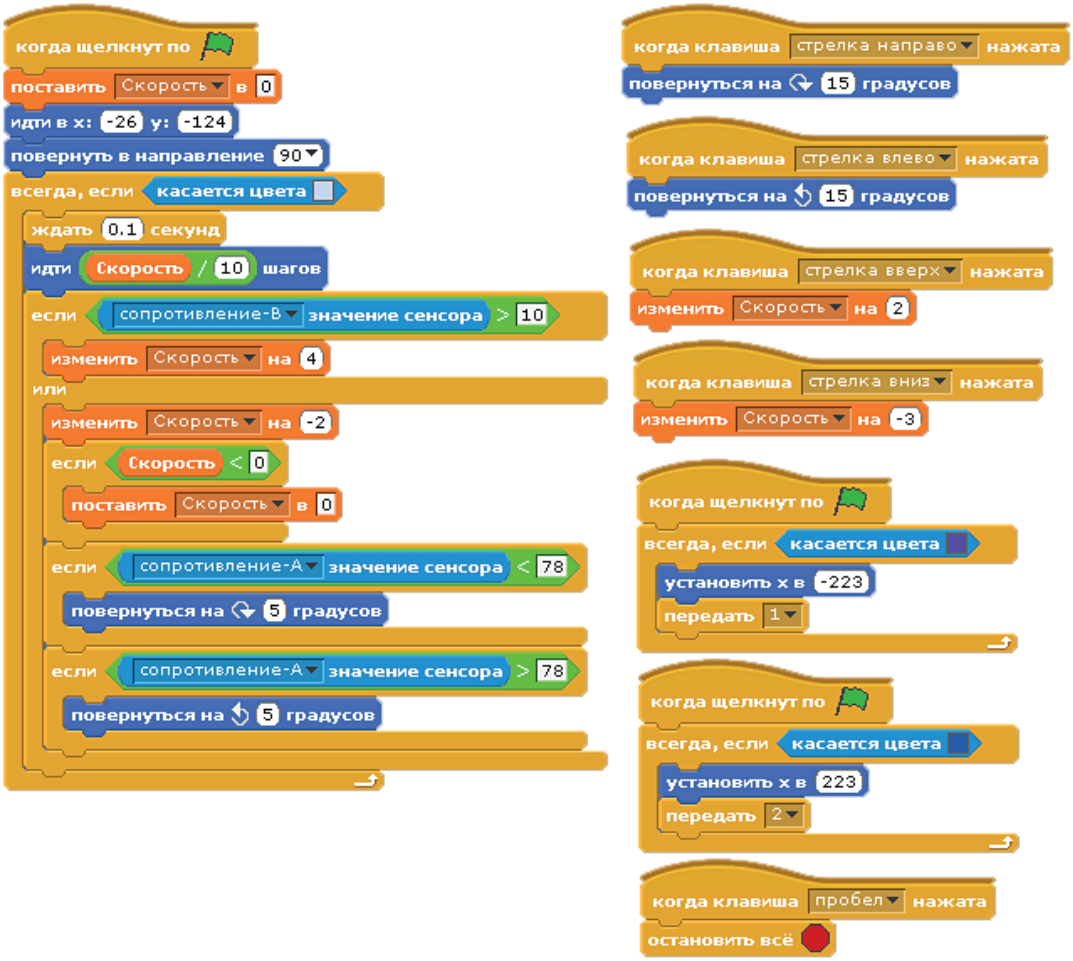


Рис. 6.22. *Полный скрипт для управления автомобилем как с помощью клавиш, так и посредством педали и руля*

Пояснения

1. Сначала мы помещаем автомобиль в исходную позицию. Начальная скорость у него равна нулю.
2. Основной цикл проходит до тех пор, пока машина находится на дороге.
3. Поскольку автомобиль перемещается на 10 шагов за 1 с (согласно инструкции 1 шаг за 0,1 с), то длина его шага равна одной десятой доли скорости (переменная **Скорость**).
4. Когда педаль газа нажата, сопротивление на входе **B** равно нулю, т. е. его значение, исходящее от датчиков платы Picoboard, значительно выше 10 единиц. После

каждого нажатия на педаль скорость повышается на 4 пункта. Если клавиша не нажата, скорость снижается на 2 пункта.

5. В ходе испытания было зафиксировано, что руль при значении сопротивления в 82 единицы находится в горизонтальном положении. Значение увеличивается, когда руль повёрнут вправо. Таким образом, при большом угле поворота руля (если цифра менее 78) автомобиль на экране поворачивается вправо на 5°.
6. Если руль сильно повёрнут влево, машина также поворачивает влево. Когда руль находится в центральной позиции (сопротивление между 78 и 87), автомобиль не сворачивает, а едет прямо.

Вопросы

1. Иногда сцене требуется скрипт. Для чего?
2. С помощью какой команды объект сообщает сцене, что должен смениться фон?
3. Каково напряжение на зажимах типа «крокодил» двойного кабеля при подключении его к входам сопротивления Picoboard (A, B, C или D)?
4. Какая электрическая величина изменяется при вращении вала потенциометра?

Задания. Садимся на Луну

Эта игра является настоящей классикой среди компьютерных приложений. Космический корабль приближается к Луне. Под действием силы тяжести он всегда падает быстрее. Космонавты запускают двигатель, корабль замедляется и, если топливо расходуется экономно, совершает мягкую посадку. Если астронавты запустят двигатель слишком рано или будут запускать его часто, топливо закончится и случится авария.

Корабль перед посадкой каждый раз появляется в новом месте. Желательно посадить его на ровную поверхность. Ты при посадке можешь смещать корабль в горизонтальной плоскости с помощью клавиш ← и →.

Создай симуляцию высадки на Луну. Двигатель будет запускаться с помощью громкого звука или хлопка, а если громких звуков не будет, двигатель заглохнет. Скорость, с которой снижается корабль, и запас топлива должны отображаться на экране.

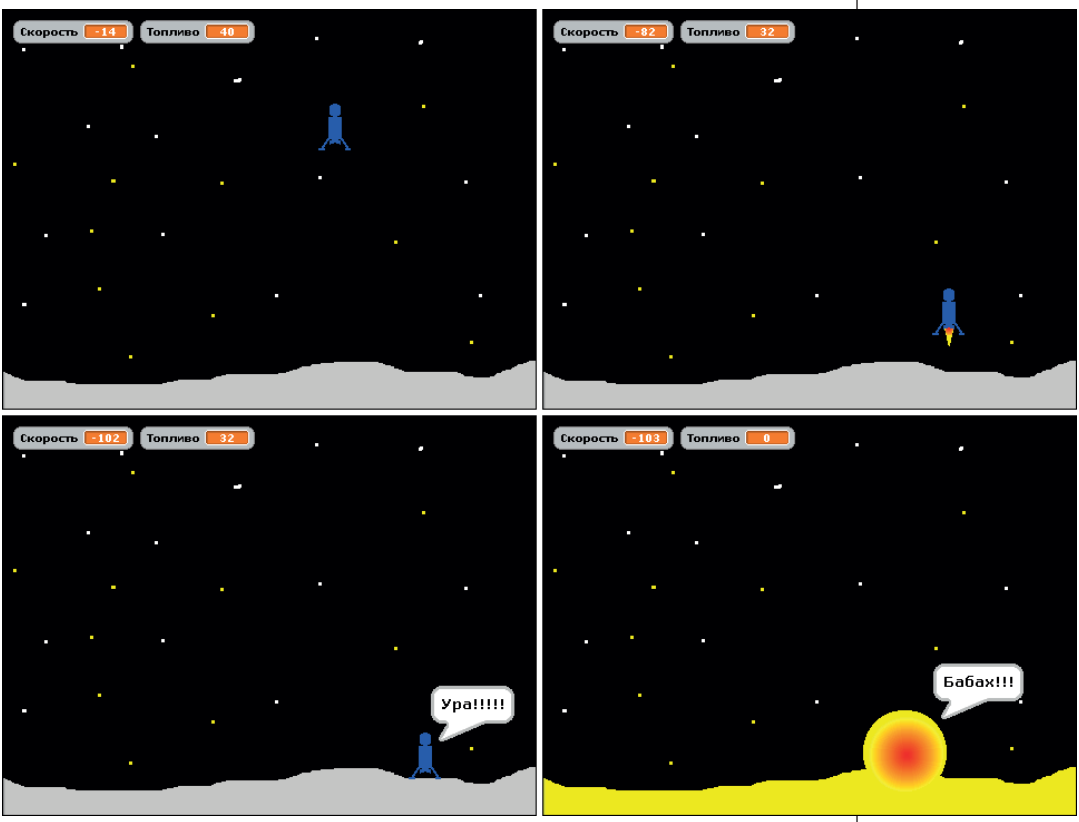


Рис. 6.23. *Безопасная посадка ну Луну, но всё может закончиться иначе*

Советы

Создай две переменные, назвав их **Скорость** и **Двигатель**. Приготовь для корабля объект из трёх костюмов (рис. 6.24).

Нарисуй два фона. На одном фоне поверхность Луны сделай светло-серой, а на другом – жёлтой. Это нужно, чтобы при штатной посадке корабль сказал «Ура!», а при аварийной – «Бабах!», а сам превратился в огненный шар.

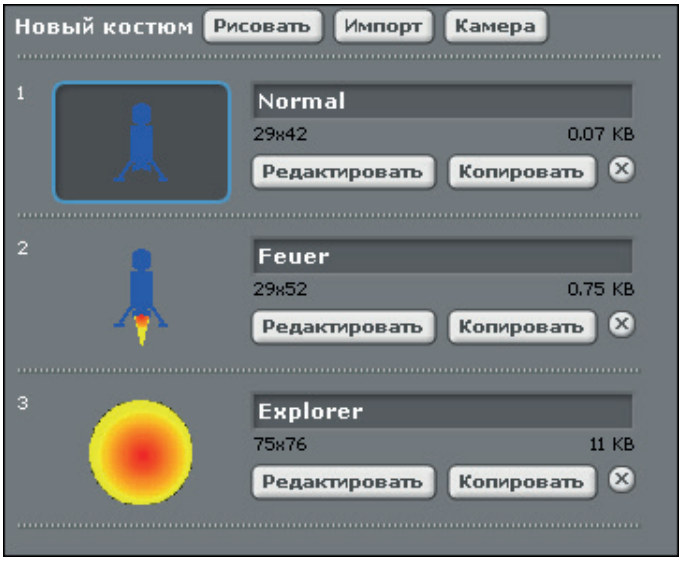


Рис. 6.24. *Костюмы для космического корабля*

Объекту **Космический корабль** понадобится четыре скрипта. Попробуй создать их из блоков, показанных на рис. 6.25.

Без использования Picoboard

Если у тебя нет Picoboard, то управлять кораблём ты можешь, нажимая клавишу **Пробел**. Для этого вместо командного блока 

установи командный блок


.

Ответы на вопросы

1. Сцене требуется скрипт, если фоновых картинок более одной. Скрипт меняет изображения фона.
2. Объект с помощью командного блока **передать** передаёт команду на смену фоновой картинки. В скрипте сцены содержится командный блок **когда я получу**. Этот командный блок принимает сообщение на смену фоновой картинки и выполняет действие, которое описывается в прикреплённом к нему командном блоке, например на смену фоновой картинки.
3. На зажимах «крокодил» напряжение 5 В.
4. Изменяется сопротивление.

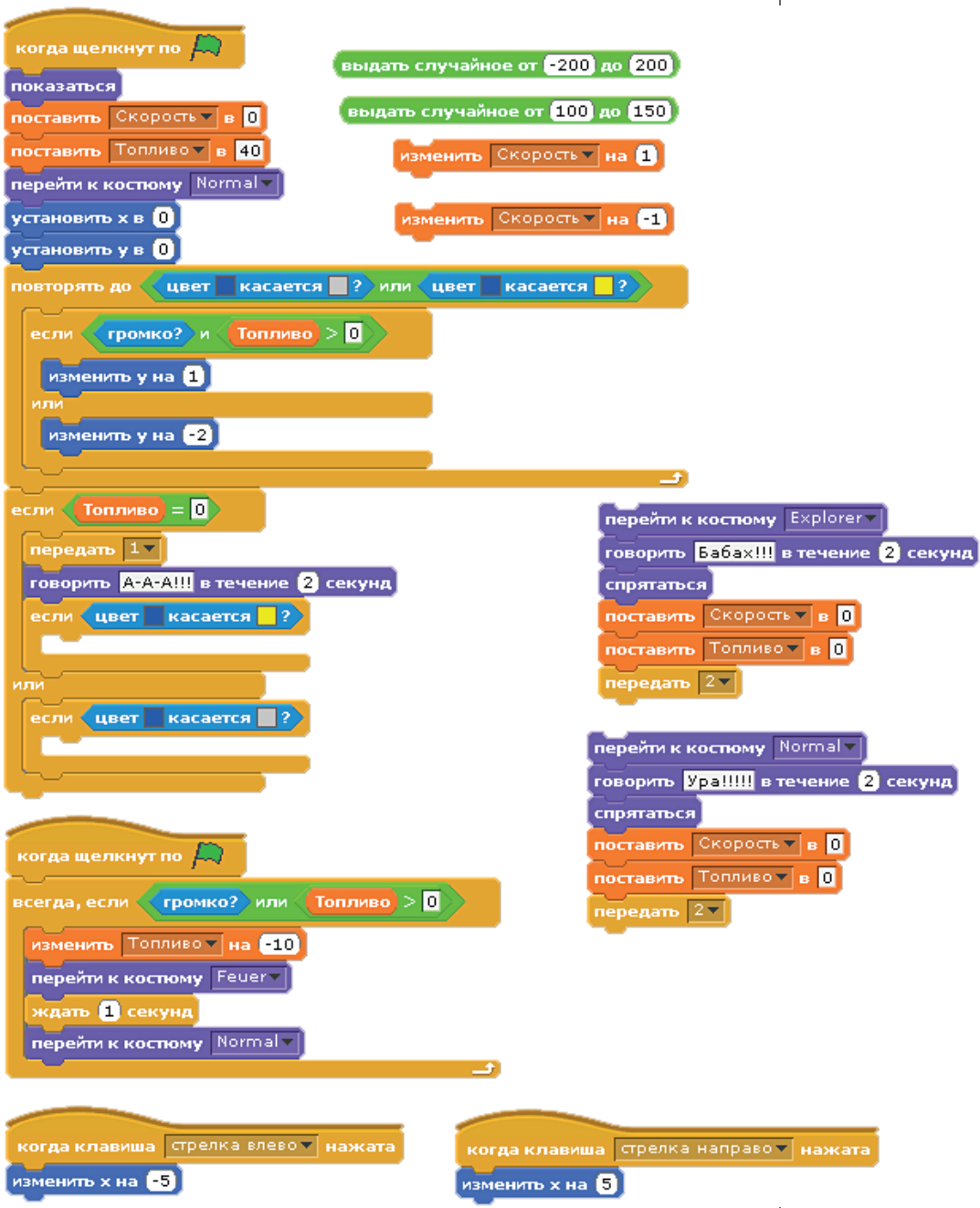


Рис. 6.25. *Из этих частей состоят скрипты для объекта* ***Космический корабль***

Решение заданий

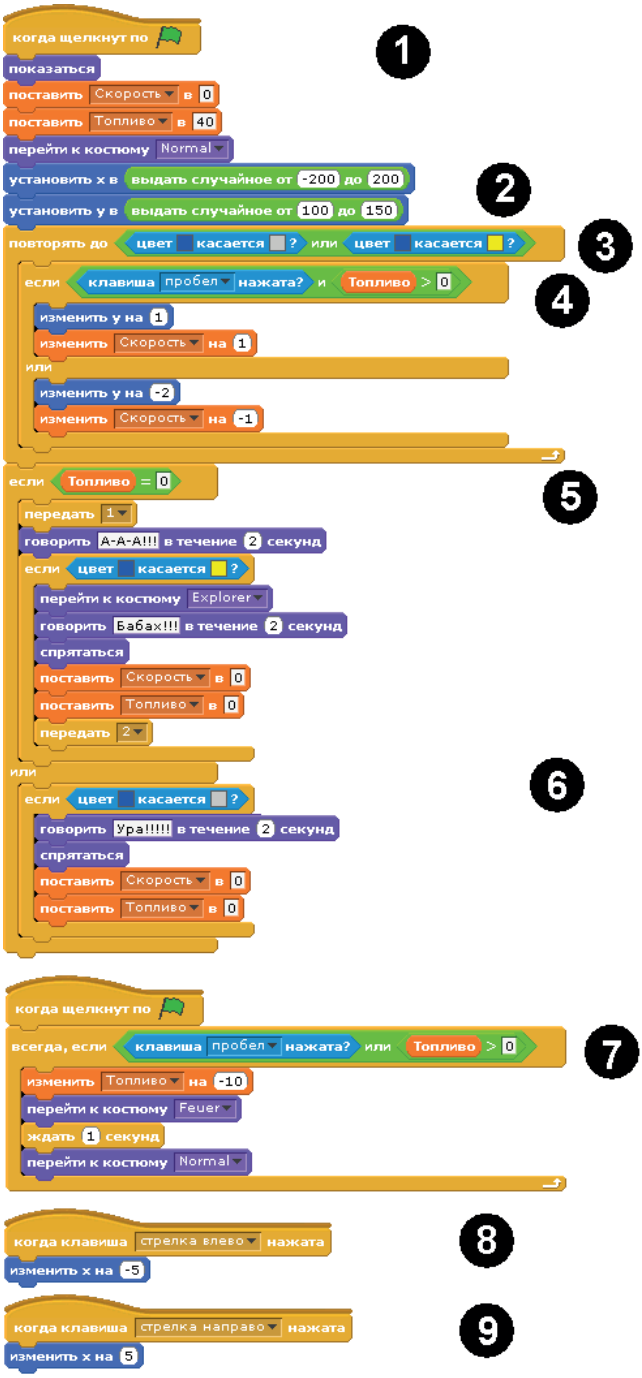


Рис. 6.26. *Скрипты для объекта Космический корабль*

Как работает скрипт

Полный скрипт для корабля показан на рис. 6.26.

Скрипт для фона ты увидишь на рис. 6.27.

1. Программа запускается после нажатия на кнопку  . Сразу после этого показания скорости обнуляются, а топливный бак «заправляется» под «пробку». Корабль «надевает» первый костюм, без пламени из сопел двигателя.
2. Положение по координатам **X** и **Y**, в которых появится корабль, выбирается случайным образом.
3. Начинает работать цикл, который работает до касания корабля (синий цвет) с грунтом (серый или жёлтый цвет).
4. В первый цикл **повторять до** вложен второй цикл **если – или**, описывающий изменение скорости снижения. Если клавиша **Пробел** не нажата, координата по Y уменьшается на 2 единицы за каждый цикл. Если клавиша **Пробел** нажата и **уровень топлива больше нуля**, координата по **Y** увеличивается на единицу. Повторяю, цикл повторяется до касания корабля с грунтом. Обрати внимание, здесь работает оператор **И**, учитывающий два важных и дополняющих друг друга условия. Если же уровень топлива становится равен нулю, нажатие клавиши **Пробел** падение корабля уже не остановит.
5. Если топливо закончилось (топливо = нулю), корабль начинает падать. Из корабля доносится крик «А-А-А!!!» и передаётся сообщение **1**. Это сообщение принимается скриптом, который управляет сменой фона (рис. 6.27). Выбирается фон с желтым грунтом. В скрипте управления кораблём запускается следующий цикл, в котором сохраняется условие: если (когда) цвет синий (корабль) касается цвета жёлтый (грунт на втором фоне), корабль надевает костюм Explorer (Взрыв), в течение 2 с говорит «Бабах!!!» и скрывается с экрана. Далее показатели **Скорость** и **Топливо** устанавливаются на 0, и посылается сообщение **2**. Скрипт фона принимает это сообщение и изменяет фоновый рисунок на тот, в котором грунт серый.
6. Если уровень топлива не равен 0, выполняется вторая ветвь цикла: при касании грунта (синий цвет соприкасается с серым цветом) корабль говорит «Ура!!!» и прячется. Показания скорости и топлива устанавливаются на 0.
7. Этот скрипт управляет расходом топлива (при каждом нажатии клавиши **Пробел** объем топлива уменьшается

на 10 единиц). Кроме того, при нажатой клавише **Пробел** корабль надевает костюм Feuer, из дюзы двигателя вырывается пламя.

8. При каждом нажатии клавиши ← смещается влево на 5 единиц.
9. При каждом нажатии клавиши → смещается вправо на 5 единиц.

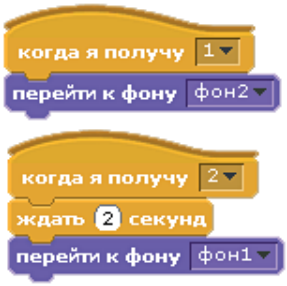
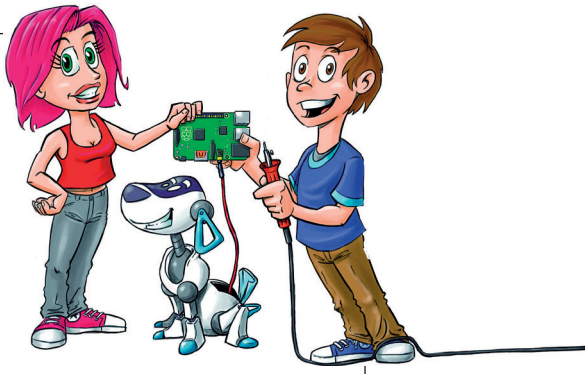


Рис. 6.27. *Скрипты для фона*



7
Знакомство с Python

Python – это ещё один язык программирования, установленный на Raspberry Pi. С его помощью пишутся тексты программ. Пример такого текста – это сама программа Scratch. Эта программа была написана как текстовые инструкции для компьютера, которые были обработаны таким образом, чтобы компьютер их понимал и исполнял. Подобная обработка называется компиляцией. Обрати внимание: в программе Scratch разрабатываемая нами игра собиралась из командных блоков и выполнялась только в самой программе Scratch.

В этой главе ты познакомишься с основами Python и сможешь написать свои первые программы.

Что такое Python?

В отличие от Scratch, Python является полноценным языком программирования, который используется в индустрии программного обеспечения для коммерческих проектов. Многие системы, с которыми ты знакомился в интернете, созданы на базе языка программирования Python: навигатор, поисковые машины, платформы для хранения данных в «облаке» (Cloud) (так называемые «облачные хранилища»). Профессионалы также с удовольствием используют Python, так как на этом языке разработка программ

идёт быстрее, чем на других языках. И в промышленности Python пришёлся как нельзя кстати, ведь время – это деньги. В школах и университетах всё чаще используют данный язык программирования, ведь он лёгок в изучении и позволяет за короткий период времени написать вполне понятные программные тексты. Python не зависит от платформ. То есть программы, которые ты напишешь для Raspberry Pi с помощью Linux, будут работать также и на других компьютерах, с другими операционными системами, например Windows или OS X.

Python имеет две версии – Python 2 и Python 3. В этой книге мы воспользуемся новой, третьей версией – Python 3.

Эта программа бесплатная. Она была создана голландским разработчиком Гвидо ван Россумом (Guido van Rossum) и в настоящий момент поддерживается организацией Python Software Foundation (http://www.python.org/psf).

Оболочка Python

Скрипт для Python состоит из инструкций. При запуске программы Python-интерпретатор последовательно прочитывает и выполняет все инструкции. Что такое интерпретатор? Компьютер не может в одиночку начать работу с текстами Python. Интерпретатор – это своего рода переводчик, программное обеспечение, которое построчно считывает тексты и сообщает компьютеру, что нужно делать.

Кроме того, есть возможность дать интерпретатору отдельные инструкции, чтобы он выполнил их напрямую. Такой способ хорош для экспериментов и лучшего понимания всех значений и команд. Именно этим мы и займёмся в данном разделе.

Рис. 7.1. *Значки для Python 2 и Python 3*

Щёлкни мышью по расположенной в левом верхнем углу кнопке . На экране появится основное меню. В появившемся меню открой **Программирование**. Справа появится

подменю с программами для программирования (рис. 7.2 слева).

Значок программы **Python 3 (IDLE)** напоминает двух скрученных змей: сверху – синяя, снизу – жёлтая. Правда, такое «змеиное» название для языка программирования Гвидо ван Россум выбрал вовсе не из любви к рептилиям, а в честь английской комик-группы «Монти Пайтон» (Monty Python).

IDLE – это стандартная среда разработки для Python. Среда разработки – это программное обеспечение, которое используют программисты для разработки программ. Оно состоит из оболочки (Shell) и блокнота:

- оболочка (Shell) – это окно, в которое вводятся собственные инструкции Python и в котором можно увидеть результат (числа или тексты);
- блокнот – это тоже окно, в котором ты можешь из нескольких инструкций написать несколько Python-программ и сохранить их там.

Щелкни мышью на строке Python 3 (IDLE). Через некоторое время откроется окно, как на рис. 7.2 справа. Это и есть **Python 3.5.3 Shell**.

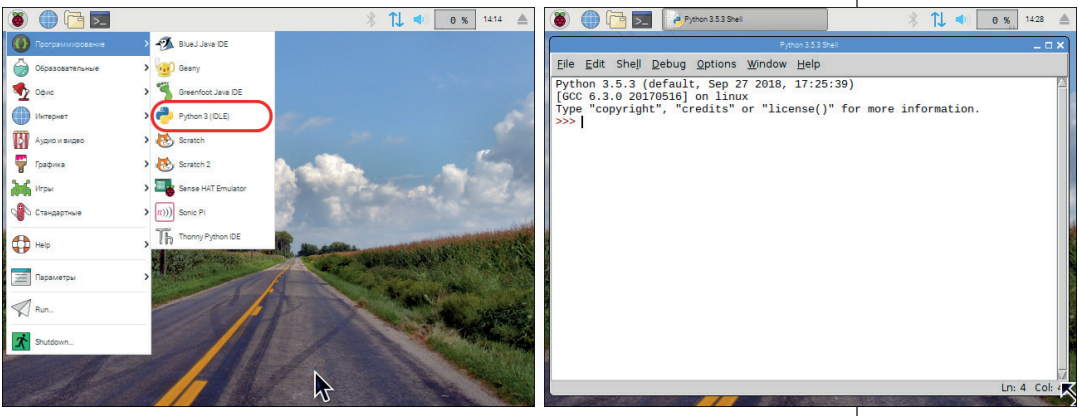


Рис. 7.2. *Запуск Python 3 (IDLE) с основного меню и внешний вид окна Python 3.5.3 Shell*

В верхней части окна **Python 3.5.3 Shell** видны три строки. В верхней строке – версия запущенной программы, во второй строке – в какой операционной системе **Python Shell** запущен. Третья строка предлагает для получения большей информации ввести команды `copyright` (авторское право), `license` (лицензия) и `more information` (больше информации). Нижняя строчка с тремя символами `>>>` называется командной строкой. Её ещё называют **Promt** (англ. «быстро»). В эту

строку вводятся команды. После того как ты введёшь нужную команду (инструкцию) и нажмёшь клавишу ↵, интерпретатор начнёт выполнять инструкцию и через некоторое время выдаст результат. Затем снова появится приглашение командной строки >>> на ввод следующей инструкции, и интерпретатор будет ожидать ее. Это как диалог или взаимодействие человека и машины. Поэтому работу с Python Shell называют *интерактивным режимом*.

Теперь попробуем ввести первые инструкции.

Введи 2+2 и нажми клавишу ↵. Получишь следующее:

```
>>> 2+2
4
>>>
```

Здесь мы видим инструкцию в виде математического выражения, так называемый *терм*. Если терм выражен некорректно, то появится сообщение об ошибке. Проверь сам: введи 2 + и нажми клавишу ↵.

```
>>> 2 +
SyntaxError: invalid syntax
>>>
```

SyntaxError: invalid syntax в переводе значит: «Синтаксическая ошибка: недопустимый синтаксис».

IDLE запускается в командной строке

Среду разработки IDLE для Python 3 можно запускать не только с основного меню, но и с командной строки терминала. Для этого в полосе меню Raspberry (в верхней части экрана) нажми кнопку >_ и введи в командную строку терминала команду `idle3` (этой командой запускается **Python Shell**) и нажми клавишу ↵.

Две самые важные комбинации клавиш

В этом разделе представлены две комбинации клавиш, которые помогут сэкономить время в работе с Shell.

Иногда захочется ввести команду, которую ты уже вводил ранее. Тогда нажми клавиши **Alt**+**P**. Буква **P** означает *previous*, т. е. «предыдущий, ранний». Допустим, ты вводишь:

```
>>> 2+2
4
>>>
```

Теперь, если ты нажмёшь комбинацию клавиш **Alt**+**P** один раз, в командной строке появится предыдущая команда:

```
>>> 2+2
```

А если ты комбинацию клавиш **Alt**+**P** нажмёшь ещё раз, то предыдущий терм, или команда, исчезнет и появится *предпоследняя* инструкция. Если комбинацию клавиш **Alt**+**P** нажать ещё раз, появится инструкция, введённая перед предпоследней командой, и т. д. Shell (оболочка) запоминает все введённые инструкции в той последовательности, в которой они сохранены в *history* (истории). Используя комбинации клавиш **Alt**+**P** и **Alt**+**N** (клавиша с буквой N значит *next*, т. е. «следующий»), ты всегда можешь сделать возврат в истории или перейти к следующей команде и найти там все инструкции, которые ты когда-либо вводил.

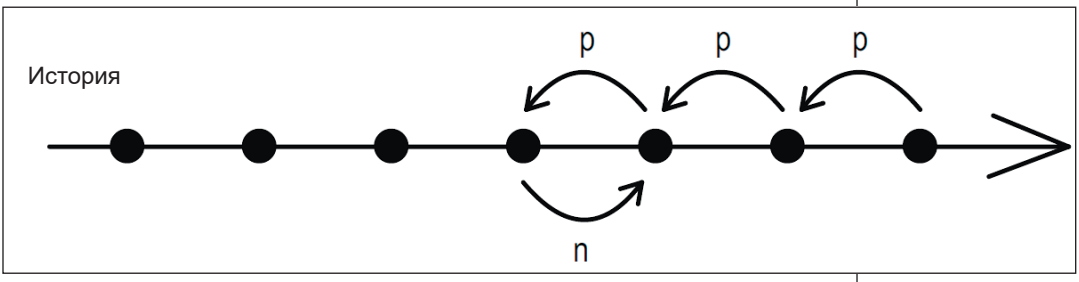


Рис. 7.3. *В истории запросов ты всегда можешь найти ранее введённые инструкции*

Python Shell и калькулятор

Python Shell ты можешь также использовать в качестве калькулятора. Для этого тебе нужно ввести математическое выражение (терм) и нажать клавишу ↵. В следующей строке появится результат.

Математические выражения

Математическое выражение может быть построено из цифр, операторов и круглых скобок. Пишутся они точно так же, как и те выражения, что ты изучал в школе, но есть несколько небольших особенностей.

При операции умножения ты будешь использовать «звёздочку» *, например:

```
>>> 100*21
2100
```

Знаков деления здесь нет. Деление выполняется через дробь. Чтобы ввести этот знак, достаточно нажать клавишу /. Для чисел или знаменателей тебе понадобятся скобки:

```
>>> (2+3)/2
2.5
```

Обрати внимание, что результат 2.5 выглядит не как 2,5. Десятичное число не содержит запятой, вместо неё ставится точка. Во многих странах такое написание является общепринятым.

Существует точное деление / и деление целого числа //. Деление целого числа (без остатка) всегда содержит только целое число. Здесь ты увидишь результат деления с округлением вниз. Попробуй:

```
>>> 3/2
1.5
>>> 3//2
1
```

Для возведения числа в степень тебе понадобится оператор **. Число 2^8 (читается как «два в восьмой степени») ты записываешь как `2**8`. По сути, это то же самое, что и `2*2*2*2*2*2*2*2`.

С программой Python ты сможешь вычислять бесконечно длинные числа.

```
>>> 2**8
256
>>> 5**1
5
>>> 5**2
25
>>> 5**-1
0.2
>>> 5**200
622301527786114170714406405378012424059025216872116713310111661478969883403538344118394482312571361695696658955512248212471604347 2900390625
>>>
```

Для выражений с несколькими операторами ты должен обращать внимание на их приоритетность. Впрочем, тебе это известно из школьного курса математики. Операции с высокой приоритетностью должны быть выполнены в первую

очередь. Самую высокую приоритетность имеет оператор возведения в степень, затем идут умножение и деление. Операции сложения и вычитания имеют самую низкую приоритетность.

```
>>> 2*3**2
18
```

В этом примере компьютер сначала 3 возвёл во вторую степень (3**2), после чего умножил на 2.

Экспериментируем далее.

```
>>> (2*3)**2
36
```

Выражение в скобках всегда вычисляется в первую очередь, так как скобки имеют наивысший приоритет.

Вызов функции

Наиболее часто в вычислениях ты будешь использовать функцию округления. Попробуй:

```
>>> round (1.4)
1
>>> round (1.5)
2
```

Функция `round()` округляет десятичное число до целого. Вызов этой функции состоит из нескольких действий, а именно:

- имени функции, например round;
- двух круглых скобок за ней;
- указанных в этих скобках данных (чисел), которые нужно проработать. Эти данные называются *аргументами*.

Аргументом может являться и математическое выражение. Попробуй:

```
>>> round(8/3)
3
```

Как это работает? Сначала вычисляем 8/3 и получаем что-то вроде 2.666… (2 и 6 в периоде). Затем это число округляется, и мы получаем 3. Это тебе должно быть известно из уроков математики. То есть если значение числа после запятой от

1 до 4 (например, 2.3), число округляется до предыдущего целого числа, в нашем примере до 2. Если значение числа после запятой в диапазоне от 5 до 9 (например, 2.7), число округляется до следующего целого числа. В приведённом примере до 3.

Ты можешь округлять числа на несколько разрядов после запятой. Для этого введи через пробел второй аргумент, т. е. число разрядов, на которое оно должно быть округлено. Попробуй, но при вводе обрати внимание: после 1/3 нужно поставить не точку, а запятую:

```
>>> round(1/3, 1)
0.3
>>> round(1/3, 2)
0.33
```

Если ты вторым аргументом введёшь значение 1, результат 1/3 округлится до одной цифры после запятой. Если вторым аргументом будет введено значение 2, результат деления округлится до второй цифры после запятой, и т. д.

Кстати, когда ты начнёшь вводить функцию, введёшь слово `round` и откроешь скобку, ниже появится подсказка (рис. 7.4). Угловые скобки [...] означают, что второй аргумент *не является* обязательным.

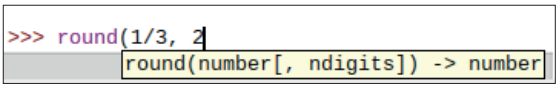


Рис. 7.4. *Стоит тебе ввести имя функции, сразу же появится окошко с текстом*

Который час? Тестируем функцию времени

Многие функции программы всегда доступны, так как являются стандартными. К ним относятся и функции `round()` и `min()`. В дополнение к стандартным функциям имеются и другие. Все они хранятся в так называемых модулях. **Модуль `time`** – это коллекция функций, так или иначе связанных со временем. В этом модуле имеется функция `asctime()`, которая показывает короткий текст с датой и временем. Прежде чем ты сможешь ею воспользоваться, её необходимо *импортировать*.

Это примерно то же самое, как если бы ты отправился в подвал за каким-то специальным инструментом, который не всегда имеется у тебя под рукой.

Попробуй ввести следующие команды. Сначала введи `from time import asctime`, нажми клавишу ↵, введи команду `asctime()` и нажми клавишу ↵:

```
>>> from time import asctime
>>> asctime ()
'Fri Mar 1 20:07:16 2019'
```

Функцию `asctime()` можно вызвать без какого-либо аргумента (скобки `()` должны быть обязательно).

Импортируем модуль

Ты также можешь импортировать весь модуль целиком. Введи команду `import time` и нажми клавишу ↵. Модуль будет импортирован, но на экране ничего не произойдёт.

```
>>> import time
>>>
```

Теперь вызови саму функцию `time.asctime()`:

```
>>> time. asctime ()
>>> 'Fri Mar 1 20:19:44 2019'
```

Обрати внимание: при вызове функции после её названия ставится точка (`time.`). Аргумент пишется с маленькой буквы.

Первый скрипт для Python

Программы состоят из одной или нескольких инструкций, которые компьютер в состоянии выполнить автоматически. Программа, написанная на языке Python, тоже называется *скриптом*. В этом разделе ты узнаешь, как написать, сохранить и выполнить скрипт для Python с помощью IDLE.

Создаём новый проект

Python-скрипт сохраняется в виде файла. Поэтому в самом начале тебе следует создать папку для сохранения своих проектов. Это можно сделать с помощью файлового менеджера.

- Если у тебя **Python Shell** запущен, сверни его или заверши его работу. Чтобы свернуть, достаточно нажать

кнопку . Завершить работу приложения ты сможешь, нажав кнопку . Эти кнопки находятся в правом верхнем углу приложения.

- В панели задач операционной системы Raspberry щёлкни мышью по значку . Запустится файловый менеджер.
- В левой части файлового менеджера проверь, что ты находишься в папке *pi* (рис. 7.5). Для этого просто щёлкни мышью по этой папке. Папка выделится, а это значит, что ты находишься именно в этой папке.
- Выбери команду меню **File** ⇨ **Create Folder** (Файл ⇨ Создать папку). Появится диалог, в поле ввода которого нужно ввести название создаваемой папки.
- Дай папке подходящее по смыслу имя, например *Python_project* (Проекты Python) и нажми клавишу ↵. Готово.

Обрати внимание: между словами *Python* и *projects* обязательно нужно поставить знак нижнего подчёркивания _ и никаких пробелов.

- Закрой файловый менеджер, нажав кнопку в правом верхнем углу.

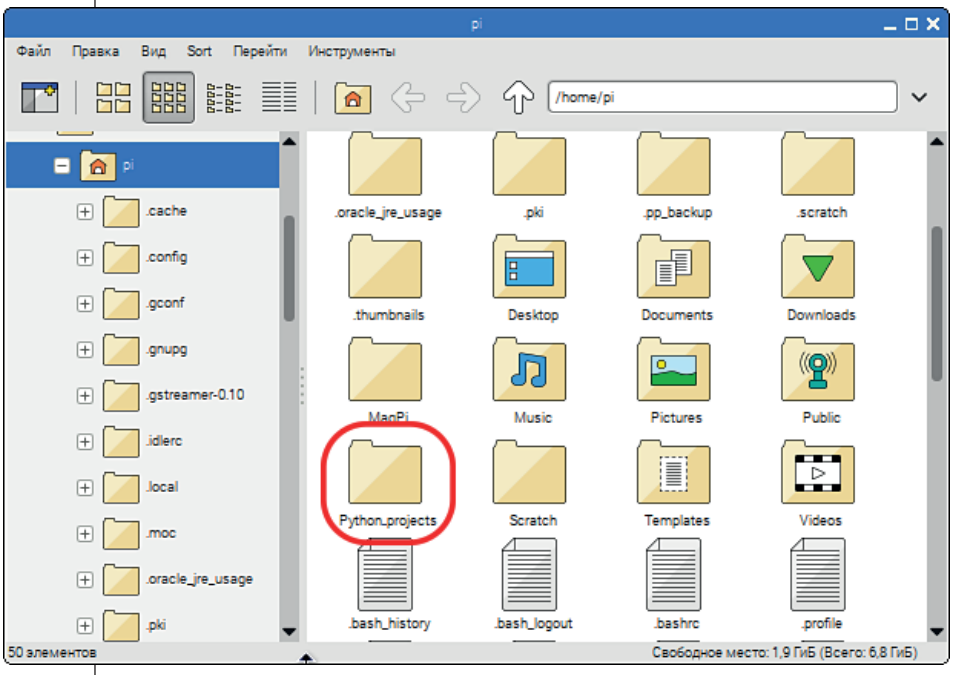


Рис. 7.5. *Создание новой папки для проектов Python*

Это было подготовкой.

- Теперь запусти Python 3 (IDLE).
- В полосе меню в верхней части экрана выбери команду меню **File** ⇨ **New File** (Файл ⇨ Новый файл). Будет открыто новое окно.

Это *блокнот* IDLE, т. е. та часть среды разработки, с помощью которой пишутся программы.

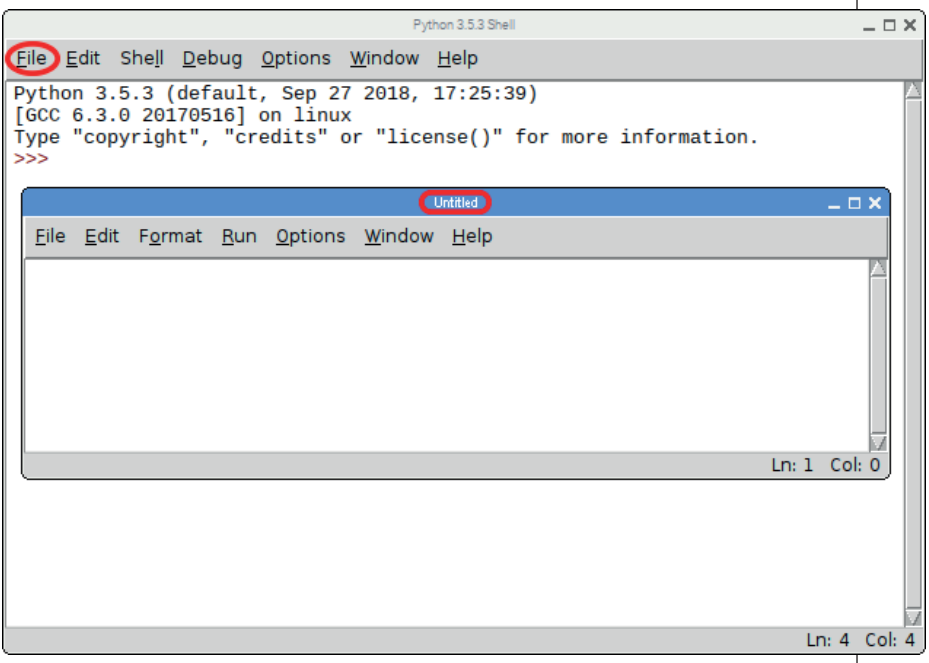


Рис. 7.6. *Пустое окно блокнота IDLE*

Несмотря на то что у нас пока ещё не написано ни одной строчки программы, тебе нужно сохранить этот проект под именем, кратко описывающим суть своего проекта.

- Выбери в блокноте команду меню **File** ⇨ **Save As...** (Файл ⇨ Сохранить как...). На экране появится диалог **Сохранить как** (Save As), в поле которого ты увидишь папки, сохранённые в папке *pi*.
- В появившемся диалоге найди и дважды щёлкни мышью по папке *Python_projects*.

Первая программа должна вывести текущее время. Значит, подходящим именем для неё будет, например, *Vremia.py*.

- Введи в поле ввода **Имя файла** (File name) имя *Vremia* и нажми клавишу ↵. Проект сохранён, а диалог **Сохранить как** скроется.

Обрати внимание на окончание имени файла *.py*. Его вводить не надо, окончание файла будет введено автоматически. Это окончание служит для опознания файла компьютером. Опознав файл, компьютер будет знать, с помощью какого приложения его открыть. Если окончания этого файла не будет, IDLE не сможет распознать программу Python. Окончание файла определяется из нижнего открывающегося списка **Тип файлов** (File Type).

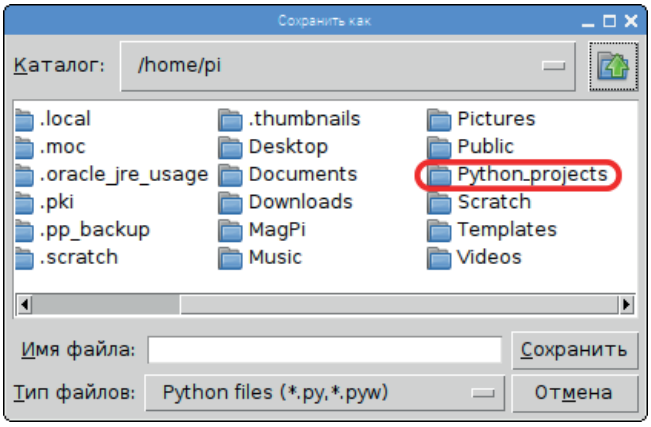


Рис. 7.7. *Сохраняем скрипт для Python в папке проектов*

Теперь приступим к написанию нашей первой программы. Для этого в окне блокнота вписываешь следующий текст:

```
from time import asctime
print('current time')
print(asctime())
```

Следи за тем, чтобы строка начиналась *без пробелов*! Отступы в Python имеют особое значение. Но об этом поговорим чуточку позже.

Как это работает?

Что означают эти три программные строки? Разберём их по отдельности.

```
from time import asctime
```

Из модуля `time`, набора функций времени, импортируется функция `asctime()`.

```
print ('current time ')
```

Чтобы что-то показать на экране, используют функцию `print()`. Этот английский глагол в русском языке означает *печать*. Но компьютерщики здесь имеют в виду *отображение* информации. Текст, появляющийся на экране, – это *отображение* программы. Текст, который требуется ввести, заключается в скобки, а после скобок – в верхние одинарные кавычки. Следующая строка:

```
print(asctime())
```

Здесь не выводится никакого определённого текста, поэтому и верхней одинарной кавычки тоже нет. Вместо неё в скобках стоит второй вызов функции. Функция `asctime` показывает текущее время, а после `print()` записывает его на экран.

Готовый скрипт ещё раз сохраняем.

- Выбери команду меню **File** ⇨ **Save** (Файл ⇨ Сохранить). Программный текст сохраняется под введённым ранее именем.

Выполняем скрипт

Чтобы запустить программу, в верхней части открытого окна блокнота выбери команду меню **Run** ⇨ **Run Module** (Выполнить ⇨ Модуль). Если окно **Python Shell** не было открыто, откроется это окно (мы с ним познакомились в предыдущем разделе), запустится Python-интерпретатор. Он считывает программный текст, переводит его в команду, которую процессор Raspberry способен понять, и обеспечивает выполнение этой команды.

В окне **Python Shell** ты увидишь вывод программы или сообщение об ошибке, если была сделана опечатка. В этом, конечно же, и состоит отличие Python от Scratch! Ошибки в написании (их ещё называют синтаксическими) происходят в работе с Python постоянно. Зачастую требуется несколько раз протестировать программу, прежде чем она действительно заработает.

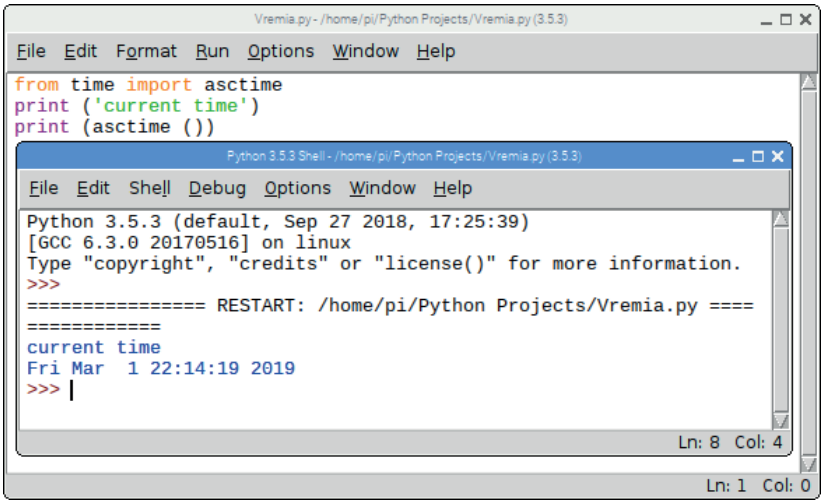


Рис. 7.8. *В верхней части блокнота находится программный текст, а в нижней части окна* ***Shell*** *– вывод на экран*

Запуск программы Python в консоли

Ты можешь запустить скрипт прямо из IDLE. Это очень удобно, когда хочется протестировать программу, которую ты сейчас разрабатываешь. Но есть и другой путь. Можно запустить программу из командной строки в консоли (LXTerminal). Делается это так.

- Открой LXTerminal (в полосе меню Raspberry щёлкни мышью по кнопке).
- Перейди в папку, в которой сохранён скрипт. Для этого в командную строку введи команду `cd Python_projects` и нажми клавишу [↵].

С помощью команды Linux `cd` ты переходишь в папку, в которой находится твоя программа. При этом удобно автоматически заполнять имена каталогов с помощью клавиши табуляции. Если ты находишься в домашнем каталоге (pi@raspberry), то, чтобы попасть в целевую папку, не нужно вводить её полное имя. Лучше после нужной команды введи две-три первые буквы названия каталога и нажми клавишу [⇆]. Введи:

```
$ cd Py
```

Далее нажми клавишу [⇆]. В командной строке после команды `cd` появится полное имя каталога:

```
$ cd Python_projects/
```

Чтобы перейти в выбранный каталог, нажми клавишу ↵.

Ты можешь посмотреть, какие файлы находятся в каталоге. Для этого введи команду `ls` и нажми клавишу ↵. Ниже командной строки появится список файлов, находящихся в выбранном каталоге.

Запусти программу с помощью следующей команды:

```
$ python3 Vremia.py
```

Запустится Python-интерпретатор, который выполнит скрипт *Vremia.py*.

Вывод программы отобразится непосредственно под этой командой.

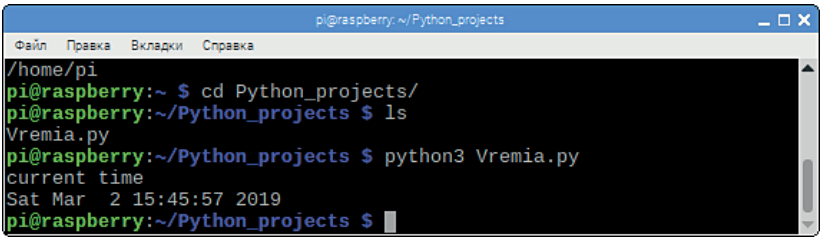


Рис. 7.9. *Запуск скрипта Python в консоли*

Программа с графическим пользовательским интерфейсом

При работе с компьютерами ты привык запускать программу, щёлкая мышью по её ярлыку. После этого открывается окно программы, и в нём уже что-то происходит. Программу с графическим интерфейсом называют пользовательской, или приложением. Такие проекты ты тоже можешь запрограммировать в Python. Этой теме в книге посвящена целая глава. Но для знакомства с графическим интерфейсом приведу тебе маленький пример – электронные часы.

Рис. 7.10. *Пользовательское окно с отображением текущего времени*

Программа выводит маленькое окно, в котором отображается текущее время, как на рис. 7.10. Значок «решётка» # перед первой строкой и знаки #1–#6 можешь не вводить. Эти знаки в тексте программы поставлены, чтобы далее рассказать тебе о каждой строке и ты понял, к какой строке какой комментарий относится.

- Запусти программу **Python Shell**.
- Выбери команду меню **File** ⇨ **New File** (Файл ⇨ Новый файл).
- Введи приведённый ниже текст программы.

```
#!/usr/bin/python3                                          #1
import time, tkinter                                        #2
window = tkinter.Tk()                                       #3
label = tkinter.Label(master=window.text=time.asctime())    #4
label.pack()                                                #5
window.mainloop()                                           #6
```

Сохрани эту программу в каталоге *Python_project* с проектами под именем *electronic_time.pyw*.

Для этого в поле ввода **Имя файла** введи имя полностью, с окончанием. Обрати внимание на окончание имени файла. Вместо *py* там должно находиться *pyw*.

- Проверь работу программы. Для этого в полосе меню блокнота выбери команду **Run** ⇨ **Run Module F5** (Выполнить ⇨ Выполнить в модуле F5).

Как это работает

`#1` Первая строчка называется «магической линией», она всегда начинается с последовательности символов `#!`. Это строка, обозначающая начало программы. За ней вводится путь хранения программного Python-интерпретатора, который программа должна выполнить. Помни, что на твоём Raspberry установлены две разные версии Python.

`#2` Эта строка импортирует два программных модуля. Модуль `tkinter` содержит все требования, относящиеся к графическому интерфейсу. Этому типу импорта при вызове функции всегда должно предшествовать имя модуля (см. следующую инструкцию).

`#3` Команда `window=tkinter.Tk()` вызывает функцию `Tk()` из модуля `tkinter`. С помощью этой команды создаётся новое окно приложения. Это внешняя рамка программы. Такое окно содержит имя *window*.

#4 Здесь формируется **Label** (Ярлык). Это и есть белое прямоугольное поле, в котором будет отображаться текст. Только компьютер знает, куда поместить этот ярлык. С помощью команд `master=window` ярлык присваивает имя *window* окну приложения. А с помощью `text=time.asctime()` в ярлык вписывается текущее время и дата.

#5 До этих пор только ярлык был связан с окном. Но этот ярлык нужно поместить в определённом месте этого окна. Размещение ярлыка и называется **Layout** (Разметка). С помощью этой команды ярлык помещают в окно так, чтобы вся площадь окна была заполнена.

#6 Эта строка программного кода активирует наше приложение.

Этот Python-скрипт запускается непосредственно с помощью *магической линии,* благодаря которой операционная система поймёт, что программный текст должен быть запущен с помощью Python 3. Больше об этой программе и работе с графическим интерфейсом ты узнаешь в главе 16.

Интерактивные программы

В работе с интерактивными программами происходит своего рода диалог между человеком и компьютером. В этом разделе мы разработаем одну такую программу, которая будет спрашивать у пользователя его имя, а также приветствовать его.

Личное приветствие

Итак, на экране мы видим:

```
Как тебя зовут?
Имя:
```

После двоеточия мерцает курсор. Компьютер ожидает, когда на клавиатуре введут текст.

Начало строки `Имя` называется *подсказкой*. Ну, здесь понятно, что имеется в виду – нужно ввести имя.

Допустим, пользователя зовут Анна. Смотрим дальше:

```
Как тебя зовут?
Имя: Анна
```

Как только пользователь нажимает ↵, процесс ввода имени будет завершён, и появится личное приветствие:

```
Привет, Анна
>>>
```

Программа

Здесь программа Python в точности выполняет то, что было описано. Три строки, которые объясняют весь процесс.

```
print('Как тебя зовут?')
name = input('Имя:')
print('Привет', name)
```

Порядок действий по написанию программы

Теперь введём эту программу в блокнот Python Shell и протестируем её.

- Если ты Python Shell закрыл, запусти его.
- В полосе меню Python Shell выбери команду меню **File** ⇨ **New File** (Файл ⇨ Новый файл). Откроется новое окно блокнота.
- Выбери команду меню **File** ⇨ **Save as...** (Файл ⇨ Сохранить как...), найди и дважды щёлкни мышью по каталогу *Python_project* (или по имени каталога, в котором хранятся твои проекты).
- Введи в поле ввода имя файла *hello.py* и нажми кнопку **Сохранить** (Save).
- Введи три строки программного текста. При вводе обрати внимание: в начале командных строк никаких заглавных букв, только прописные. Следи, чтобы не было лишних пробелов. Впрочем, если ты допустишь ошибку в синтаксисе, компьютер тебе укажет на это. На экране появится диалог **invalid syntax** (недопустимый синтаксис), а в блокноте, возможно, ошибка будет выделена.
- Выбери в полосе меню блокнота команду меню **File** ⇨ **Save** (Файл ⇨ Сохранить) или нажми комбинацию клавиш **Ctrl**+**S**.

Если ты забудешь внесённые изменения сохранить и нажмёшь клавишу **F5**, компьютер тебе об этом напомнит, выведя на экран диалог **Source Must Be Saved OK to Save?** (Источник должен быть сохранён, для сохранения нажми кнопку OK).

- Запусти программу. Для этого нажми клавишу **F5** или выбери в полосе меню блокнота команду меню **Run** ⇨ **Run Module F5** (Выполнить ⇨ Выполнить в модуле F5).

Как это работает?

Теперь рассмотрим эти три инструкции по порядку.

```
print('Как тебя зовут?')
```

Текст выводится с помощью функции `print()`. В скобках пишется текст, который требуется вывести на экран. Обрати внимание: сам выводимый на экран текст обязательно следует заключить в верхние одиночные кавычки. Строка, заключённая в верхние одиночные кавычки, называется строкой символов. Сами кавычки на экране после вывода текста не появляются.

Следующая строка:

```
name = input('Имя:')
```

Функция `input` позволяет пользователю вводить данные с клавиатуры. В скобках, после имени функции, находится текст командной строки, т. е. `'имя:'`. Это снова строка символов, которая выводится на экран. Не забудь строку символов заключить верхними одиночными запятыми. Пробел после двоеточия относится к строке символов и выводится на экран после слова `Имя`.

Во время выполнения команды `input()` компьютер ждёт, пока пользователь введёт текст и нажмёт клавишу ↵. После того как текст был введён и клавиша ↵ нажата, компьютер сохраняет этот текст (например, `'Анна'`) как переменную `name`. Заметь, имена переменных желательно писать только на английском языке. Но это не догма, так как Python уже воспринимает и кириллицу. То есть ты можешь написать как `name`, так и `имя`. Важно, чтобы в программе переменная именовалась на одном языке. Или кириллицей, или латиницей. И имя переменной в начале второй строки (`name`) должно в точности совпадать с переменной (`name`) в третьей строке. Это называется *переменной*. Подобное действие можно представить себе, как если бы кто-то в коробку (контейнер) с надписью «Имя» положил листок бумаги с написанным на нём именем Анна (рис. 7.11 слева).

```
print('Привет',name)
```

Здесь была вызвана функция `print()`. На этот раз у этой функции два аргумента, которые разделены запятой. Это

две строки, которые вписываются на экране подряд друг за другом. Первая строка – слово «привет». Вторая содержит слово «имя», но это не слово, а функция, которая *не* пишется на экране. На экране выводится содержание этого контейнера.

```
Привет имя
```

Взгляни на третью программную строку ещё раз! Второй аргумент «name» не отделён верхней запятой, а это значит, что здесь подразумевается *содержание* переменной `name`, т. е. то, что пользователь уже вводил ранее с клавиатуры. Если мы представим переменную в виде коробки (контейнера) с содержимым, то выглядеть это будет так, словно мы сделали копию содержимого, а затем её выдали (рис. 7.11 справа).

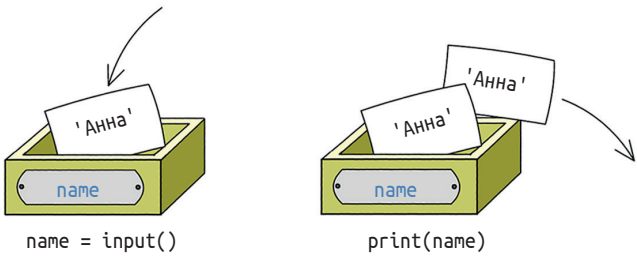


Рис. 7.11. *Программа запоминает в переменной текст, который был введён с клавиатуры*

Ввод и вывод данных

Многие компьютерные программы работают по следующему принципу.

- Сначала запрашиваются входные данные (вводится информация).
- Из этих данных программа высчитывает новые данные (обрабатывает их).
- Результат всего этого выводится на дисплей (выдаёт информацию).

Последовательность этих действий называется *вводом и выводом данных*.

Проект 17. Тормозной путь

Мы разработаем интерактивную Python-программу для решения следующих задач:

вначале программа определит скорость V и величину замедления при торможении A транспортного средства, а затем выдаст длину тормозного пути.

Замедление указывает, насколько быстро снижается скорость транспортного средства при полной его остановке. Оно зависит и от состояния дорожного покрытия, и от погодных условий. Тормозной путь автомобиля на мокром гравии гораздо длиннее, чем на сухом асфальте. Таблица 7.1 демонстрирует несколько показателей замедления автомобиля при торможении на разных типах покрытия, рассчитанных для легкового автомобиля.

Таблица 7.1. Замедление при торможении легкового автомобиля на различных видах дорожного покрытия

Дорожное покрытие	Замедление автомобиля при торможении
Сухой асфальт	7,5–8,0 м/с^2
Мокрый асфальт	6,0 м/с^2
Сухое бетонное покрытие	7,5 м/с^2
Мокрое бетонное покрытие (новое полотно)	7,0 м/с^2
Мокрое бетонное покрытие (старое полотно)	5,0 м/с^2
Сухая брусчатка	7,0 м/с^2
Мокрая брусчатка	5,5 м/с^2
Сухая булыжная мостовая	6,0 м/с^2
Мокрая булыжная мостовая	5,0 м/с^2
Сухой песок или галька	5,5 м/с^2
Мокрый песок или галька	4,5 м/с^2
Заснеженное дорожное покрытие	2,0–3,0 м/с^2
Лёд (независимо от температуры таяния льда)	0,5–2,0 м/с^2

А это скрипт для Python, отвечающий всем требованиям.

```
# Ввод
ввод_a = input( 'Замедление при торможении: ')
ввод_v = input('Скорость (км/ч): ')
```

```
# Обработка
a = float(ввод_a)
v = float(ввод_v)/3.6
stopping_distance = v*v/(2*a)

#Вывод
print('тормозной путь (м):')
print(round (stopping_distance,2))
```

Введи эту программу в блокнот и сохрани её в своей папке под именем *stopping_distance.py*. Запусти программу клавишей **F5**. Сама программа подробно описана ниже. Не волнуйся, если тебе сейчас не всё понятно.

Поехали. После нажатия клавиши **F5** программа в Python Shell выполняет следующие действия.

Выводит на экран строку

```
Замедление при торможении:
```

Далее компьютер ждёт, когда ты введёшь нужное значение, которое выбираешь из табл. 7.1. Значение выбирается в зависимости от состояния дорожного покрытия. Введи, например, значение 7.5 для сухого асфальта (обрати внимание: между целой и дробной частями значения нужно ставить **точку**, а не запятую, иначе компьютер выдаст ошибку) и нажми клавишу ↵. На экране появится:

```
Тормозной путь: 7.5
Скорость (км/ч):
```

Теперь нужно ввести скорость автомобиля. Введи, например, значение 50 и нажми клавишу ↵. На экране появится результат:

```
Тормозной путь: 7.5
Скорость (км/ч): 50
Длина тормозного пути (м):
12.86
```

Комментарии

Программный текст может содержать *комментарии*. Комментарии кратко объясняют, что происходит в программе, а также делают программный текст понятнее и читабельнее. Это комментарии для человека, не для компьютера. Программа прекрасно работает и безо всяких объяснений.

Комментарий обозначается значком #. Весь текст за «решёткой» интерпретатор просто пропускает. Комментарии не являются инструкцией. Поэтому при вводе программы комментарии можно и пропустить. В примере программы разделы были помечены комментариями, например `# Ввод`, `# Обработка` и `# Вывод`.

Далее мы рассмотрим, как работает программа.

Ввод данных

Вводим строчки программы по порядку.

```
# Ввод
ввод_a = input('Замедление при торможении: ')
ввод_v = input('Скорость (км/ч): ')
```

В первой части программы происходит маленький диалог между человеком и машиной. Пользователь вводит с клавиатуры показатели для замедления торможения и для скорости. Эти показатели будут сохранены как строки символов в переменных `ввод_a` и `ввод_v`.

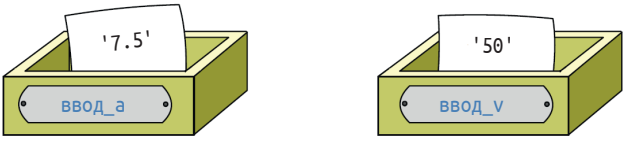


Рис. 7.12. *Переменные, содержащие строки символов*

Обработка и преобразование типов

```
# Обработка
a = float(ввод_a)
```

Функция `input()` всегда выводит символы, будь то буквы или цифры. Компьютер производит расчёты, оперируя цифрами, а не символами. Даже если у тебя в **строке символов** находится число, компьютер это число всё равно воспринимает как **символ**. Поэтому этот символ сначала требуется преобразовать в число с плавающей точкой (десятичную дробь), а только после этого производить расчёты. За такое преобразование отвечает функция `float()`. Рисунок 7.13 наглядно демонстрирует, как можно представить себе выполнение этой инструкции. Важно: переменная A содержит не символ, а число (это число уже в верхние одинарные запятые не заключается). Теперь можно приступать к вычислениям.

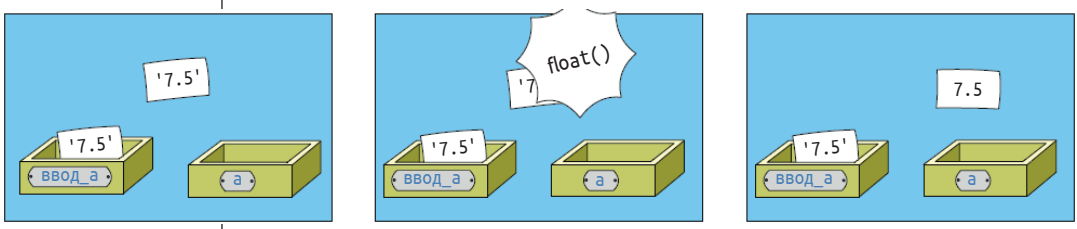


Рис. 7.13. *Содержимое ввод_а преобразовывается в десятичную дробь и закрепляется за переменной А*

```
v = float(ввод_v)/3.6
```

В этой команде значение в скобках – это символьная запись числа, которое с помощью функции `float()` нужно преобразовать в число с плавающей запятой. Мы сейчас не будем рассматривать, что значит число с плавающей запятой. Просто скажем, что с помощью такого формата представления числа компьютеру легче считать. Итак, с помощью команды `float()` мы в первую очередь преобразуем символы `ввод_v` в число с плавающей запятой. После чего полученное таким образом значение умножаем на 3,6. Это делается, чтобы количество километров в час преобразовать в количество метров в секунду. Один километр равен 1000 метрам, а один час содержит в себе 3600 секунд. Таким образом, мы имеем:

1 км/ч = 1000 м / 3600 с = 1 / 3,6 м/с.

Переходим к следующей строке:

```
stopping_distance = v*v/(2*a)
```

Здесь рассчитывается тормозной путь, а данные сохраняются в переменной `stopping_distance` (англ. *stopping_distance* – «тормозной путь»). На кириллице писать имена переменных нельзя. И обрати внимание на ещё одну вещь. Между словами `stopping` и `distance` обязательно должен стоять знак нижнего подчёркивания _.

Что означает эта расчётная формула? Значение скорости возводится в квадрат (умножается на себя), а потом делится на значение замедления при ускорении, умноженном на два. Обрати внимание: значение замедления при торможении тоже в начале программы вводится и хранится в виде символов. Эти символы, прежде чем подставлять в формулу

в виде переменной a, нужно сначала преобразовать в число с плавающей запятой. Этим занимается третья строка нашего программного кода: a = float(ввод_a).

Вывод данных

Рассмотрим третью часть нашей программы.

```
# Вывод
print('Тормозной путь (м): ')
print(round (stopping_distance.2))
```

Первая функция print() выводит неизменяемый текст 'Тормозной путь:'. При выполнении второй функции print() происходит следующее. Вначале с помощью функции round() ранее полученное значение длины тормозного пути, которое сохраняется в переменной stopping_distance, округляется до двух разрядов после запятой. После чего это значение выводится на экран.

Улучшение процедуры вывода

В образце программы функция вывода распределена на две строки. После каждого вызова функции print() по умолчанию выходит новая строка. Один вывод функции print() может вывести несколько значений. Между каждым фрагментом введённого текста Python-интерпретатор вставляет пробел.

Замени в программе выходную часть следующей инструкцией и удали последнюю строчку программного кода: print(round (stopping_distance,2)). При вводе строки программного кода обрати особое внимание на знаки препинания. После символов 'Тормозной путь:' должна стоять запятая, а не точка! Так же, как после (stopping_distance.2) нужно поставить запятую, а не точку. Иначе программа выдаст ошибку.

- Измени шестую командную строку так, чтобы она выглядела: print('Тормозной путь:', round (stopping_distance,2),'m').
- Удали строку: print(round (stopping_distance,2)).
- Сохрани внесённые изменения, нажав комбинацию клавиш **Ctrl**+**S**.
- Нажми клавишу **F5**.

И результатом этого будет:

```
тормозной путь: 12.86 м
```

Как загрузить образец программы?

Программные тексты проекта **Тормозной путь** и другие образцы ты найдёшь на сайте этой книги. Обрати внимание: все программы рассчитаны на немецкоязычного читателя. Если ты решишь, что тебе архив с такими программами будет полезен, для загрузки выполни следующие действия.

Щёлкни мышью по ярлыку браузера и введи в адресную строку адрес: www.mitp.de/767.

- Прокручивая вертикальную полосу прокрутки, отобрази на экране ярлык вкладки **Downloads** (Загрузки). Рядом с этим ярлыком ты увидишь ярлык вкладки **Inhalt** (Содержимое).
- Щелкни мышью по ярлыку вкладки **Downloads** (Загрузки) и нажми в этой вкладке кнопку **Downloads** (Загрузки). Начнётся загрузка архива с программным обеспечением. Архив по умолчанию будет сохранён в каталоге */home/pi/Downloads*.
- Закрой браузер, щёлкнув мышью по кнопке в правом верхнем углу окна программы.
- Запусти файловый менеджер, щёлкнув мышью по ярлыку, и зайди в каталог **Downloads** (Загрузки).
- Щёлкни правой кнопкой мыши по архиву *9783958457676_Projecte.zip* и выбери из появившегося меню команду **Извлечь в** (Extract to). На экране появится диалоговое окно **Извлечение из архива** (Extract from archive) (рис. 7.14).

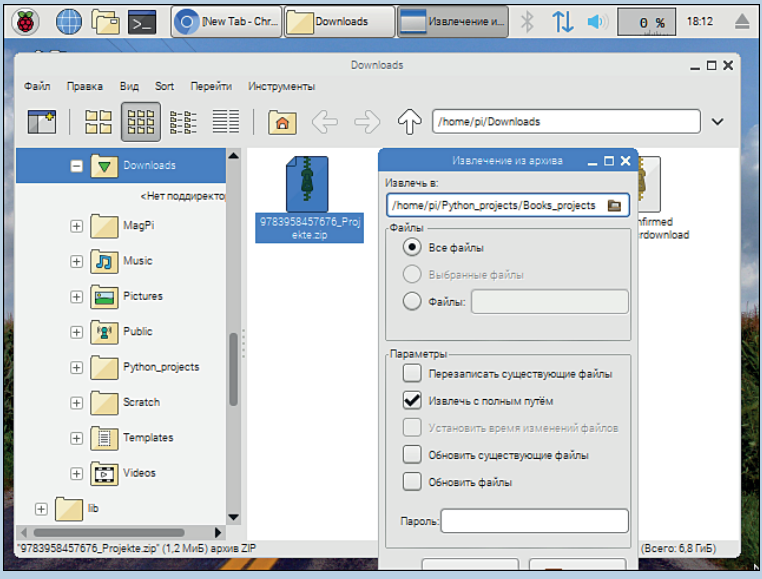


Рис. 7.14. *Диалог* ***Извлечение из архива*** *(Extract from archive)*

- В поле ввода **Извлечь в** (Extract to) введи путь, по которому будет распакован архив. Например, */home/pi/Python_projects/Books_projects*.
- Нажми кнопку **Извлечь** (Extract). Будет создан каталог *Books_projects*, в который архив и распакуется.

Имена и переменные

Имена и переменные играют в программировании очень важную роль. В первой программе мы с понятием переменной уже встречались. В этом же разделе ты по этой теме найдёшь ещё больше важной информации. Для начала попробуй примеры в Python Shell от IDLE.

Простые переменные

Простейшая формула переменных выглядит так:

```
имя = значение
```

- Например, введи в командную строку Python Shell (не в блокнот): x = 12 и нажми клавишу ↵:

```
>>> x = 12
```

Знак равенства не является операцией сравнения, как в математике, а является операцией присвоения переменной x значения 12. Лучше всего прочесть эту инструкцию можно как «Установи значение x, равное 12».

Имя позволит тебе получить доступ к значению переменной.

- Введи в командную строку букву x и нажми клавишу ↵. В командной строке появится введённое ранее значение 12.

```
>>> x
12
```

Это как если бы ты спросил у компьютера, какое значение ранее было присвоено переменной x.

Представь, что число 12 сохранено в переменной x, а переменная для наглядности – это коробка, содержащая цифру.

Имя переменной указано на коробке. Всякий раз при появлении имени x в какой-либо программе компьютер проверяет, что находится в коробке.

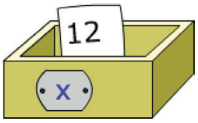


Рис. 7.15. *Переменная x содержит цифру 12*

Если имя уже было использовано в распределении один раз, оно может быть использовано вместо числового значения.

```
>>> x * x
144
```

Значение переменной может быть присвоено другой переменной.

```
>>> y = x
>>> y
12
```

Представить это можно как перенос значения. Переменная y получает то же значение, что и переменная x.

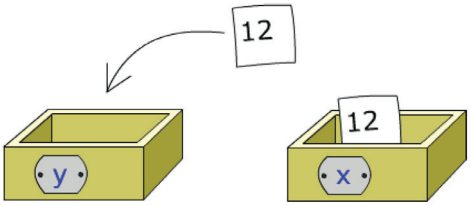


Рис. 7.16. *Перенос значения из «коробки» x в «коробку» y*

Имя переменной можно представить в виде этикетки, наклеенной на цифру или иной объект. Иногда такой пример выглядит убедительнее.

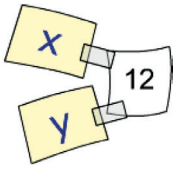


Рис. 7.17. *Функцию распределения можно представить как добавление нового имени к уже имеющемуся*

Назначение с несколькими переменными

В одной инструкции ты можешь использовать несколько имён с одинаковым значением. Например, вместо

```
>>> x = 1
>>> y = 1
```

ты можешь написать

```
>>> x = y = 1
```

Если тебе в одной строке разным переменным нужно присвоить разные значения, ты эту функцию можешь записать следующим образом:

```
>>> x, y = 1, 2
```

Проверь ещё раз:

```
>>> x
1
>>> y
2
```

Какие имена допустимы?

Не всякая последовательность символов может быть использована в качестве имени переменной. Тебе нужно соблюдать несколько правил.

1. Имя переменной не должно быть похожим на команды Python. Кодовые слова программных языков имеют определённые значения, например команда `import`.
2. Имя переменной должно состоять только из букв (от a до z), цифр (от 0 до 9) и нижнего подчёркивания _.
3. Имя переменной должно начинаться с буквы или нижнего подчёркивания.

Это может быть прописная или заглавная буква (`a`, `B`), цифра с нижним подчёркиванием впереди (`_1`). Например, `financial_contribution` или `growth_1`.

Допускается ввод имён переменных на кириллице.

Не допускаются следующие имена:

- `1_цифра` (здесь имя переменной начинается с цифры, перед которой нет нижнего подчёркивания);
- `Respirator-Frequency` (содержит недопустимый символ дефис «-»).

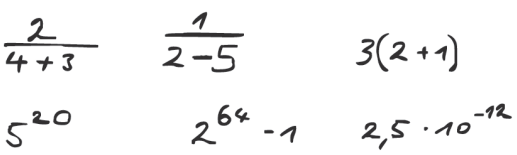
Вопросы

1. Для работы со многими командами блокнота в IDLE существуют горячие клавиши или комбинации клавиш. Если, скажем, ты введёшь комбинацию **Ctrl**+**S**, то программный текст, над которым ты сейчас работаешь, сохранится. Список горячих клавиш расположен в открывающемся меню окна блокнота. Выясни, какая функция присвоена клавише **F5**.
2. Какие недостатки существуют у Python перед Scratch?
3. Какие преимущества есть у Python перед Scratch?
4. Какое отличие в Python между значениями 0 и 0.0?
5. Кто-то однажды сказал: «Вычисление с помощью целых чисел гораздо точнее вычисления чисел с десятичной дробью». Так ли это?

Задания

Задание 1

Напиши выражение Python, способное вычислить рукописные формулы. Протестируй его с помощью Python Shell.



Задание 2

В первом столбце таблицы находятся имена переменных. В остальных столбцах расположены значения переменных *после* выполнения соответствующего присвоения. Дополни таблицу!

Переменные	x	y	z
`x = y = z = 1`	1	1	1
`x = 2* x`	2	1	1
`y = x`			
`z = x = y`			
`y, z = z, y`			
`x = x ** 3 + 1`			
`z // = 2`			

Задание 3. Визуальное распределение

Переменные можно представить в виде коробок. Имя переменной – это этикетка на коробке, а её содержимое – это листок с данными. Руководствуясь рис. 7.18, напиши к этим картинкам подходящие инструкции.

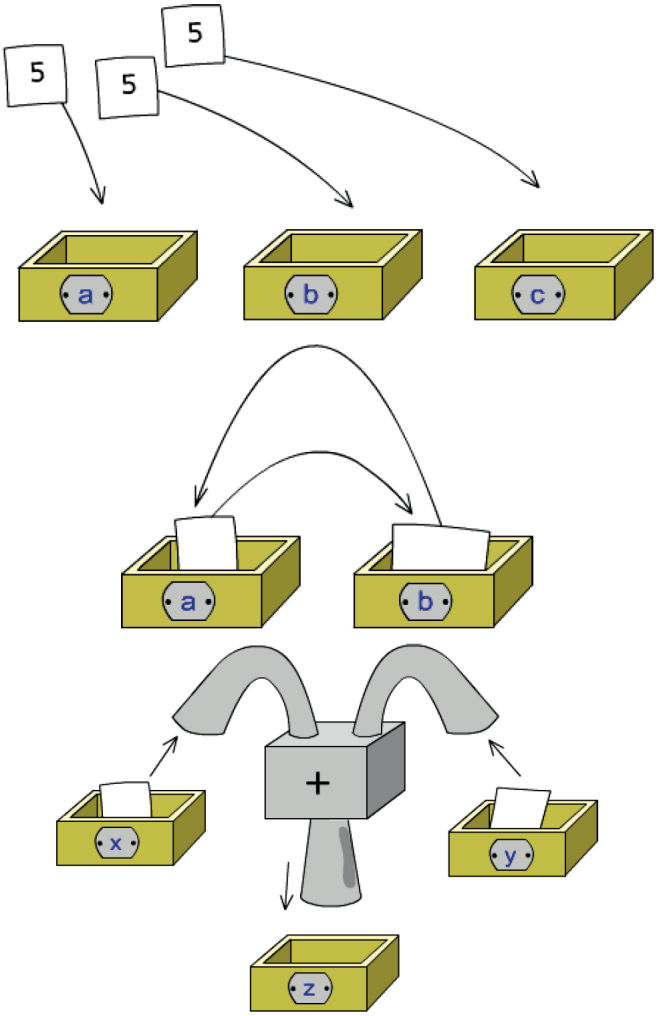


Рис. 7.18. *Картинки с заданиями*

Задание 4. Выражения

Какой результат дадут эти выражения?

Выражение	Результат
`1 + 1`	`2`
`15//2`	
`'2' + '2'`	
`10 * '2'`	
`2+10**2`	
`(2 + 10)**2`	

Задание 5

Есть множество рецептов приготовления теста для пирога. Напиши интерактивную программу, которая высчитывает количество калорий в пироге. Программа будет следовать принципу ввода-вывода и обработки данных. Сначала у пользователя запросят ингредиенты, а затем программа выдаст информацию о содержании калорий в 100 г пирога.

Образец диалога

```
Сколько ингредиентов ты использовал?
Яйца (шт.): 2
Молоко (г): 0
Пшеничная мука (г): 100
Сахар (г): 100
Масло (г): 100
Пирог содержит 420 ккал.
```

Советы

Введённые значения ты можешь сразу преобразовать в цифры. В следующем примере переменная `мука` будет содержать десятичную дробь.

```
flour = float(input('пшеничная мука (г): '))
```

Эту таблицу калорийности ты можешь использовать во время программирования.

Продукты питания	Энергетическая ценность
Куриное яйцо	89 ккал 1 шт.
Молоко (3,5%-ной жирности)	69 ккал в 100 г
Пшеничная мука	360 ккал в 100 г
Сахар	410 ккал в 100 г
Масло	776 ккал в 100 г

Ответы на вопросы

1. Клавиша **F5** означает модуль **Run** (исполнение написанного скрипта).
2. Работая с Python, ты будешь допускать синтаксические ошибки. В работе со Scratch синтактические ошибки ты не сделаешь. В Python важно знать, как построены инструкции. Scratch в виде командных блоков предлагает уже готовые инструкции, которые нужно лишь правильно собрать.
3. Python является полноценным языком программирования и по сравнению со Scratch имеет множество преимуществ. Некоторые из них были описаны в этой главе, например можно делать расчёты с большими цифрами. Python обладает намного большим количеством функций, а ещё можно управлять портами общего назначения GPIO твоего компьютера.
4. Цифра 0 является целым числом (int), а 0.0 – это десятичная дробь (float).
5. Да, это так. Целые числа могут быть сколь угодно длинными, а операции вычисления на целые числа всегда выдают точный результат. Цифры с десятичной дробью имеют максимальное количество разрядов после запятой, и поэтому их точность ограничена. Пример:

```
>>> 1/3
0.3333333333333333
```

Решение задач

Решение 1

```
>>> 2/(4+3)
>>> 1/(2-5)
>>> 3*(2+1)
```

```
>>> 5**20
>>> 2**64-1
>>> 2.5*10**(-12)
```

Решение 2

Переменные	x	y	z
x = y = z = 1	1	1	1
x = 2 * x	2	1	1
y = x	2	2	1
z = x + y	2	2	4
y, z = z, y	2	4	2
x = x ** 3 + 1	9	4	2
x //= 2	4	4	2

Решение 3

```
>>> a = b = c = 5
>>> a, b = b, a
>>> z = x + y
```

Решение 4

Выражение	Результат
1 + 1	2
15//2	7
'2' + '2'	'22'
10 * '2'	'2222222222'
2+10**2	102
(2 + 10)**2	144

Решение 5

```
# Ввод
print('Сколько ингредиентов ты использовал?')
яйца = int(input('яйца (штук): '))                    #1
молоко = float(input('Молоко (г): '))
мука = float(input('Пшеничная мука (г): '))
сахар = float(input('Сахар (г): '))
масло = float(input('Масло (г): '))

# Подготовка
масса = яйца*55 + молоко + мука + сахар + масло       #2
калории = яйца*89 + молоко*69/100 + мука*360/100      #3
калории += сахар*410/100 + масло*776/100              #4
калории_на_100г = round(калории/состав*100)
```

```
# Вывод
print('Пирог содержит', калории_в_100г,
      'ккал на 100 г')  #5
```

Как это работает

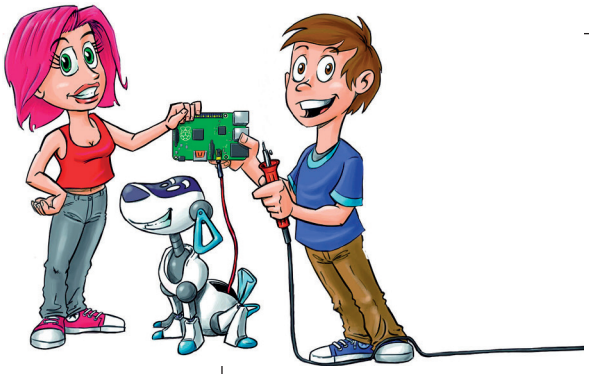
#1 В первую строку переменной яйца введена функция input() (вход) для преобразования символов, введённых в строку яйца (штук), в целое число. Это число сохраняется в переменной. Всем переменным в этой части кода присвоены имена ингредиентов. Для обозначения количества других ингредиентов пользователь может ввести дробное число. Поэтому для преобразования символов в числовые значения с плавающей запятой мы воспользуемся функцией float.

#2 Переменная масса. Здесь рассчитывается общая масса теста. Количество яиц должно быть умножено на массу одного яйца.

#3 В этой части кода (переменная калории) рассчитывается общее количество калорий пирога. Поскольку в таблице задания указано количество калорий из расчёта на 100 г, то указанные данные нужно поделить на 100 (за исключением яиц).

#4 Расчёт общего количества калорий делится на две части.

#5 Вывод данных на экране состоит из трёх частей: текста «Пирог содержит», рассчитанного ранее значения переменной калории_в_100г и ещё одного текста «ккал на 100 г». Все данные выводятся на экран последовательно в одной строке.



8

А что это там мигает? Управляем светодиодами с помощью Raspberry Pi

С помощью Python ты можешь писать программы, которые будут управлять светодиодами. Твой Raspberry, например, сможет автоматически отправлять сигналы азбуки Морзе.

Для этого тебе нужно построить электронную схему и подключить к ней свой Raspberry. Как это сделать, ты узнаешь в этой главе.

Сигналы SOS. Как подавать их с помощью команд Python и светодиодов?

С помощью Raspberry можно управлять разными устройствами. Мы начнём с самого простого и, используя команды Python, заставим осмысленно мигать светодиоды. Прежде чем приступить к программированию, тебе нужно собрать электронную схему.

Для проектов в этом разделе тебе понадобятся следующие электронные комплектующие:

- макетная плата;
- резистор номиналом 130 Ом;
- светодиод;
- перемычки для соединения контактов на макетной плате;
- два гибких кабеля с разъёмами типа «папа-мама». То есть с одной стороны провода припаян штырёк, который можно вставить в контактное отверстие монтажной платы, с другой стороны – разъём с контактным отверстием.

На рис. 8.1 все детали изображены в точной последовательности.

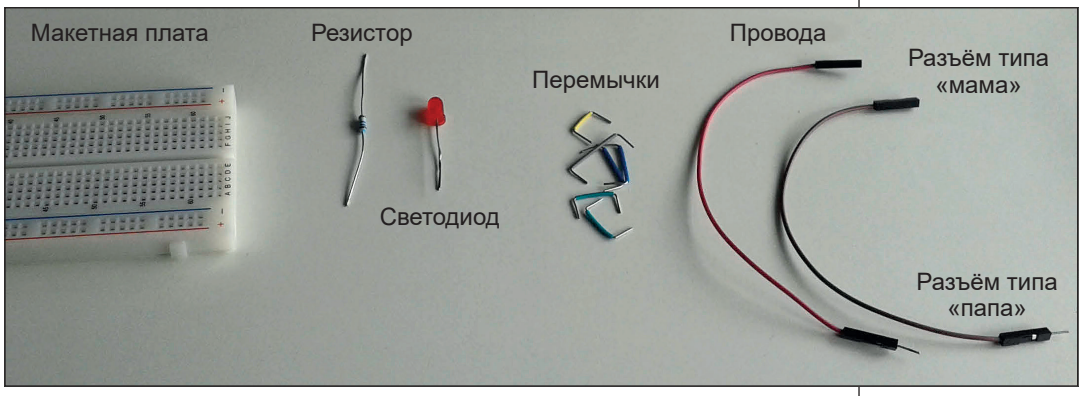


Рис. 8.1. *Необходимые для создания LED-проекта материалы*

В магазинах, торгующих электроникой, или в онлайн-магазинах за очень небольшие деньги ты обнаружишь огромный ассортимент товаров: соединительные провода, светодиоды различных цветов и резисторы различных номиналов. Это дешёвые детали (по нескольку рублей за штуку), особенно если брать их в большом количестве. В этом случае тебе лучше скооперироваться со своими друзьями, которые тоже любят мастерить, и сообща заказать всё, что вам нужно.

Измеряем сопротивление резистора с помощью мультиметра

Резисторы – это, как правило, маленькие детали в форме цилиндра, с правой и с левой сторон которых установлены

контактные выводы. Резистор ограничивает силу проходящего через него тока. Ток – это поток электронов, проходящий через проводник. Проводящий слой резистора в зависимости от номинала детали ограничивает проходящий через него ток. Всё это напоминает камни в ручье, которые замедляют поток воды.

Величина электрического сопротивления измеряется в омах (Ом) и килоомах (кОм). 1 кОм – это 1000 Ом. В зависимости от страны и фирмы-производителя номинал резистора может обозначаться как цифрами и буквами, так и цветными кольцами. В таблице расшифровки ты можешь прочитать номинал каждой комбинации цветных колец. Однако проще всего измерять сопротивление с помощью мультиметра (универсального измерительного прибора). Делается это так.

Подключи к прибору измерительные щупы. Чёрный или синий подключается к разъёму с обозначением COM и значком земли ⏚. Для красного кабеля, в зависимости от того, что тебе нужно измерить, предусмотрено два гнезда. Тебе для измерения сопротивления следует красный кабель подключить к разъёму, обозначенному как VΩmA. Этот разъём предназначен для измерения напряжения (В), сопротивления (Ом) и слабых токов (мА). Назначение другого разъёма, обозначенного как 10 ADC (до 10 А, буквы DC обозначают постоянный ток), – измерение переменного тока силой до 10 А.

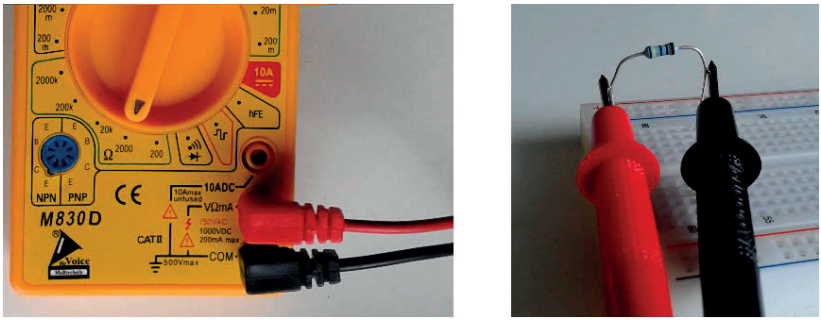


Рис. 8.2. *Измеряем сопротивление с помощью мультиметра*

Теперь выбери диапазон измерения: поверни переключатель режимов так, чтобы стрелка в секторе, обозначенном буквой Ω, остановилась напротив 2000. То есть предел измерений мы выберем о 0 до 2000 Ом (2 кОм) (рис. 8.2 слева).

Коснись щупами выводов измеряемого сопротивления (рис. 8.2 справа). На экране прибора будет показан номи-

нал измеряемой детали. На рис. 8.3 ты увидишь типичные показания мультиметра при выборе диапазона от 0 до 2000 Ом (рис. 8.3 слева), от 0 до 200 Ом, когда номинал детали больше, чем диапазон измерения (рис. 8.3 в центре) и от 0 до 20 кОм, когда номинал детали гораздо меньше, чем измеряемый диапазон.

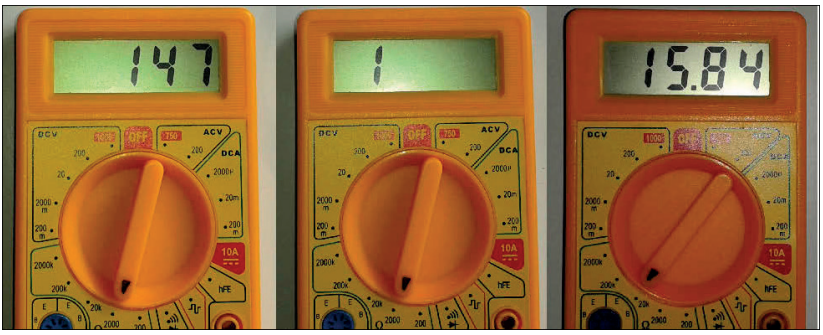


Рис. 8.3. *Показания сопротивления: слева значение 147 Ом, в центре диапазон сопротивления превышен, а справа 15,84 кОм*

Обрати внимание: измерить сопротивление ты можешь не только мультиметром, но и стрелочным тестером. Если у тебя измерительных приборов нет, выбери номинал резистора по нанесённой на его корпус маркировке. Маркировка может быть обозначена цифрами и буквами (на детали должна быть надпись *130* Ω; *130 Ом*; *0,13 кОм*) или по цветовой маркировке. Если номинал обозначен четырьмя цветовыми кольцами, ты увидишь четыре цветных кольца: коричневое, оранжевое, коричневое, золотистое (). Первое кольцо то, которое ближе к контактному выводу. Если номинал обозначен тремя кольцами, ты увидишь кольца следующих цветов: коричневое, оранжевое, коричневое (). При обозначении пятью кольцами ты увидишь следующие цветные кольца: коричневое, оранжевое, чёрное, чёрное, коричневое ().

GPIO (интерфейс ввода-вывода)

На материнской плате Raspberry смонтирован (впаян) разъём расширения портов ввода-вывода, англ. GPIO (*General Purpose Input/Output*). Разъём состоит из 40 позолоченных контактов (пинов). К этим контактам с помощью гибких проводов с разъёмами ты можешь подключить электрическую схему, смонтированную на макетной плате. Распайка разъёмов (расположение контактов) для материнских плат моделей RPi B+, RPi 2 и RPi 3 одинакова.

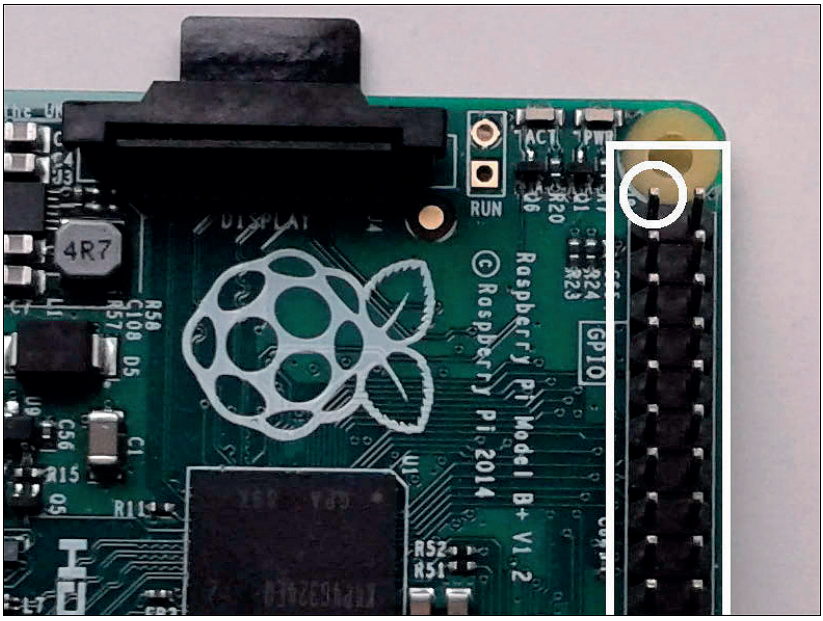


Рис. 8.4. *Разъём расширения портов ввода-вывода Raspberry Pi. Кружком обозначен контакт 1*

На рис. 8.5 видно расположение первых 26 контактов. Здесь указания касательно самых важных из них, которые мы будем использовать всегда.

- Контакт 1 находится слева вверху (на рис. 8.4 он обведён). На этом контакте напряжение +3,3 В.
- Второй контакт находится в том же ряду справа от первого контакта. На этом контакте напряжение +5,0 В. Следи за тем, чтобы никогда этот контакт не был ошибочно закорочен с другим контактом (подключён проводом к другому контакту). Может случиться короткое замыкание, и твой Raspberry будет повреждён.
- Контакт 6 – это масса (этот контакт ещё называют «земля» или «корпус», так как часто этот контакт соединён с корпусом).
- Все 26 контактов свободно программируются и могут быть одинаково использованы в качестве портов ввода и вывода. На каждом из этих контактов присутствует напряжение или 0 или +3,3 В. Во время программирования при напряжении в 0 В на контакте будет присутствовать значение `False` (ложь), а при напряжении 3,3 В появится значение `True` (истина).
- Контакты 3 и 5 обладают ещё одной особенностью – у них имеется нагрузочный резистор. Это значит, что,

если их использовать в качестве портов ввода, к ним будет приложено положительное напряжение. Через резистор в 1,7 кОм они соединяются с плюсовым контактом, на котором присутствует напряжение в +3,3 В. Будь осторожен, когда будешь подключать дополнительную электрическую схему к разъёму расширения портов ввода-вывода GPIO.

- На рис. 8.5 показана распиновка (расположение контактов, их назначение и их значения напряжения) контактов разъёма расширения портов ввода-вывода.

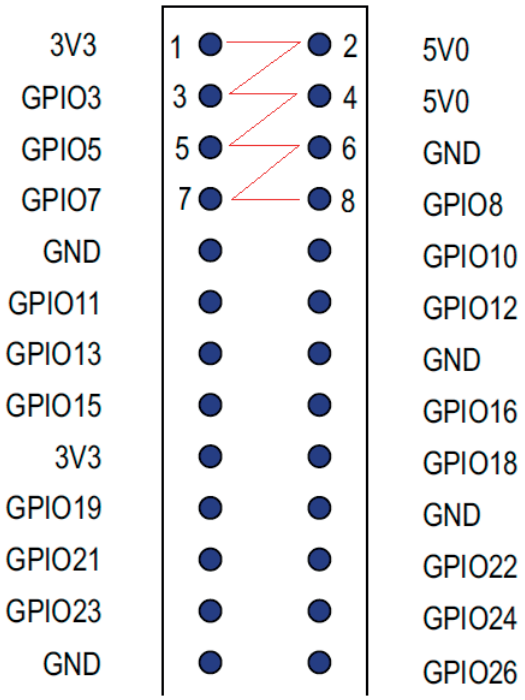


Рис. 8.5. *Распиновка первых 26 контактов разъёма (GPIO)*

Все контакты нумеруются зигзагом. На рис. 8.5 этот условный зигзаг красного цвета. Первый контакт находится слева вверху, справа от него находится контакт 2. Под первым контактом расположился контакт 3 и т. д. Все программируемые контакты носят обозначения GPIO3, GPIO5 и т. д. Цифры в этих обозначениях соответствуют нумерации самих контактов. Такое расположение носит название панели (BOARD). Существуют и другие способы нумерации, но в этой книге мы их использовать не будем.

Подключаем диоды к GPIO

С помощью макетной платы ты сможешь создавать электрические и электронные схемы без пайки, просто вставляя выводы деталей в отверстия макетной платы. Плата состоит из пластика и имеет решётку с отверстиями. В каждом отверстии находится серебряный контакт. В центре платы ты увидишь глубокую прорезь, которая как бы делит плату на две части – правую и левую. На каждой половине по пять вертикальных рядов отверстий с контактами, помеченных буквами A, B, C, D, E и F, G, H, I, J. Горизонтальные ряды отмечены цифрами. Важно: на каждой половине каждый ряд горизонтальных отверстий соединён проводниками. То есть, например, контактные отверстия A1, B1, C1, D1 и E1 между собой соединены, но с отверстиями A2, B2, C2, D2 и E2 у них контакта нет. И так каждый горизонтальный ряд отверстий (рис. 8.6). Справа и слева по краям макетной платы находятся два вертикальных ряда отверстий. Каждый вертикальный ряд отверстий соединён между собой проводником. Эти контакты (их ещё называют «шины») предназначены для подключения к ним напряжения питания.

Рис. 8.6. *Так выглядит макетная плата снизу, если снять защитное покрытие. На каждой половине платы каждый ряд контактных отверстий A, B, C, D, E и F, G, H, I, J соединён проводником*

На рис. 8.7 показано, как выглядит собранная схема. Давай шаг за шагом рассмотрим, как собрать электрическую схему.

Сначала нам нужно подать на макетную плату электрическое питание напряжением +3,3 В. Это напряжение мы возьмём с контакта 1 разъёма расширения портов ввода-вывода. Положительное напряжение (+3,3 В) подаётся

на вертикальную группу контактов макетной платы, помеченных красной полосой и значком «+».

- Подключи гибким проводом контакт 1 разъёма на материнской плате с вертикальной группой контактов макетной платы, как показано на рис. 8.7.
- Соедини гибким проводом вертикальный ряд макетной платы, отмеченный синей полосой, с контактом 10 разъёма расширения портов ввода-вывода, как на рис. 8.7.
- Вставь светодиод выводами в макетную плату, как на рис. 8.7 слева. Обрати внимание, в какое отверстие будет вставлен короткий вывод. Короткий вывод – это катод. Катод светодиода соедини перемычкой с вертикальным рядом, отмеченным синей полосой и значком «–». Вывод катода светодиода немного короче, чем анодный вывод. Если перепутать выводы (полярность питания светодиода) и на анод подать минус, а на катод – плюс, светодиод светиться не будет.

Мнемоническое правило, или запоминалка: катод – короткий.

- Анод светодиода соедини с группой контактов «+» через резистор номиналом 130 Ом и через перемычку (рис. 8.7).

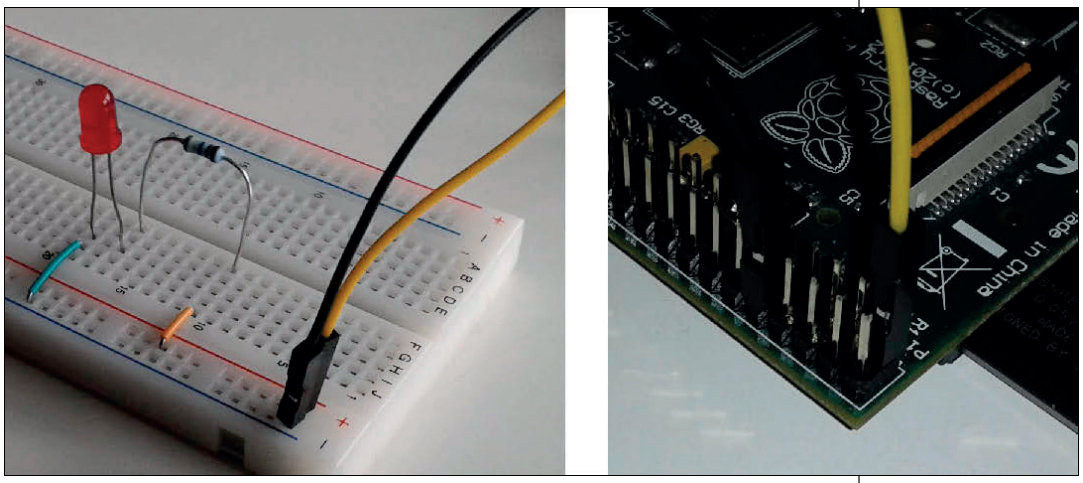


Рис. 8.7. *Схема подключения и соединения к плате расширения (GPIO)*

Схема подключения показана на принципиальной схеме (рис. 8.8).

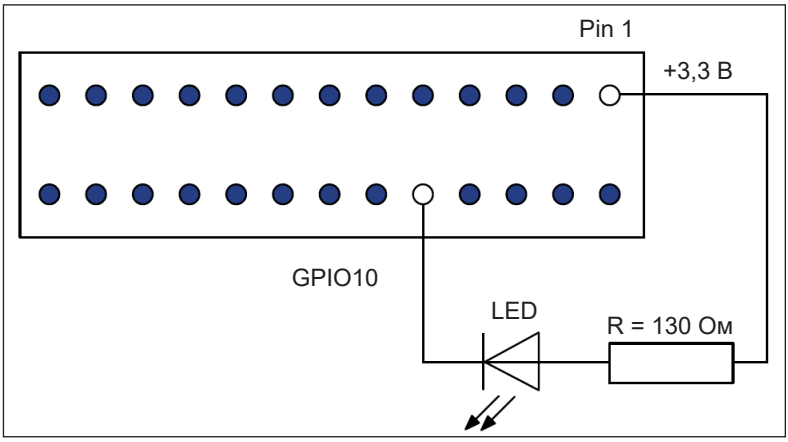


Рис. 8.8. *Принципиальная схема подключения светодиода к контактам разъёма расширения портов ввода-вывода GPIO. Катод светодиода направлен влево*

Впрочем, если у тебя макетной платы нет, ты просто можешь подпаять резистор к аноду светодиода, резистор подключить к первому контакту разъёма расширения портов ввода-вывода, а катод светодиода – к контакту 10 разъёма.

Для чего светодиоду нужен дополнительный резистор?

Светодиоды (англ. *light emitting diode* (LED) – «светодиод») – это полупроводниковый элемент, который ведёт себя как диод и при этом ещё и излучает свет. Диод – это клапан для электронов, в одну сторону ток пропускающий, а в другую нет. С помощью мощных диодов переменный ток преобразуется в постоянный. У светодиода имеется два контактных вывода: *анод* и *катод*. Постоянный электрический ток протекает от анода к катоду. То есть, чтобы пропустить электрический ток через светодиод, тебе нужно к *аноду* (более длинный вывод) подвести напряжение положительной полярности, к *катоду* (вывод покороче) – напряжение отрицательной полярности. Если к аноду приложить напряжение отрицательной полярности, а к катоду – положительной, диод закроется и ток через себя пропускать не станет. Из уроков физики ты, наверное, знаешь, что ток всегда течёт от положительного полюса к отрицательному. Направление движения тока через диод показано изображением в виде стрелки. Вертикальная чёрточка на конце этой стрелки показывает, что обратное прохождение электронов закрыто.

Каждый радиоэлектронный элемент рассчитан на работу с определённым напряжением, приложенным к его выво-

дам. Если это напряжение превысить, радиоэлектронный элемент выйдет из строя (сгорит). Каждый активный радиоэлектронный элемент (светодиод, транзистор, микросхема) имеет свой потребляемый ток, определяемый его внутренним сопротивлением. Не путай два показателя: напряжение и ток. Напряжение, прикладываемое к выводам, и потребляемый элементом ток зависят только от его конструкции, и для каждого отдельного элемента эти характеристики нужно смотреть в справочниках по радиоэлементам. Светодиод потребляет ток не более 10 мА, а напряжение, прикладываемое к его выводам, 2 В. Учитывая, что напряжение питания, подаваемое с материнской платы, 3,3 В, нам следует напряжение погасить на 1,3 В. Это мы сделаем с помощью пассивного радиоэлектронного элемента под названием *резистор*. Часть «лишнего» напряжения на резисторе будет преобразована в тепло, остальное напряжение будет подано на вывод светодиода. Учитывая, что ток светодиод потребляет маленький, нагрев резистора практически ощущаться не будет. На резисторе в виде тепла будет рассеяна мощность 0,013 Вт.

Номинал нужного нам сопротивления рассчитывается следующим образом. Согласно закону Ома, сопротивление равно напряжению, которое нужно погасить, делённому на силу тока, проходящего через это сопротивление: $R = U/I$. Мы уже ранее говорили, если напряжение питания, снимаемое с первого контакта разъёма расширения портов ввода-вывода 3,3 В, напряжение питания светодиода 2 В, нам нужно погасить 1,3 В. Ток потребления светодиода 10 мА. Из этого следует, что сопротивление резистора должно быть 1,3 В / 0,01 А = 130 Ом.

После того как схема собрана и подключена к контактам разъёма расширения ввода-вывода, нам нужно настроить контакт 10.

Управление светодиодами. Модуль RPi.GPIO

Теперь перейдём к программированию. Доступ к интерфейсу ввода-вывода ты можешь получить, только обладая правами «суперпользователя» (администратора).

- Открой LXTerminal и запусти Python Shell с помощью команды

```
$ sudo idle3
```

Подготовка

Сначала проведём несколько экспериментов и попробуем управлять светодиодами вручную с помощью одной-единственной команды в Python Shell.

Импортируй модуль `RPi.GPIO`. Обрати внимание на написание маленькой буквы «i» в слове RPi.

```
>>> from RPi import GPIO
```

С помощью функции `setmode()` можно определить, с каким типом нумерации контактов мы сейчас работаем. Настроим модус `Board`.

```
>>> GPIO.Setmode(GPIO.BOARD)
```

Сделай канал 10 выходным каналом.

```
>>> GPIO.Setup(10, GPIO.OUT)
```

В этот момент светодиод начнёт светиться, так как контакт 10 будет подключён к массе. Это состояние `False`. Таким образом, напряжение между контактами 1 и 10 будет равным 3,3 В. Так как светодиод с сопротивлением подключен к этим контактам, то, если полярность светодиода соблюдена, ток потечёт через эту цепь и светодиод начнёт светиться.

Таблица 8.1. Истинное значение напряжения и состояние контактов

Истинное значение	Напряжение
`False`	0 В
`True`	+3,3 В

Измеряем напряжение

Как уже говорилось ранее, на светодиоде должно быть напряжение, равное 2 В. Ты можешь это легко проверить. Настрой свой мультиметр на режим измерения постоянного напряжения до 20 В (рис. 8.9). Обозначение *DCV* на приборе расшифровывается как *Direct Current Voltage* (напряжение постоянного тока), что значит постоянный ток.

Теперь, когда светодиод излучает свет, прикоснись щупами мультиметра к его выводам. На экране прибора ты увидишь, какое в данный момент подаётся напряжение на

светодиод. Учитывая, что номинал резистора всегда имеет небольшой разброс (в пределах 5 %), попробуй подобрать номинал резистора, взяв, например, деталь номиналом 140 Ом, если напряжение на светодиоде немного больше. Замени резистор в схеме и повтори измерение. Напряжение в светодиоде слегка понизится.

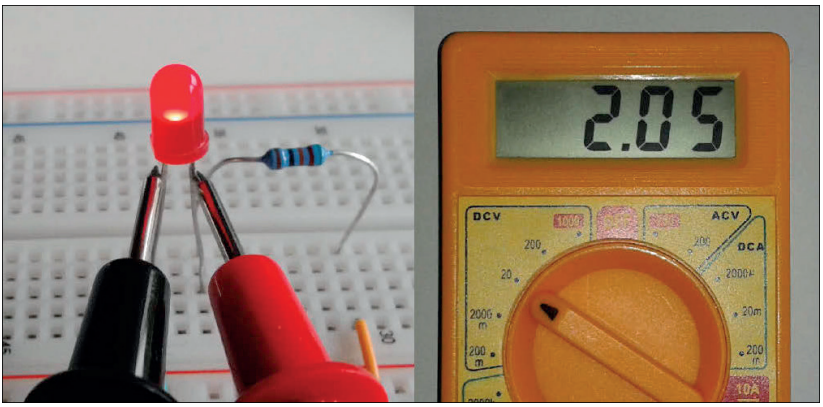


Рис. 8.9. *Напряжение в светодиоде составляет 2,05 В. Справа анод (положительный вывод), слева – катод (отрицательный вывод)*

Включаем и выключаем светодиод

Теперь выключим светодиод.

- Выключи светодиод, используя команду `GPIO.output(10, True)`.

```
>>> GPIO.output(10, True)
```

Теперь контакт 10 будет отключен от массы, и напряжение на нем будет таким же, как и на контакте 1, т. е. 3,3 В. Они оба имеют одинаковый электрический потенциал по отношению к корпусу. Между ними нет разницы напряжения, и поэтому ток через светодиод не течёт.

- Определи на контакте 10 назначение `False`:

```
>>> GPIO.output(10, False)
```

Теперь контакт 10 снова подключится к массе, через светодиод потечёт ток, и светодиод начнёт светиться.

С помощью команды `cleanup()` ты можешь одним махом отменить все настройки GPIO (англ. *to clean up* – «почистить, убраться»). Попробуй:

```
>>> GPIO.cleanup()
```

Контакт 10 перестанет быть каналом вывода и отключится от корпуса. Следовательно, и светодиод светиться перестанет.

Таблица 8.2 даёт обзор важнейших команд модуля GPIO.

Таблица 8.2. Важнейшие функции модуля RPi. GPIO

Команда	Пояснение
`cleanup()`	Отмена всех настроек
`input(pin)`	Вместо слова `pin` в скобках ты указываешь номер контакта, например 10. Функция проверяет состояние контакта. Если на нём 3,3 В, функция возвращает значение `True`. Если 0 В, то возвращает значение `False`
`output(pin, state)`	Контакт GPIO с помощью своего номера перемещается в состояние state (`True` или `False`). Вместо `state` тебе придётся писать `True` или `False`. При значении `True` контакт получает напряжение мощностью 3,3 В. При значении `False` этот показатель равен нулю
`setmode(mode)`	Устанавливает модус, в котором нумеруются контакты GPIO. В этой книге мы используем лишь аргумент GPIO. BOARD
`setup(pin, mode)`	Первый аргумент – это номер контакта. Вместо `pin` ты вписываешь его номер (например, 10). Если режим `mode` – это константа GPIO.IN, то контакт становится входным каналом. Если режим `mode` – это константа GPIO.OUT, то контакт становится выходным каналом

Проект 18. Программируем сигнал SOS

С помощью световых сигналов можно передавать информацию. В азбуке Морзе буквы и цифры кодируются путём коротких и длинных сигналов. Самой известной комбинацией букв является международный сигнал бедствия SOS: три коротких сигнала – три длинных – три коротких. На письме выглядит так: ... – – – ... . Длинный сигнал (тире) в три раза длиннее короткого (точки). Пауза после каждого сигнала по длительности соответствует точке.

Вопреки расхожему мнению последовательность букв SOS выбрана из-за удобства передачи этого сигнала (три корот-

ких, три длинных, три коротких). Есть, как минимум, две расшифровки этого сочетания букв, но эти расшифровки были придуманы после создания самого сигнала SOS.То есть ответ подогнан под результат. Вот наиболее часто употребляемые фразы, придуманные для расшифровки этого сигнала (фразы, очень близкие к настоящему значению этого сигнала): **Save Our Ship** (спасите наш корабль), **Save Our Souls**, **Save Our Spirits** (спасите наши души), **Swim Or Sink** (плывите или утонем) или даже **Stop Other Signals** (прекратите другие сигналы). В последней фразе имеется в виду, что все остальные сигналы должны быть прекращены, чтобы не мешать терпящему бедствие кораблю (радисту корабля) передавать свои координаты.

Программа Python передаёт SOS, включая и выключая светодиод. Для этого проекта ты воспользуешься принципиальной схемой, показанной на рис. 8.8. Тебе нужно лишь записать первые строчки, потом ты можешь их скопировать, выделить этот абзац жирным шрифтом и при необходимости видоизменять его. Функция копирования в IDLE такая же, как и в любом другом редакторе.

- Установи курсор в начало строки, которую нужно скопировать.
- Нажми и удерживай левую кнопку мыши.
- Удерживая левую кнопку мыши нажатой, перемести курсор в конец строки. Когда строка выделится, левую клавишу мыши отпусти.
- Набери комбинацию **Ctrl**+**C**. Буква C означает *копировать* (англ. *copy*). При этом текст копируется в буфер обмена компьютера.
- Нажми клавишу ↵, чтобы перейти на новую строку.
- Введи комбинацию клавиш **Ctrl**+**V**. Выделенный текст появится в новой строке.
- Если ещё раз нажать комбинацию клавиш **Ctrl**+**V**, содержимое буфера обмена будет вставлено в месте нахождения курсора.

```
#s
GPIO.output(10,False)
time.sleep(0.2)
GPIO.output(10,True)
time.sleep(0.2)
```

Рис. 8.10. *Так выглядит кусочек программного текста, который мы выделили*

Комментарии #1, #2, #3, #4 ты можешь пропустить. Я добавил их лишь затем, чтобы объяснить последующие строки.

```
from RPi import GPIO
import time

#GPIO подготовка
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.OUT)

#S
GPIO.output(10, False)          #1
time.sleep(0.2)                 #2
GPIO.output(10, True)           #3
time.sleep(0.2)

GPIO.output(10, False)
time.sleep(0.2)
GPIO.output(10, True)
time.sleep(0.2)

GPIO.output(10, False)
time.sleep(0.2)
GPIO.output(10, True)
time.sleep(0.2)

#O
GPIO.output(10, False)
time.sleep(0.6)                 #4
GPIO.output(10, True)
time.sleep(0.2)

GPIO.output(10, False)
time.sleep(0.6)
GPIO.output(10, True)
time.sleep(0.2)

GPIO.output(10, False)
time.sleep(0.6)
GPIO.output(10, True)
time.sleep(0.2)

#S
GPIO.output(10, False)
time.sleep(0.2)
GPIO.output(10, True)
time.sleep(0.2)
```

```
GPIO.output(10, False)
time.sleep(0.2)
GPIO.output(10, True)
time.sleep(0.2)

GPIO.output(10, False)
time.sleep(0.2)
GPIO.output(10, True)
```

Как это работает

#1 Контакт 10 переходит в состояние `False`. Контакт подключится к массе, ток от контакта 1 через резистор и светодиод потечёт к контакту 10, и светодиод начнет светиться.

#2 Функция `sleep()` (сон) тайм-режима гарантирует, что во время выполнения программа будет приостановлена на некоторое время. В скобках указано количество секунд ожидания. Время ожидания компьютера составляет 0,2 с. На это время контакт 10 будет подключен к массе, и светодиод будет светиться.

#3 Контакт 10 от массы отключится (состояние `True`). Светодиод погаснет, но он светил в течение 0,2 с.

#4 При длинном сигнале светодиод остаётся включённым 0,6 с.

Вопросы

1. Какой особенной электрической характеристикой обладает светодиод?
2. По каким признакам ты опознаешь катод светодиода?
3. С какой полярностью источника тока (положительной или отрицательной) катод должен быть подключён к светодиоду?
4. Зачем светодиоду требуется дополнительный резистор?
5. Как коммутируются контакты GPIO в состоянии `True`?
6. Какой контакт постоянно подключён к массе?
7. На каком контакте постоянно присутствует напряжение в 3,3 В?

Задания. Создай два мигающих светодиода

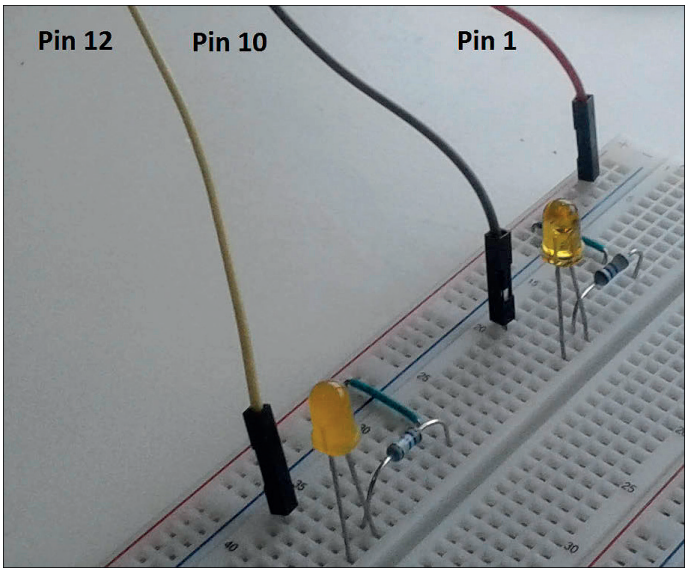


Рис. 8.11. *Два светодиода на съёмной плате. Катоды (с более короткими выводами) находятся слева*

Напиши программу, которая будет управлять двумя светодиодами. Мигать они должны в последовательности, показанной на рис. 8.12. Сначала оба светодиода выключены. Далее загорается нижний светодиод. Следующий такт – оба светодиода выключены. Ну и т. д.

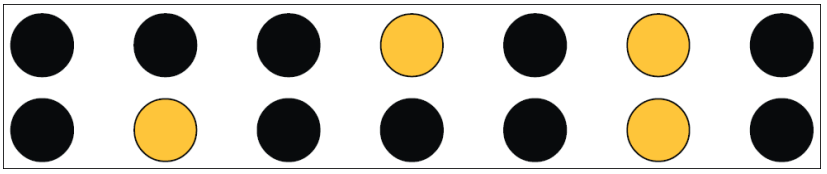


Рис. 8.12. *Последовательность мигания светодиодов*

На рис. 8.13 показана принципиальная схема подключения двух светодиодов к разъёму расширения портов ввода-вывода компьютера. Питаются оба светодиода от контакта 1. Катоды светодиодов подключаются к контактам 12 и 10.

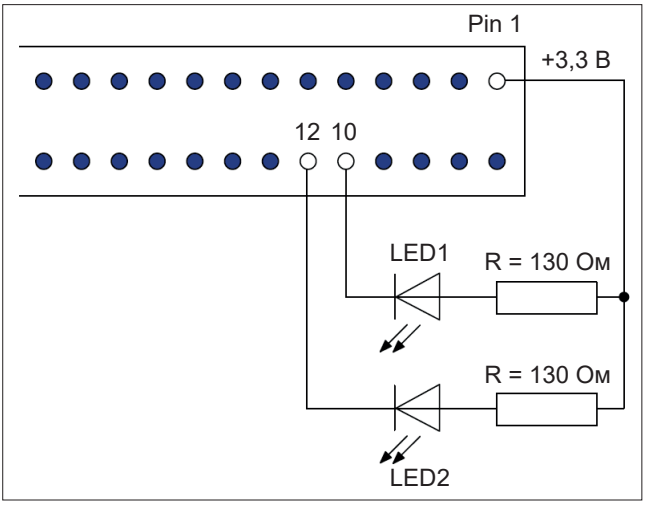


Рис. 8.13. *Схема подключения двух светодиодов к контактам разъёма*

Ответы на вопросы

1. Диод действует как клапан для электронов. Благодаря ему ток течёт в одном направлении.
2. На стороне катода основание светодиода имеет прямой край, а контактный вывод короче. Мнемоническое правило (читай: подсказка): *катод = край = короткий*.
3. Катод соединяется с массой (корпусом).
4. Резистор за счёт своего резистивного электропроводящего слоя понижает напряжение, подаваемое на нагрузку (светодиод).
5. В состоянии `True` контакты отключены от массы (корпуса). В состоянии `False` контакт подключается к массе.
6. Контакт 6 постоянно подключён к массе (корпусу).
7. На контакте 1 всегда присутствует напряжение +3,3 В.

Ответы на задания

Программа

```
from RPi import GPIO          #1
import time

GPIO.setmode(GPIO.BOARD)      #2
GPIO.setup(10, GPIO.OUT)      #3
```

```
GPIO.setup(12, GPIO.OUT)

GPIO.output(10, True)          #4
GPIO.output(12, True)
time.sleep(0.5)                #5

GPIO.output(12, False)         #6
time.sleep(0.5)
GPIO.output(12, True)          #7
time.sleep(0.5)

GPIO.output(10, False)         #8
time.sleep(0.5)
GPIO.output(10, True)          #9
time.sleep(0.5)

GPIO.output(12, False)         #10
GPIO.output(10, False)
time.sleep(0.5)
GPIO.output(12, True)          #11
GPIO.output(10, True)
```

Как это работает

#1 Импорт модуля GPIO и time-модуля.

#2 Настройка коммутации контактов. Эта настройка всегда выполняется на панели (Board).

#3 GPIO-контакты 10 и 12 настраиваются в качестве каналов вывода.

#4 Контакты 10 и 12 переводятся в состояние `True`. В этом случае контакты отключены от массы, ток не течёт, и светодиоды не светятся.

#5 Программа в режиме ожидания находится 0,5 с.

#6 Контакт 12 переводится в состояние `False`, подключается к массе. Ток начинает течь, и светодиод излучает свет.

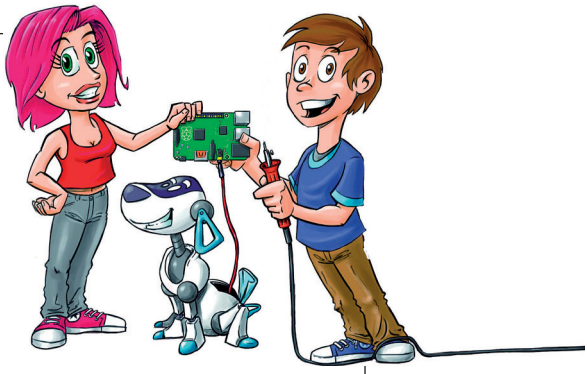
#7 Контакт 12 от массы отключён. Нижний светодиод гаснет.

#8 Контакт 10 подключается к массе. Ток начинает течь, и светодиод загорается.

#9 Верхний светодиод снова выключен.

#10 Включены оба диода.

#11 Выключены оба диода.



9 Компьютер принимает решения

В этой главе ты познакомишься с ещё несколькими техниками Python – оператором ветвления `if-else` (если–иначе) и оператором цикла `while` (пока выполняется условие).

Используя операторы `if-else`, твой проект, написанный на языке Python, сможет выбрать решение из нескольких предложенных вариантов. Ты напишешь программу, способную, исходя из наблюдений, распознать вид пластмассы.

А с помощью оператора цикла (`while`) твой светодиод будет мигать в заданном ритме в нужное тебе время.

Оператор ветвления

По работе со Scratch ты знаешь – программы могут содержать ветвления. То есть если мы получили одни данные, с помощью оператора ветвления программа выбирает один вариант выполнения работы. Если мы получаем другие данные, выбирается другой вариант работы программы. Таким образом, для выбора варианта программы требуется выполнение определённого условия.

Условный оператор с одной ветвью (if)

Самым простым ответвлением программы является условная инструкция, или инструкция `if` (если). Если условие выполнено (`True`), выполняется совокупный блок инструкций. Если условие не выполнено, ничего не происходит.

```
if условие:
    блок инструкций
```

Блок инструкций может состоять из нескольких операторов. Важно, чтобы все инструкции были отделены друг от друга равным количеством пробелов.

Следующая программа запрашивает имя и, если это, например, Александр, его приветствует. В остальных случаях (если пользователя зовут, например, Том, Сергей, Ваня или Боб) ничего не происходит.

Выполнение программы

Программа выполняет следующее.

```
Имя: Александр
Привет, Александр!
Добро пожаловать!

Name = input('имя: ')
if name == 'Tom'                        #1
    print('Привет, Том!')               #2
    print('Добро пожаловать!')
```

Выбери команду меню  ⇨ **Программирование** ⇨ **Python 3 (IDLE)**. Будет запущена оболочка *Python Shell*.

Выбери команду меню **File** ⇨ **New File** (Файл ⇨ Новый файл). Откроется блокнот.

Выбери команду меню **File** ⇨ **Save As** (Файл ⇨ Сохранить как) и сохрани проект под именем, например, *Hello_Tom.py*.

Введи в блокнот текст, показанный ниже программы, соблюдая знаки препинания и пробелы. Заметь, после оператора `if name == 'Tom'` для следующих двух операторов отступы от левого края сформируются автоматически.

```
name = input('Имя:')
if name == 'Tom!'
    print('Привет, Том!')
    print('Добро пожаловать!')
```

Обрати внимание, имя нужно писать латиницей. Если будут символы на кириллице, программа работать не будет!

Нажми комбинацию клавиш **Ctrl**+**S**, после чего нажми клавишу **F5** и проверь работу программы.

Дальше, чтобы сэкономить место в книге, мы процедуру запуска оболочки Python, открытия блокнота, процесс введения программы, её сохранения и проверки работы кода описывать не будем. При необходимости ты это будешь делать сам.

Как это работает

#1 После введения оператора `if` задаётся условие, которое необходимо выполнить. В данном случае условие – это имя Tom, введённое через двойной знак равенства `==`. Это имя сравнивается с именем, введённым в первую строку `name = input('Имя: ')`. Два знака равенства являются операторами сравнения. Обрати внимание на двоеточие после условия! Это двоеточие должно быть обязательно введено.

#2 Если условие выполнено, исполняются две связанные инструкции: `print('Привет, Том!')` и `print('Добро пожаловать!')`. Если условие не выполняется, эти две инструкции игнорируются, и работа программы останавливается.

Условный оператор с двумя ветвями (if-else)

Оператор с двумя ветвями, или, по-другому, инструкция `if-else`, – это решение с двумя условиями. Если выполняется условие 1, выполняются действия А. При выполнении условия 2 выполняется действие Б. К примеру, если светит солнце, я пойду играть в футбол, а если нет – поиграю в настольный теннис.

При выполнении оператора `if-else` есть *два* блока инструкций: выражение `if` и выражение `else`.

- Выражение `if` (если) (допустим, играть в футбол) выполняется при соблюдении условия (например, светит солнце).
- Выражение `else` (иначе) (скажем, играть в настольный теннис) выполняется, если условие не соблюдено (солнце не светит).

```
if Условие:
    Блок инструкций 1
else:
    Блок инструкций 2
```

Следующая программа запрашивает возраст пользователя и, исходя из этого, решает, как к нему обращаться – на *ты* или на *вы*.

Ход программы 1 (условие выполнено)

```
Возраст (лет): 15
Что я могу для тебя сделать?
```

Ход программы 2 (условие не выполнено)

```
Возраст (лет): 19
Что я могу для вас сделать?
```

Программа

```
возраст = int(input('возраст (лет): '))        #1
if возраст < 18:                               #2
    print('Что я могу для тебя сделать? ')     #3
else:                                          #4
    print('Что я могу для вас сделать? ')      #5
```

Как это работает

#1 Компьютер ожидает ввода с клавиатуры. Знаки, которые вводятся с помощью функции `int()`, преобразовываются в целое число. Это число назначается переменной. Имя переменной – возраст. Ранее мы уже обсуждали, что имена переменных можно вводить кириллицей. Главное, чтобы в пределах кода написание имени переменной было полностью одинаковым.

#2 После оператора `if` находится имя переменной *возраст*, в которой хранится введённое значение возраста. Это значение сравнивается с условием (значение переменной – возраст меньше 18) , и, если это условие выполняется, программа переходит к выполнению связанной с этим условием строки кода (инструкции): `print('Что я могу для тебя сделать? ')`. Обрати внимание: после условия обязательно должно стоять двоеточие.

#3 Инструкция `print('Что я могу для тебя сделать? ')` вводится с отступом от левого края.

#4 Следи, чтобы после команды `else` находилось двоеточие! Отступ от левого края перед командой `else` отсутствует!

#5 Эта строка кода (инструкция) вводится с отступом от левого края. Данная строка кода выполняется, если введённое значение переменной возраст больше или равно 18.

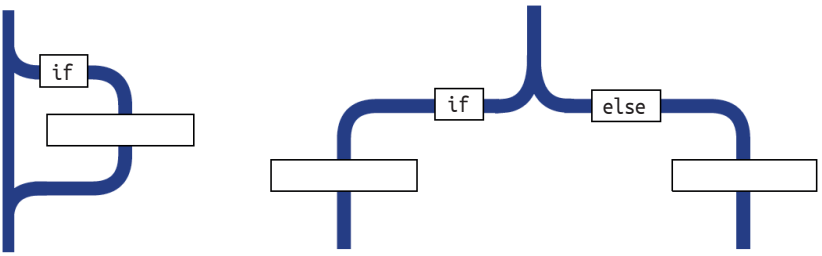


Рис. 9.1. *Пример того, как выглядит условный оператор с одной и двумя ветвями*

Проект 19. А что это за пластик?

Большое количество упаковки сегодня состоит из пластика: пакеты, бутылки, баночки из-под йогурта и т. д. Для того чтобы это всё более качественно утилизировать, на упаковке стоит краткая маркировка: ПЭ (полиэтилен), ПП (полипропилен), ПС (полистирол), ПЭТФ (полиэтилентерефталат). Но узнать материал упаковки можно и путём маленького эксперимента. Для этого мы напишем одну программу.

```
Вставь кусочек пластика в стакан с водой. Он плавает? (д/н) д
Сожги на улице (не в квартире) маленький кусочек материала, держа
его на поддоне. Что ты видишь при этом?
1: При горении он сильно чадит.
2: Сгорая, он пахнет свечным воском.
3: Другие наблюдения.
Наблюдения (1, 2 или 3): 2
Результат: речь идет о ПЭ
```

Пишем программу

```
# Ввод
print('Помести кусочек пластика в стакан с водой.')
плавать = input('Он плавает? (д/н) :')                    #1
print('Сожги на улице (не в квартире)',
      'маленький кусочек материала ',
      'держа его на поддоне. Что ты при этом видишь?')    #2
print('1: При горении он сильно чадит.')                  #3
print('2: Сгорая, он пахнет свечным воском.')
print('3: Другие наблюдения.')
гореть = input('Наблюдения (1, 2 или 3):')                #4

# Обработка
if плавать == 'д':                                        #5
    if гореть == '2':                                     #5
```

```
        материал = 'ПЭ'                                    #6
    else:                                                  #7
        материал = 'ПП'
else:
    if гореть == '1':
        материал = 'ПС'
    else:
        материал = 'ПЭТФ'

# Вывод
print ('Результат: речь идет о',материал)                  #8
```

Так как в данной программе очень важны отступы, на рис. 9.2 ты увидишь снимок с экрана, где в блокноте введён код этой программы. Если отступы ты расставишь неправильно, программа работать не будет. Обрати внимание: отступы очень удобно расставлять с помощью клавиши ⇥.

```
# Ввод
print('Помести кусочек пластика в стакан с водой.')
плавать=input('Он плавает? (д/н) :')
print('Сожги на улице (не в квартире)',
      'маленький кусочек материала',
      'держа его на металлическом поддоне. Что ты при этом видишь?')
print('1: При горении он сильно чадит.')
print('2: Сгорая, он воняет старым воском.')
print('3: Другие наблюдения.')
гореть=input('Наблюдения (1, 2 или 3):')

# Обработка
if плавать == 'д':
    if гореть == '2':
        материал = 'ПЭ'
    else:
        материал = 'ПП'
else:
    if гореть == '1':
        материал = 'ПС'
    else:
        материал = 'ПЭТФ'

# Вывод
print ('Результат: речь идёт о', материал)
```

Рис. 9.2. *Код программы введён в блокнот*

Как это работает

#1 Информация, которая стоит за скобками после `input`, – это строка, появляющаяся на экране. Переменная `плавать` сохраняет значения («д» или «н»), которые пользователь ввёл.

#2 На экране отображается текст, который вписывается в три строки.

#3 С оператора `print()` начинается новая строка. Из этой строки мы для дальнейшей обработки используем

цифровые значения 1, 2 или 3, вводимые с клавиатуры. Текст только объясняет, какую клавишу при определённом варианте нужно нажать.

#4 Вводя с клавиатуры нужное цифровое значение, ты присваиваешь это значение переменной `гореть`.

#5 Начало вложенного цикла ветвления `if-else`.

#6 Переменная `материал` содержит сокращение марки материала этого пластика.

#7 Оператор `else` должен быть напечатан точно под соответствующим оператором `if`. То есть `if` и относящийся к нему оператор `else` должны располагаться на одинаковом расстоянии от левого края (равном количестве отступов).

#8 Эта строка выводит на экран результат. Вывод состоит из двух частей: «`Результат: речь идёт о`» и значения переменной `материал`.

Программа содержит одну вложенную инструкцию `if-else`. Рисунок 9.3 наглядно демонстрирует ветвление хода программы.

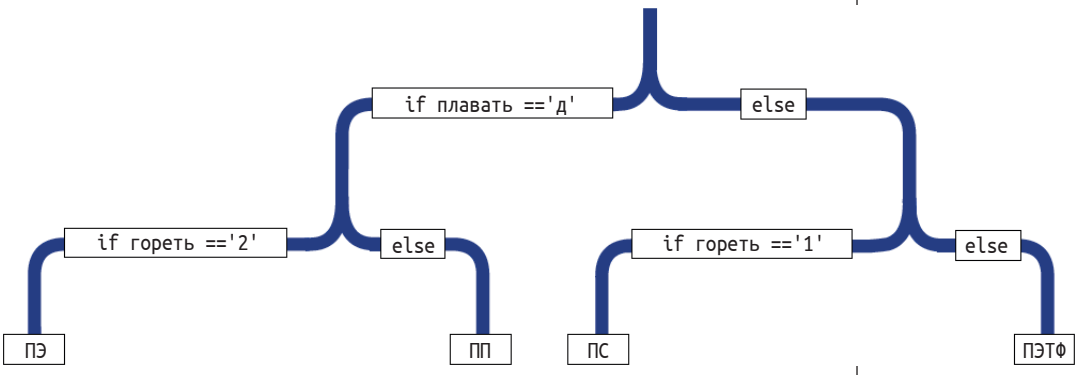


Рис. 9.3. *Вложенные инструкции `if-else` можно проиллюстрировать с помощью дерева выбора*

Обрати внимание на структуру программных блоков (рис. 9.2)! В каждом из них инструкции расположены с одинаковым количеством отступов.

Важно!

Каждый оператор `else` должен находится прямо под соответствующим оператором `if` (рис. 9.2). Оба кода имеют одинаковое количество пробелов при вводе.

Условия

Каждая ветвь программы содержит условие. Условие – это логическое выражение, которое может быть как истинным (`True`), так и ложным (`False`). Большинство условий похоже на условия в предыдущих примерах для инструкций `if`.

В интерактивном модусе ты можешь ввести условие без инструкции `if` и с помощью интерпретатора проанализировать его. Результат будет либо `True`, либо `False`.

```
>>> 1== >1
True
>>> 1== 2
False
>>> 1/3 == 2/6
True
```

Обрати внимание, что оператор сравнения состоит из двух знаков равенства. Обычный знак равенства «=» ты используешь в случае, если присваиваешь значение переменной. В таблице ниже ты найдёшь обзор всех важнейших операторов сравнения.

Таблица 9.1. Операторы сравнения Python

Оператор	Пояснение	Пример	Истинное значение
<	меньше	10 < 20 10 < 10	True False
<=	меньше или равно	10 <= 20 10 <= 10	True True
>	больше	5.0 > 5.0	False
>=	больше или равно	5 >= 6	False
==	равно	1 == 1.000	True
!=	не равно	2 != 3	True

Для условий у Python имеется тип данных `bool`. Попробуй:

```
>>> type(2 > 1)
<class 'bool'>
```

Истинные значения чисел

Отличительной особенностью Python является то, что в дополнение к их фактическому значению числа и другие объекты также имеют значение истины. Есть пустые и непустые объекты. Пустые объекты – это числа с нулевым зна-

чением (0 и 0.0) или пустая строка ('') – строка, состоящая из двух одинарных верхних кавычек и не содержащая ни одной буквы.

Все пустые объекты имеют истинное значение `False`, а непустые – значение `True`. С помощью функции `bool` ты сможешь запросить истинное значение объектов. Взгляни:

```
>>> bool(0)
False
>>> bool(100)
True
>>> bool(23)
True
```

Истинные значения чисел часто запрашиваются в операторах ветвления и повторения.

Приведём здесь небольшой пример использования. Обратное значение числа может быть вычислено, если число не равно нулю.

Программа

```
число = int(input('Введи целое число: '))
if число != 0:
    print('Обратное значение: ',1/число)
```

Ход программы

```
Введи целое число: 2
Обратное значение: 0,5
```

Вместо

```
if число != 0:
```

можно просто ввести следующее выражение:

```
if число:
```

Это упростит программный текст.

И ещё один пример, не касающийся этого программного кода. Если ты в терминале, когда курсор будет находиться в командной строке `>>>`, нажмёшь клавишу ↵, то функция `input()` снова выдаст пустую строку `>>>`. То есть истинным будет значение `False`.

Пишем программу

```
имя = input('Как тебя зовут?:')
if имя != '':  #1
    print('Привет',имя,'!')
else:
    print('Здравствуй, незнакомец!')
```

Ответвление программы 1

```
Как тебя зовут?: Вася
Привет, Вася!
```

Ответвление программы 2

```
Как тебя зовут?
Здравствуй, незнакомец!
```

В строке #1 можно было бы написать следующее:

```
if имя:
```

Полагаю, сразу можно понять, что здесь имеется в виду: «Если имя введено».

Условный повтор или инструкция while

При работе с условными повторами инструкция или последовательность инструкций выполняется столько раз, сколько раз выполняется *условие*. Вот примеры из обычной жизни.

- Пока идёт игра, мы останемся на стадионе и будем поддерживать нашу команду.
- Я сплю, пока не проснусь.

Условный повтор ты можешь представить в виде цикла, который, в свою очередь, можно представить в виде железнодорожного кольца, а выполняемые инструкции – в виде поезда. В этом примере исполнительным элементом условия у нас будет железнодорожный стрелочный перевод. Пока заданное условие соблюдается, железнодорожная стрелка гарантирует прохождение поезда по петле. Как только условие перестанет выполняться, стрелочный перевод «переключится» на отметку «прямо», и поезд уйдёт с петли.

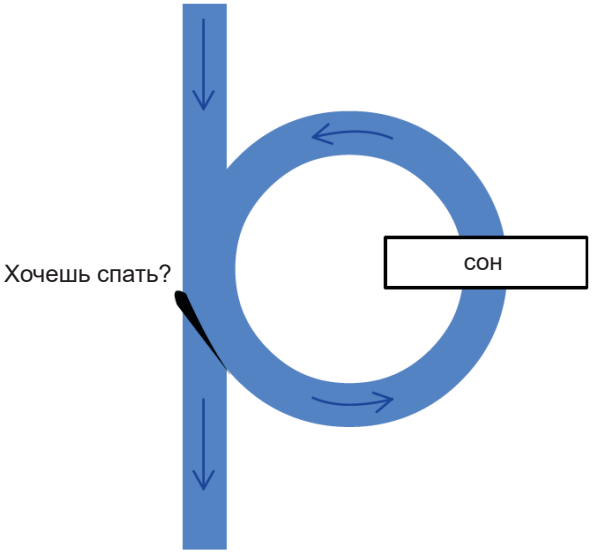


Рис. 9.4. *Обзор условного повтора (цикла)*

Такие повторы программируются с помощью инструкции `while`. Формат у этой команды следующий:

```
While Условие:
    Блок инструкций
```

После команды `while` ставится условие, за которым следует двоеточие. В следующей строке вводятся инструкции, которые нужно повторить. Это *внутренний цикл*. Они должны располагаться на равном количестве пробелов, чтобы интерпретатор смог их распознать как один взаимосвязанный блок.

Возможны случаи, когда инструкции внутреннего цикла выполняются, только если условие не было соблюдено вначале.

Подумай о примере «Я сплю, пока не проснусь». Если ты изначально спишь, то это твоё состояние продолжается до твоего пробуждения.

Проект 10. Угадай число

Правила игры следующие: компьютер выбирает (загадывает) случайное число, а игрок должен с наименьшим количеством попыток это число угадать. После ввода игроком числа компьютер сообщает выбранное (загаданное) им число больше или меньше того, которое ввёл ты.

Пример диалога

```
Я выбрал число между 1 и 100.
Угадай, какое это число.
Число: 50
Слишком маленькое!
Число: 80
Слишком большое!
Число: 70
Верно!
```

В этой программе мы используем функцию `randint()` из модуля `random`.

С помощью функции

```
randint(a,b)
```

ты можешь создавать случайные числа между значениями `a` и `b` включительно. Например, вызывая функцию

```
randint(1,6),
```

мы получаем случайное число в диапазоне от 1 до 6.

Попробуй в IDLE Shell:

```
>>> from random import randint
>>> randint(1.6)
3
```

Программа

```
# угадай число.py
from random import randint              #1
случайное_число = randint(1,100)       #2
print('Выбранное число находится между 1 и 100.')
print('Угадай, какое это число.')
Число = int(input('число:')             #3
while число != случайное число:         #4
    if число < случайное_число:
        print('Маловато будет!')
    else:
        print('Это много!')
    число = int(input('число: '))       #5
Print('Верно!')
```

Как это работает

#1 Функция `randint()` импортируется из модуля `random`. Эта функция выдаёт целое случайное число.

#2 Вызов функции выдаёт число от 1 до 100. Полученное значение присваивается переменной `случайное_число`.

#3 Интерпретатор ожидает ввода с клавиатуры. Введённые с клавиатуры символы «преобразовываются» в целое число (тип `int`).

#4 Оператор `!=` означает «неравный». Пока число не будет найдено, будет выполняться следующий цикл.

#5 Пользователю нужно ввести новое число. После этой строчки блок инструкций `while` (петля) завершён. Если введённое число совпадает со случайным, мы выходим из цикла, и компьютер выдаёт результат «Верно!». Если число не совпадает с «задуманным» компьютером числом, цикл повторяется.

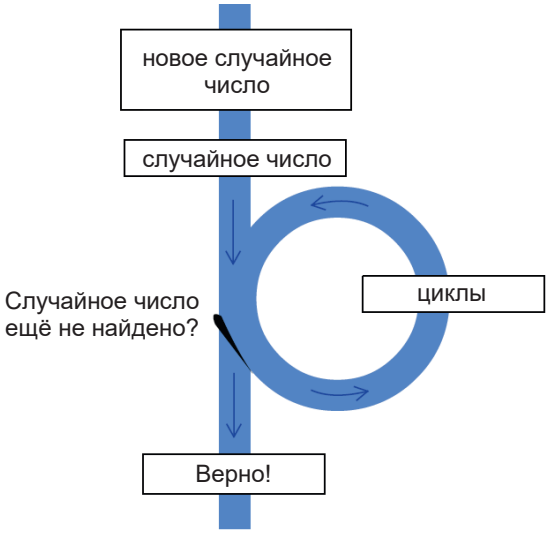


Рис. 9.5. *Обзор программы по угадыванию чисел*

Заверши работу оболочки Shell, сохранив перед этим созданный тобой проект.

Световые сигналы

Циклы `while` важны для создания повторяющихся шаблонов. В этом разделе мы займёмся разработкой различных вариантов мигающих огоньков.

Проект 21. Простая мигалка

Используй схему со светодиодом, которую ты собрал, читая восьмую главу.

Открой LXTerminal и запусти IDLE3 с правами администратора. Для этого введи в командную строку:

```
$ sudo idle3
```

Следующая программа заставит мигать наш светодиод.

Пишем программу

```
from RPi import GPIO          #1
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.OUT)      #2
while True:                   #3
    GPIO.output(10, False)    #4
    sleep(0.5)                #5
    GPIO.output(10, True)     #6
    sleep(0.5)
```

Как это работает?

#1 Для начала из модулей `RPi` и `time` извлекаются функции, которые будут использованы в этом программном коде.

#2 Контакт 10 в GPIO конфигурируется в качестве выходного канала.

#3 Замкнутый цикл. Следующий связанный блок будет повторяться бесконечно.

#4 Контакт 10 установлен на отметке 0. Ток поступает, и светодиод горит.

#5 Ожидание 0,5 с.

#6 Светодиод выключен, потому что контакт 10 отключён от массы. Ток через электрическую цепь контакт 1 – резистор – светодиод – контакт 10 не течёт.

Программа работает неограниченное количество времени. Она содержит бесконечное число повторов, так как условие инструкции `while` всегда имеет истинное значение (`True`). Это называется *бесконечным циклом*. Для остановки нажми комбинацию клавиш **Ctrl**+**C**.

Проект 22. Шаблон мигалки

Вместо простой мигалки, которая включает и выключает светодиод, ты можешь запрограммировать более сложные примеры. С помощью следующего `while`-цикла светодиод будет, не переставая, отправлять сигналы азбуки Морзе в виде буквы А (короткий – длинный, т. е. точка – тире).

```
while True:
    GPIO.output(10, False)     #1
    sleep(0.2)
    GPIO.output(10, True)
    sleep(0.2)
    GPIO.output(10, False)     #2
    sleep(0.6)
    GPIO.output(10, True)
    sleep(1)
```

Как это работает?

`#1` Светодиод включается на 0,2 с.

`#2` Светодиод включается на 0,6 с.

Вопросы

1. Чем отличаются знаки равенства = и ==?
2. Что такое блок инструкций?
3. Как завершить программу Python, которая содержит бесконечный цикл?
4. Какую нужно вызвать функцию, чтобы запрограммировать случайное число от 1 до 6, символизирующее бросок кубика?
5. Какое истинное значение имеет строчка `'0'`?

Задания. Идти на улицу или не идти?

Компьютерные программы могут поддерживать людей во время поиска решений. Разработай интерактивную программу, которая отвечает на вопрос: «Стоит ли мне сегодня идти гулять на улицу?»

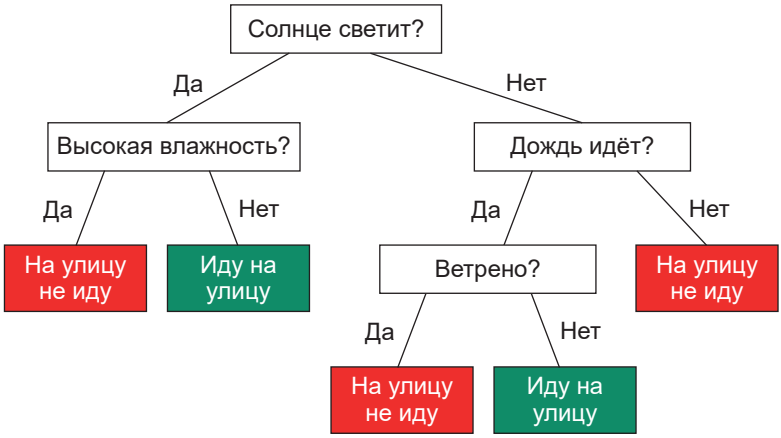


Рис. 9.6. *Так выглядит дерево решений*

Ответы на вопросы

1. Знак равенства `=` используется для присваивания переменной нужного значения. К примеру, выражение `a = 1` присваивает переменной `a` значение `1`. И напротив, двойной знак равенства `==` является оператором сравнения. Например, `a == 1` даёт значение `True`, если переменная `a` обладает значением `1`.
2. Блок инструкций состоит из одной или нескольких инструкций, имеющих равное количество отступов от левого края.
3. Ты можешь завершить программу Python с помощью Keyboard-Interrupt. Для остановки программы нажми комбинацию клавиш **Ctrl**+**C**.
4. Симулировать бросок кубика можно с помощью инструкции `randint(1, 6)`.
5. Строка `'0'` не является пустой, она содержит знак (`0`). Поэтому её значение истинно (`True`).

Решение задачи

Программа

```
# дерево решений.py
print('Стоит ли идти гулять на улицу?')
a=input('Светит солнце? (д/н: '))              #1
if a == 'д':                                   #2
    a = input('Высокая влажность? (д/н): ')
```

```
    if a == 'д':
        print('На улицу не иду.')
    else:
        print('Иду на улицу.')
else:                                                    #3
    a= input('Дождь идёт? (д/н): ')
    if a== 'д':
        print('На улицу не иду.')
    else:
        a= input('Ветренно? (д/н):')
        if a== 'д':
            print('На улицу не иду.')
        else:
            print('Иду на улицу.')
```

Как это работает

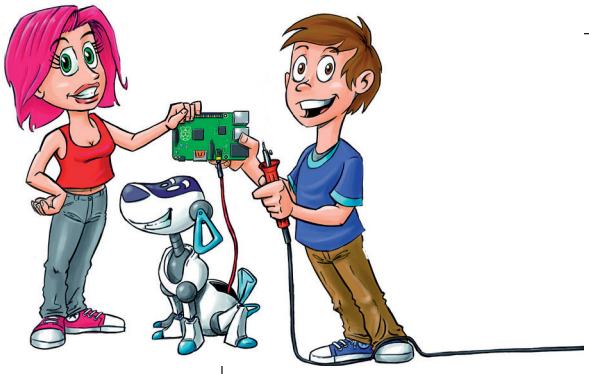
#1 Для всех ответов мы используем одну и ту же переменную a.

#2 Условие подходит в случае, если пользователь введёт маленькую букву.

#3 Оператор `else` относится к оператору `if` в строке #2. Обрати внимание на правильную структуру блока с корректными отступами. Каждый оператор `else` должен находиться точно под оператором `if` (рис. 9.7).

```
print('Стоит ли идти гулять на улицу?')
a=input('Солнце светит? (д/н): ')
if a =='д':
    a=input('Высокая влажность? (д/н):')
    if a =='д':
        print('На улицу не иду.')
    else:
        print('Иду на улицу.')
else:
    a=input('Дождь идёт? (д/н):')
    if a=='д':
       print('На улицу не иду.')
    else:
        a=input('Ветрено? (д/н):')
        if a=='д':
           print('На улицу не иду.')
        else:
           print('Иду на улицу.')
```

Рис. 9.7. *Код программы для выбора решения, идти ли на улицу*



10 Управление с помощью переключателя

Ты к Raspberry можешь подключить самостоятельно изготовленную схему. Компьютерная программа будет запрашивать состояние ключей и в зависимости от их состояния принимать решение.

Мы соберём счётчик, дверной звонок и систему сигнализации. Более сложный проект – это считыватель с перфокарт, с помощью которого можно читать цифровые ключи.

Переключатель

Мы использовали интерфейс GPIO в качестве выходного канала. Теперь же он послужит входным портом к одному из контактов GPIO (эти контакты ещё называют пинами), мы подключим собранную схему. Программа (в цикле `while`) считывает состояние ключей и реагирует соответствующим образом.

Задача смонтированного в схеме ключа состоит в том, чтобы замыкать и разрывать цепь, по которой течёт ток. Ты с лёгкостью можешь самостоятельно собрать схему для

различных целей, используя бумагу, фольгу, липкую ленту и клей. Поскольку наши цепи будут слаботочными (максимальный ток потребления не более 50 мА) и работать будут с низким напряжением (максимум 5 В), этот эксперимент полностью безопасен. В главе 5 ты найдёшь инструкцию по изготовлению базовой модели из бумаги, достойно зарекомендовавшую себя во многих других проектах.

Контакт GPIO, используемый в качестве входного порта, всегда находится в одном из следующих состояний:

- `False` (ложь), когда на контакт подаётся низкий уровень напряжения (от 0 до 0,5 В);
- `True` (истина), когда к контакту прикладывается высокий уровень напряжения (от +2,4 до +3,3 В).

На управляющий контакт подается или низкий уровень (от 0 до +0,5 В), что соответствует логическому нулю `False` (ложь), или высокий уровень напряжения (от +2,4 до +3,3 В), соответствующий логической единице, или `True` (истина). Среднего значения напряжения от +0,5 до +2,4 В в этой схеме быть не должно.

Ещё раз следует повторить, для Python-интерпретатора `False` (ложь) значит то же самое, что и 0, а значение `True` (истина) – то же самое, что и 1. Ты легко можешь это проверить в Python Shell:

```
>>> False == 0
True
>>> True == 1
True
```

На рис. 10.1 представлена принципиальная схема подключения резистора и выключателя, когда контакт 10 используется в качестве входного порта. Этот контакт подключается к корпусу (GND) (контакт 6) через резистор мощностью в 10 кОм. Проводники, по которым течёт ток напряжением от +2,4 до +3,3 В, обозначены жирной линией.

- Если контакт выключателя разомкнут (рис. 10.1 слева), на контакт 10 напряжение от +2,4 до +3,3 В не подаётся (на контакте 10 низкий уровень напряжения), и этот контакт находится в состоянии `False` (ложь). Этот контакт подключён к корпусу через нагрузочный резистор (англ. *to pull down* – «загружать»).
- Если контакт выключателя замкнут, контакт 10 соединён с контактом 1 (+3,3 В). В этом случае на контакте 10 присутствует высокий уровень напряжения (примерно

+3,3 В) и находится в состоянии `True` (истина) (рис. 10.1 справа).

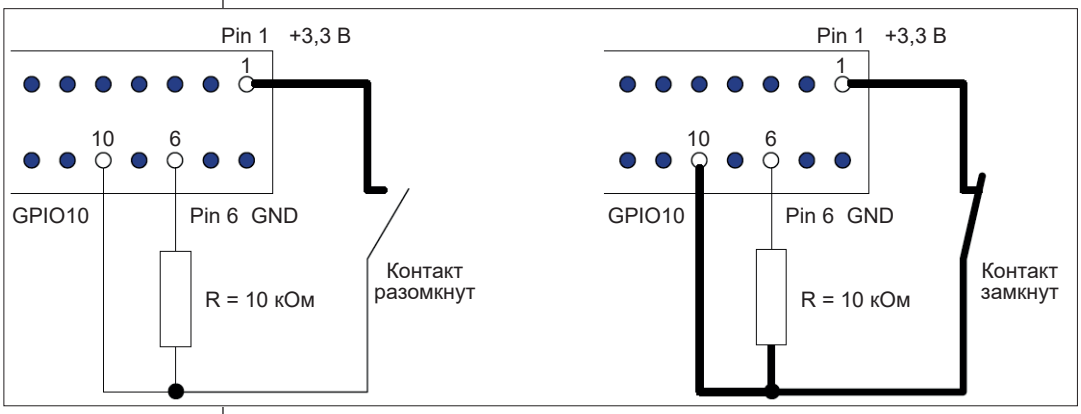


Рис. 10.1. *Переключатель подключен к контакту 10 через нагрузочный резистор. Слева состояние `False`, справа состояние `True`*

Нагрузочный резистор можно сравнить с камнем на качелях. Взгляни на рис. 10.2. На картинке слева качели находятся в непонятном положении – немного наклонены влево, немного – вправо. Камень, если его поместить на одну из сторон, добавит ей веса, обеспечив качелям устойчивое положение. Если кто-то сядет на другую сторону (картинка справа), то положение изменится.

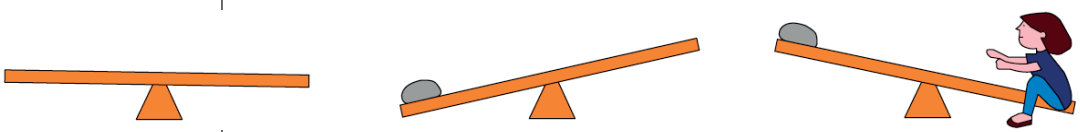


Рис. 10.2. *Слева – неясное положение; в центре – состояние `False`; справа – состояние `True`*

Теперь поговорим о том, как сигнал, появляющийся на контакте 10, может быть обработан в программе Python.

Тестируем выключатель

Если компьютер должен реагировать на состояние выключателя (контакт замкнут или разомкнут), ему придётся постоянно контролировать это состояние, чтобы суметь распознать любые изменения. Следующая программа позволит нам протестировать выключатель. Его состояние проверяется программой каждую секунду. Для этого в программе необходимо запустить бесконечный цикл, кото-

рый будет постоянно опрашивать контакт 10, к которому подключен контакт выключателя. Время, пока программа ждет, чтобы повторить проверку контактов выключателя, задается в строке `sleep(1)`.

Ход программы

```
0
0
1
1
…
```

Судя по всему, через две секунды контакты выключателя будут разомкнуты.

Пишем программу

```
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.IN)        #1
while True:                    #2
    print(GPIO,input(10))      #3
    sleep(1)
```

Как это работает?

`#1` Контакт 10 настроен в качестве входного канала.

`#2` Начало бесконечного цикла.

`#3` Текущее состояние контакта 10 отображается на экране.

Завершить программу в IDLE3 Shell ты можешь, нажав **Ctrl**+**C**.

Проект 23. Счётчик

Выключатель может быть использован для подсчёта результатов. В аэропортах, общественных местах или организациях постоянно проводится подсчёт людей, так как необходимо знать, сколько человек находится в зоне контроля.

В этом проекте Raspberry будет считать, сколько людей открывают и закрывают дверь. Таким образом, ты сможешь установить, как часто заходили в твою комнату, если дверь постоянно открывалась и закрывалась.

На рис. 10.3 показан переключатель, который можно смастерить из бумаги, фольги или клейкой ленты всего за несколько минут. Он состоит из двух частей.

- Над дверью у нас находится переключатель по описанию такой же, как и в примере с «лимонадной машиной» (проект 13). К стене он крепится с помощью клейкой ленты.
- А к самой двери мы с помощью тканевого скотча крепим угол из картона. Следи за тем, чтобы «волны» картона легли в продольном направлении, для того чтобы конструкция была прочной.

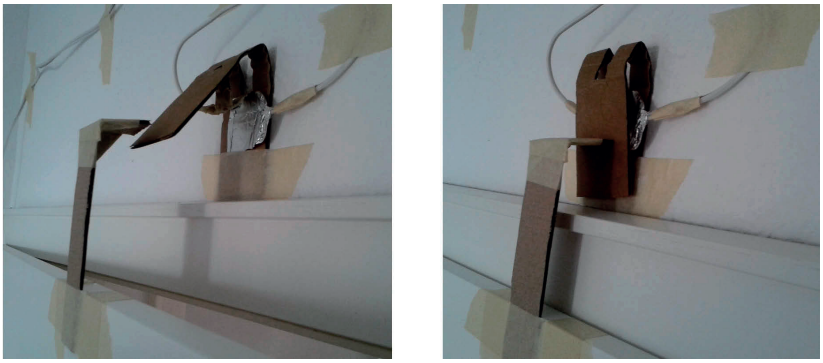


Рис. 10.3. *Переключатель (схема) для двери, изготовленный из бумаги, фольги и клейкой ленты. Слева дверь открыта, справа – закрыта*

Ты будешь использовать схему, показанную на рис. 10.1. Программа в режиме бесконечного цикла должна подсчитать, сколько раз контакты переключателя размыкались и замыкались, а потом вывести на экран количество замыканий, а также дату и время.

Ход программы (пример)

```
1 ThuJul 17 07:27 2014
2 ThuJul 17 07:34 2014
3 ThuJul 17 09:21 2014
```

Пишем программу

```
# счетчик.py
from RPi import GPIO
from time import *                    #1
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.IN)               #2
```

```
count=0                                #3
while True:
    if GPIO.input(10)== False          #4
        count += 1                     #5
        print(count, asctime())        #6
        sleep(0.1)                     #7
        while not GPIO.input(10):      #8
            sleep(0.2)                 #9
```

Как это работает?

#1 Звёздочка * означает, что из модуля time импортируются *все* функции.

#2 Контакт 10 настроен в качестве входного канала.

#3 Счётчик устанавливается в нулевое положение.

#4 Если контакт размыкается, напряжение на контакте 10 падает с 3,3 В (True) до 0 В (False). Далее выполняется блок инструкций с отступом (выражение if).

#5 Содержание count повышается до 1.

#6 Текущее число и время открывания двери выводится на экран.

#7 Контакты выключателя могут выдать ошибку (дребезг контактов). Это значит, что контакт в первые миллисекунды неустойчив. Поэтому, чтобы переходные процессы успокоились, программа выжидает 0,1 с (= 100 миллисекунд).

#8 Теперь программа ждёт до тех пор, пока контакты выключателя снова замкнутся, а на контакте 10 напряжение поднимется до 3,3 В (True).

#9 Здесь завершаются все три блока (while, if, while), чтобы через 0,2 с начался следующий цикл.

Расширения для экспертов. Сохраняем данные

Возможно, ты захочешь сохранить время в журнале событий. Такой журнал представляет собой текстовый документ, в который записываются определённые в программе события. Своего рода протокол. Журнал можно позднее открыть в любом текстовом редакторе и просмотреть содержимое.

Чтобы добавить возможность записи контролируемых событий в журнал, в коде программы нужно одну строку добавить, а одну – изменить.

Изменения

После строчки #1 введи следующую инструкцию:

```
log = open('log.txt', 'w')
```

Таким образом, в той же папке, где хранится программа, создаётся файл с именем `'log.txt'`. С помощью второго аргумента `'w'` ты фиксируешь, что файл открывается для письма (англ. *to write* – «писать»).

В строке #4 добавляешь две инструкции:

```
file=log, flush=True.print(count,asctime(), file=log, flush=True)
```

То есть к функции `print()` добавлены два параметра:

- `file=log` способствует тому, что данные записываются не на экране, а в конце файла;
- `flush=True` отвечает за то, чтобы данные после внесения записи сохранились. То есть если кто-то выключит компьютер, прервав тем самым выполнение программы, информация не будет утеряна.

Ты можешь открыть журнал событий в обычном редакторе (например, TextEditor) и посмотреть свои данные. Чтобы запустить TextEditor, выполни следующие действия.

- В левом верхнем углу нажми кнопку . Откроется основное меню. В основном меню установи указатель мыши на строку **Стандартные** (Standard) и в появившемся подменю щёлкни мышью на строке **Text Editor**.
- В окне программы выбери команду меню **Файл ⇨ Открыть** (File ⇨ Open), в появившемся диалоговом окне выбери папку, где сохранён журнал, и дважды щёлкни мышью по файлу журнала *log.txt*.

Журнал событий находится в той же папке, в которой сохранена сама программа.

Проект 24. Дверной звонок – проигрываем звуковые файлы

Выключатель фиксации количества открытий двери, описанный в последнем разделе, ты можешь использовать в качестве датчика для проигрывателя звуковых файлов. Когда кто-то будет открывать дверь, музыкальный файл

будет проигрываться. Таким звуковым файлом может быть файл со звуком гонга, свистом, колокольным звоном или приветствием мужским голосом («Добро пожаловать в мою комнату»). Главное, чтобы такой звуковой файл находился в формате *WAV*, т. е. окончанием этого файла будет *.wav*. Бесплатные файлы в этом формате ты можешь найти в интернете. Если ты скачал архив с программами для этой книги, несколько звуковых файлов ты найдёшь в папке *SOUNDS*, вложенной в папку *kap_10*.

Музыка, которую умеет играть Raspberry

Чтобы слышать звуки, тебе понадобится динамик. У самого Raspberry динамика нет, но, возможно, ты через порт HDMI подключил мультимедийный монитор, и у этого монитора имеется встроенный динамик. Чтобы проверить наличие динамика, проведи тест.

Открой окно LXTerminal и введи следующую команду:

```
$ speaker-test
```

Если слышишь шумы (какой-то шорох), значит, твой монитор оснащён динамиком. Останови тест, нажав комбинацию клавиш **Ctrl**+**C**. Чтобы улучшить звучание, подключи к своему Raspberry колонки со встроенным усилителем (такие колонки ещё называют активными). Колонки подключаются к аудиопорту через стереоразъём со штекером диаметром 3,5 мм (рис. 10.4).

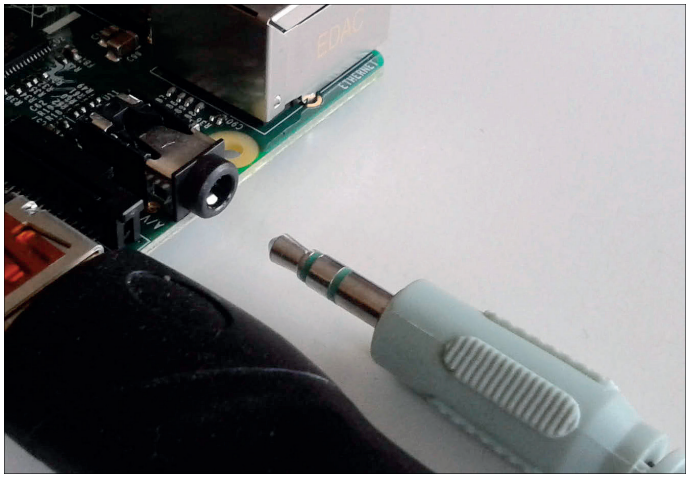


Рис. 10.4. *Аналоговый аудиопорт вход-выход RPi (модель B+) расположен рядом с гнездом HDMI*

Операционная система Raspberry приоритетом выбирает канал HDMI. Если ты хочешь вывести звук через гнездо 3,5 мм, тебе нужно изменить конфигурацию. Для этого в окне терминала LXTerminal введи в командную строку следующую команду:

```
$ sudo raspi-config
```

Запустится программа конфигурации, окно которой ты увидишь на экране. Нажимая клавишу [↓], выбери пункт 9 **Advanced Options** (Дополнительные параметры) и нажми клавишу [↵].

Затем выбери пункт **A9 Audio** и снова нажми клавишу [↵].

Далее выбери опцию **Force 3,5 mm ('Headphone') Jack**, с помощью клавиши [→] выбери **OK** и нажми [↵].

Чтобы проверить, работают ли колонки, введи в командную строку терминала команду

```
$ speaker-test
```

Послушай, есть ли тихое шипение из колонок. Если шум в виде тихого шороха слышен, звук на колонки подаётся.

Нажми комбинацию клавиш **Ctrl**+**C**.

Проигрываем файлы

С помощью программы **Aplay**, установленной на твоём Raspberry, ты можешь проиграть файлы в форматах *.voc, .wav, .raw .au*. По умолчанию в папке */usr/share/sound/alsa* сохранено несколько звуковых файлов. Сам проигрыватель вызывается с помощью команды. Проще всего вызвать программу **Aplay** можно вот по этому образцу:

```
$ aplay имя звукового файла
```

Где `имя` – имя файла с его расширением. Но сначала нужно открыть папку, в которой этот звуковой файл хранится. Чтобы перейти в целевую папку и посмотреть, какие файлы в ней хранятся, введи в командной строке терминала команду `cd` и путь к целевой папке и нажми клавишу [↵]. Например:

```
$ cd /usr/share/sounds/alsa
$ ls
```

В ответ ты увидишь файлы, содержащиеся в папке *alsa*:

```
Front_Center.wav  Noise.wav         Rear_Right.wav
Front_Left.wav    Real_Center.wav   Side_Left.wav
Front_Right.wav   Real_Left.wav     Side_Right.wav
```

Введи в командную строку терминала команду и нажми клавишу ↵:

```
$ aplay Side_Right.wav
```

Ты услышишь очень короткую фразу, тихо сказанную женским голосом.

В папке проекта к этой книге ты найдёшь папку с музыкальными темами, которые можно установить в качестве «дверного звонка». Но ещё больший выбор ты найдёшь в интернете по ссылке, например http://www.wavsource.com.

Мы же разработаем простенькую программу Python, которая при нажатии на ↵ будет воспроизводить музыкальные файлы снова и снова.

- Открой файловый менеджер (для этого щёлкни мышью по кнопке в верхней части экрана). Компьютер автоматически откроет папку *pi*, содержание которой ты увидишь в правой части окна. Дважды щёлкни мышью по папке со своими программами (ранее этой папке мы присвоили имя *Python_projects*).
- Выбери команду меню **Файл** ⇨ **Создать папку** (File ⇨ Create Folder). Назови эту папку *sounds*. Эта папка должна содержать файлы формата *wav*, среди которых будет файл с именем *gong.wav*. Этот файл мы сейчас переместим в папку *sounds*.
- Открой папку, в которой ты сохранил (скачал из интернета) программы для этой книги. Тексты программ рассортированы по главам. Если эта папка ещё не разархивирована, разархивируй её.
- Открой папку *kap_10*, найди файл под именем *gong.wav* и щелкни по нему правой кнопкой мыши.
- Выбери из появившегося меню команду **Копировать** (Copy).
- Открой папку *Python_projects* и вложенную в неё папку *sounds*.
- Щёлкни правой кнопкой мыши в правой (пустой) части диалогового окна обозревателя и выбери из появивше-

гося меню команду **Вставить** (Paste). Скопированный файл будет вставлен в открытую папку.

- Закрой окно обозревателя.
- Запусти оболочку IDLE 3. Для этого нажми кнопку в левом верхнем углу окна, выбери в появившемся основном меню команду меню **Программирование** (Programming) и в появившемся подменю щёлкни мышью на строке **Python 3 (IDLE)**.
- В полосе меню оболочки выбери команду **Файл ⇨ Сохранить как** (File ⇨ Save As), в появившемся диалоговом окне выбери папку *Python_projects* и сохрани в ней новый проект под именем *sound_test.py*.

Важно, чтобы в папке, где сохранён новый проект, находилась вложенная папка *sounds*, содержащая звуковые файлы (рис. 10.5), иначе ссылка, которую мы введём в коде нашей программы, не найдёт нужный файл для воспроизведения.

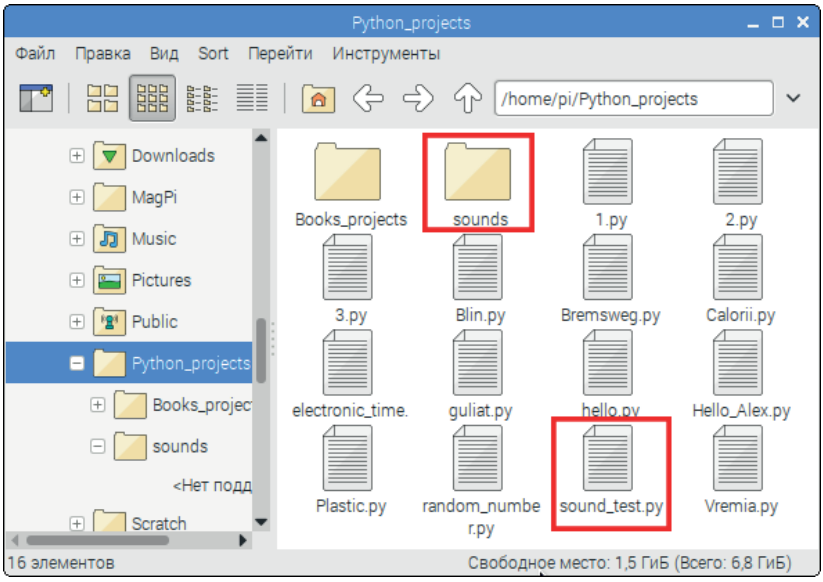


Рис. 10.5. *Папка проектов с программой Python и папка со звуковыми файлами*

Пишем программу

Настало время создать новую программу под именем *sound_test.py*.

- Выбери в оболочке **Python Shell** команду меню **File ⇨ New File** (Файл ⇨ Новый файл). Откроется блокнот оболочки PythonShell.

➤ Введи в блокнот код программы:

```
import os                                                    #1
while True:
    input('Запускать музыку клавишей <ENTER>')               #2
    os.system('aplaysounds/gong.wav')                        #3
```

Как это работает?

#1 Импортируется модуль os. Он содержит функции, которые взаимодействуют с операционной системой (*os* – операционная система).

#2 В режиме бесконечного цикла модуль os ожидает, когда пользователь нажмёт ↵. Речь здесь идёт только об ожидании. Введёт ли пользователь что-то другое перед нажатием ↵ или нет, совершенно не важно.

#3 С помощью модуля os.system() отправляется команда, как если бы её ввели в командной строке в консоли (LXTerminal). Команда – это цепь символов, и она берётся в кавычки. Состоит эта команда из имени *aplay* и пути к аудиофайлу: *sounds/gong.wav*.

Дверной звонок

Ты используешь схему, показанную на рис. 10.1, и выключатель, смонтированный на двери (рис. 10.3). Следующая программа будет проигрывать аудиофайл до тех пор, пока контакты разомкнуты.

Пишем программу

```
import os
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.IN)

while True:
    if not GPIO.input(10)                          #1
        os.system('aplay sounds/gong.wav')         #2
        sleep(1)                                   #3
```

Как это работает?

#1 Когда контакты выключателя разомкнуты, на контакте 10 напряжение отсутствует (низкий уровень напряжение из-за нагрузочного резистора), что соответствует состоянию False (ложь). Поэтому вызов функции GPIO.

`input(10)` устанавливает контакт 10 в состояние `False` (ложь). Эту строку можно записать так:

```
if GPIO.input(10) == 0
```

`#2` Проигрывается аудиофайл из *gong.wav* из папки *sounds*.

`#3` Программа перед следующим циклом ожидает одну секунду.

Проект 25. Сигнализация

Сигнализация должна работать без использования клавиатуры и монитора. Вот из каких деталей она состоит:

- RPi с блоком питания и динамиком;
- светодиод с добавочным резистором номиналом 130 Ом;
- кабель;
- выключатель для двери как на рис. 10.3;
- обычный переключатель, спрятанный где-то в помещении.

Что происходит?

Спустя 10 с после запуска программы сигнализация переходит в активное состояние. До истечения этого времени ты можешь покинуть помещение без риска, что сигнал тревоги сработает. Через 10 с это сделать уже нельзя, так как устройство приводится в действие. Сначала сигнализация находится в состоянии *ожидания*. Если кто-то откроет дверь, она перейдёт в активную фазу. Для контроля включается светодиод.

Дальше у нас имеется два варианта. Если ничего не происходит, то через 5 с срабатывает сигнализация, и начинает постоянно звучать выбранный нами аудиофайл. Этот звук должен напугать нарушителя.

Второй вариант: ты снова закрываешь дверь, нажимаешь вторую потайную кнопку, местонахождение которой знают только посвящённые, и удерживаешь её до тех пор, пока светодиод не погаснет. В этом случае сигнализация выключается и снова переходит в режим *ожидания*.

Если же сработал сигнал тревоги, он выключится лишь тогда, когда кто-нибудь нажмёт секретный выключатель.

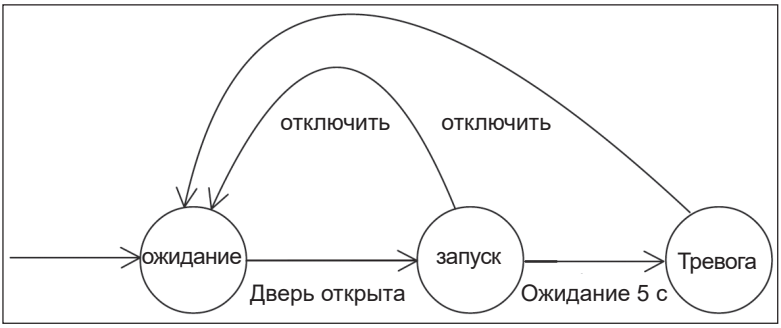


Рис. 10.6. *Состояние сигнализации – режим активный и режим ожидания*

Аппаратное и программное обеспечение

Аппаратная часть здесь очень простая. Для выключателя тебе не понадобятся резисторы, потому что мы будем использовать контакты 3 и 5 GPIO. У них есть одна особенность. Они обладают встроенными нагрузочными резисторами, которые поддерживают на этих контактах положительное напряжение (примерно 3,3 В).

Неподвижный контакт каждого выключателя подключён к контактам 3 и 5 разъёма расширения портов ввода-вывода GPIO. Подвижные контакты выключателей через контакт 6 соединены с корпусом (GND). То есть если на контакте уровень напряжения низкий (логический ноль, или `False`), значит, контакты выключателя разомкнуты. Если на контакте 10 присутствует высокое напряжение (логическая единица, или `True`), значит, контакты выключателя замкнуты.

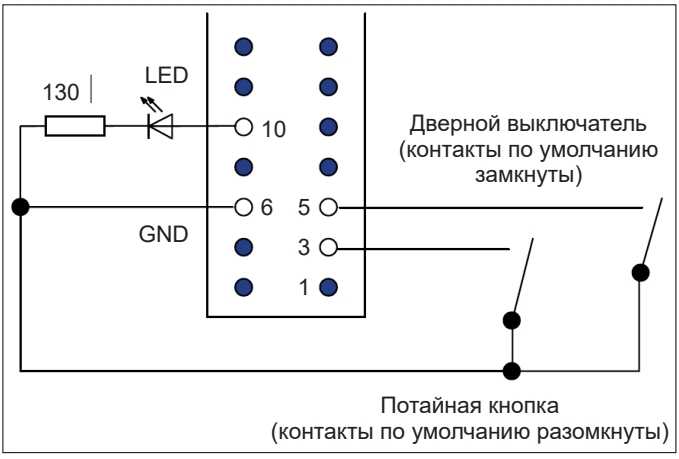


Рис. 10.7. *План схемы сигнализации. Светодиод горит, если на контакте 10 положительное напряжение*

Пишем программу

```
import os
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(3, GPIO.IN)
GPIO.setup(5, GPIO.IN)
GPIO.setup(10, GPIO.OUT)

sleep(10)

while True:                              #1
    GPIO.output(10, False)               #2
    if GPIO.input(5):                    #3
        GPIO.output(10, True)            #4
        sleep(5)                         #5
        while GPIO.input(3):             #6
            os.system('aplay sounds/huhu.wav')
```

Как это работает?

#1 Начало бесконечного цикла.

#2 Светодиод выключается.

#3 Если дверь открыта, контакты выключателя двери разомкнуты, а на контакте 5 логическая единица (высокое напряжение, определяемое нагрузочным резистором). В этом случае состояние контакта 5 `True`. Далее выполняется следующий блок инструкций, выделенный отступом.

#4 Светодиод включается.

#5 Время ожидания 5 с.

#6 Пока не будет нажата кнопка отключения, звук тревоги будет воспроизводиться. Здесь тебе следует продумать следующие моменты:

- если контакты кнопки отключения сигнала тревоги разомкнуты, на контакте 3 присутствует логическая единица (состояние `True` определяется встроенным нагрузочным резистором) и выполняется бесконечный цикл;
- если кнопку отключения тревожного сигнала нажать до истечения 5-секундного ожидания, цикл завершится, и сигнализация будет находиться в дежурном режиме ожидания размыкания контактов выключателя двери.

Проект 26. Единички и нолики. Перфокарта в качестве цифрового ключа

Уже в восемнадцатом столетии перфорированная лента использовалась в управлении музыкальной шкатулкой и ткацким станком. Перфолента – это такая длинная лента из картона, в которой пробит шифр из отверстий. Лента разделена на строчки, в каждой из которых определённое количество позиций. Каждая позиция – это или пробитое отверстие, или нетронутая поверхность картона. То есть здесь закладывается два варианта. Условно можно сказать, что каждое отверстие – это логическая единица, а отсутствие отверстия – логический ноль. Шифр из нулей и единиц кодирует в одной строке цифры и/или буквы. Таким образом перфолента может сохранять информацию.

Перфорированная *карта* – это перфолента определённой длины. Самым распространённым форматом карт является карта с 80 ячейками в длину и 12 строками в ширину. Такие перфокарты использовались в больших ЭВМ (электронных вычислительных машинах), а позже – в первых компьютерах. Да, на заре становления компьютерных технологий перфокарты играли большую роль. На них сохраняли компьютерные программы и данные. В 60-е годы им на смену пришли магнитные ленты. Но перфокарты как прочные механические носители небольшого объёма информации (например, с данными опросов) и сегодня могут использоваться. В этом проекте мы будем использовать перфокарту в качестве цифрового ключа.

Как это работает?

Мигает красный светодиод. Это значит, что перфокарту можно поместить в паз считывателя карт (далее просто считыватель). Перфокарта – это кусочек пластика с пробитыми в ней прямоугольными отверстиями. Есть два варианта, как вставить карту в считыватель.

1. Последовательность из отверстий соответствует серии нулей и единиц, копия которых сохранена в компьютере. Если считанный с перфокарты код совпадает с сохранённым на компьютере кодом, значит, ключ подходит. Светодиод жёлтого (или любого другого) цвета светится. Вместо жёлтого светодиода можно включить,

конечно, исполнительное электрическое устройство для открывания двери или иной механизм.

2. Последовательность из отверстий не соответствует сохранённой комбинации. В этом случае ключ неправильный. Светится красный светодиод.

Для извлечения карты из считывателя тоже существует два мигающих сигнала. Рисунок 10.8 наглядно демонстрирует принцип действия через диаграмму переходов состояния.

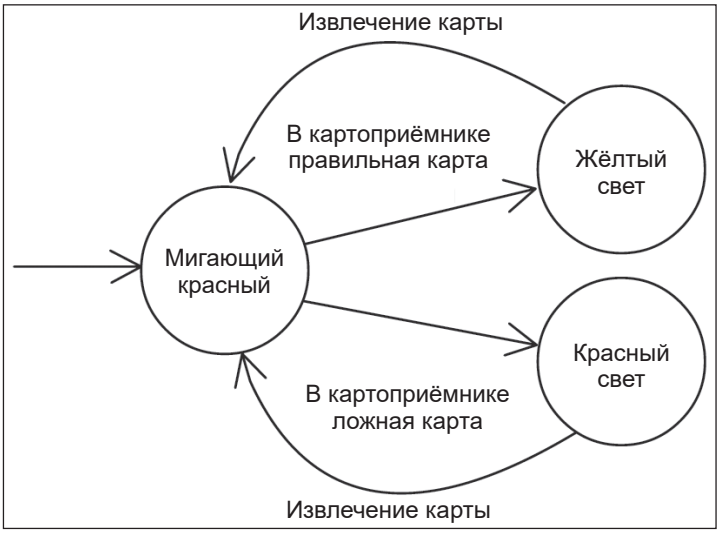


Рис. 10.8. *Диаграмма переходов состояния для «цифрового ключа»*

Как самому изготовить считыватель перфокарт?

Считыватель перфокарт содержит ряд контактов. Нижняя часть считывателя перфокарты – это общий контакт. Каждый верхний скользящий контакт подключается к разным контактам разъёма GPIO компьютера. Если между нижним общим контактом и верхним скользящим контактом находится перфокарта (отверстия нет), значит, и контакт отсутствует. Если в месте перфокарты, предназначенном для отверстия, отверстие есть, значит, нижний, общий контакт замкнут с верхним, скользящим контактом. Принцип работы перфокарт показан на рис. 10.9.

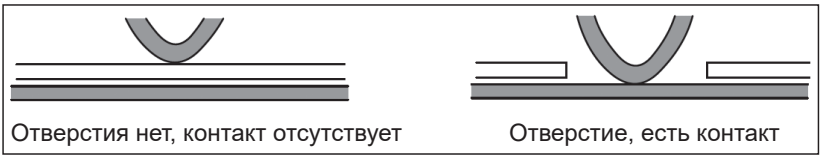


Рис. 10.9. *Принцип работы считывателя перфокарт*

Всего через четверть часа ты сам из подручных материалов сможешь собрать такой считыватель. Тебе понадобятся:

- гофрированный картон (можно взять упаковочный);
- клей;
- тканевый скотч;
- алюминиевая фольга;
- канцелярские скрепки;
- плоский ленточный кабель (шлейф);
- возможно, несколько монет или винтиков в качестве груза.

Возьми четыре скрепки и согни их кончиком вперёд (рис. 10.10).

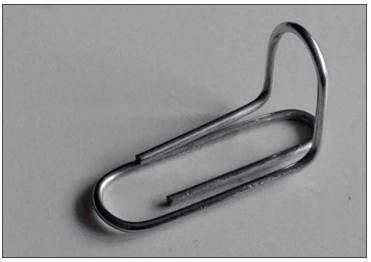


Рис. 10.10. *Так должны выглядеть скрепки с согнутым концом*

Отдели от шлейфа ленту с пятью проводами.

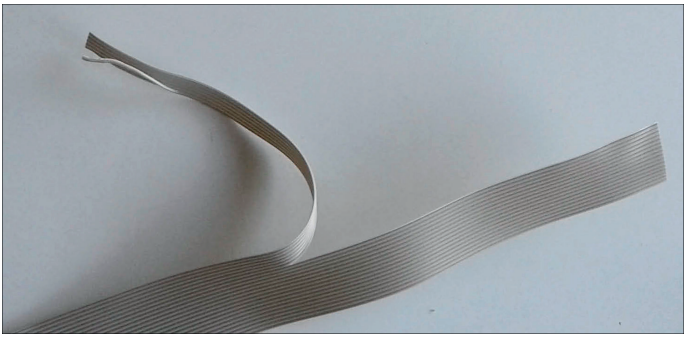


Рис. 10.11. *Плоский ленточный кабель*

Из картона вырежи прямоугольник (примерно 15×20 см). Это будет опорная пластина (основание) нашего картоприёмника. На эту пластину наклей полоску фольги. С одной стороны полоска фольги должна выступать, чтобы её можно было подсоединить к кабелю. На этот выступающий участок фольги наклей тканевый скотч, чтобы потом перфокарта случайно не соскользнула под фольгу. Наклей три картонные полоски на опорную пластину, как показано на рис. 10.12.

Рис. 10.12. *Так выглядит опорная пластина*

Вырежи из гофрированного картона четыре полоски. Они нам понадобятся для выключателя. Следи за тем, чтобы «волны» картона лежали в продольном направлении, чтобы конструкция сохраняла прочность.

Соедини первую «жилу» шлейфа с полоской фольги.

Вставь согнутую скрепку на картонную полоску. Закрепи эту полоску, как показано на рис. 10.13. Скрепка должна касаться фольги. Соедини вторую «жилу» шлейфа со скрепкой.

Рис. 10.13. *Переключатель, считывающий перфокарту. Если он через отверстие касается фольги, электрическая цепь замыкается. Если нет, цепь остаётся разомкнутой*

Проделай то же самое с тремя другими полосками картона. Ты можешь эти полоски приложить какой-нибудь толстой

книгой, чтобы они хорошо закрепились, пока затвердеет клей. Приклей переднюю часть картона в виде крышки над опорной платой так, чтобы появилась прорезь для вставки перфокарты.

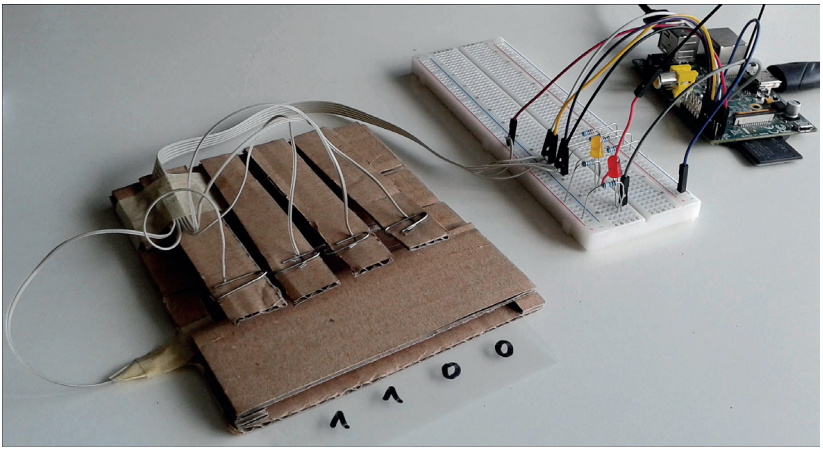


Рис. 10.14. *Считыватель для проекта «Цифровой ключ» (с перфокартой)*

Для изготовления перфокарты возьми кусочек твёрдого пластика (можешь позаимствовать из коробки от конфет или чего-то похожего) и с помощью канцелярского ножа прорежь в нём несколько прямоугольных отверстий. Можно, конечно, взять бумагу или тонкий картон, но, мне кажется, настоящий пластик выглядит как-то элегантнее и даже немного напоминает научно-фантастические фильмы.

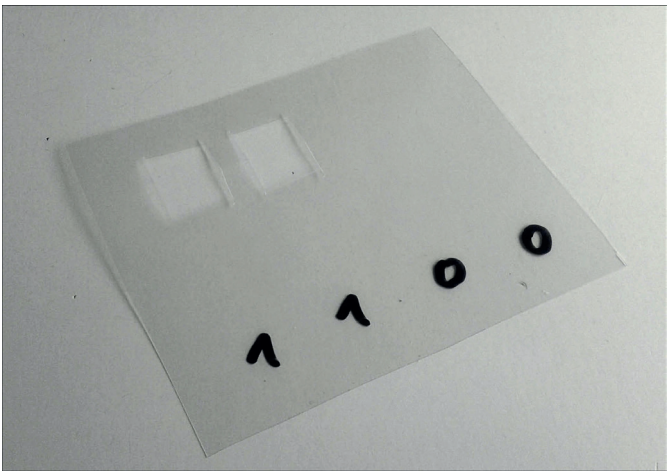


Рис. 10.15. *Перфорированная карта из пластика*

Следи за тем, чтобы отверстия, прорезанные в «перфокарте», находились точно под скрепками-контактами, когда карта до упора будет входить в прорезь считывателя.

Схема

Для электронной схемы на основе макетной платы в качестве нагрузочного сопротивления тебе понадобятся:

- четыре резистора номиналом 10 кОм;
- два светодиода (например, красный и жёлтый);
- два резистора номиналом 130 Ом в (это будут нагрузочные резисторы для светодиодов).

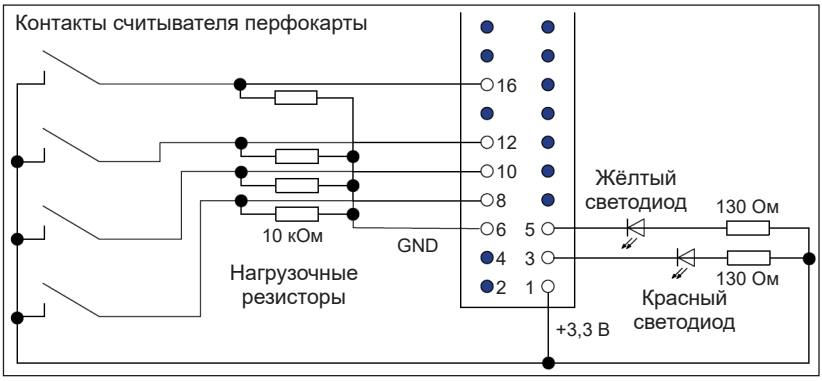


Рис. 10.16. *Принципиальная схема подключения считывателя карт к Raspberry*

Порядок сборки следующий.

Вначале вставь в монтажную плату электронные детали (резисторы и светодиоды). На рис. 10.17 показано, как это должно выглядеть.

Затем соедини эти детали с GPIO через провод с разъёмами «папа-мама».

В качестве выходного канала для управления светодиодами мы используем контакты (пины) 3 и 5 разъёма GPIO. Контакты 8, 10, 12 и 16 будут служить входными портами (конечно же можно вместо них запрограммировать и какие-то другие контакты).

Пять проводов ты можешь закрепить в монтажной плате с помощью проводов с разъёмами, которые идут к GPIO (рис. 10.18).

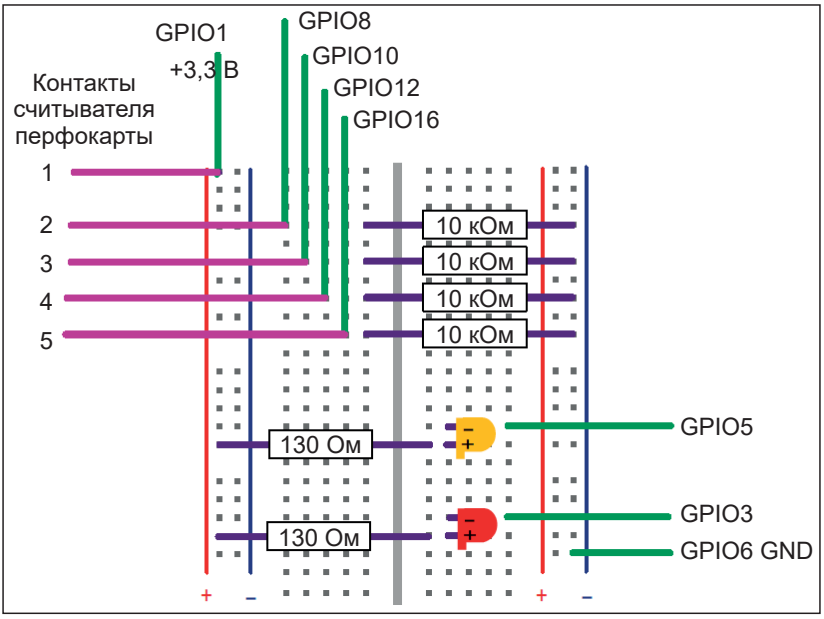


Рис. 10.17. *Расположение электронных деталей на съёмной плате*

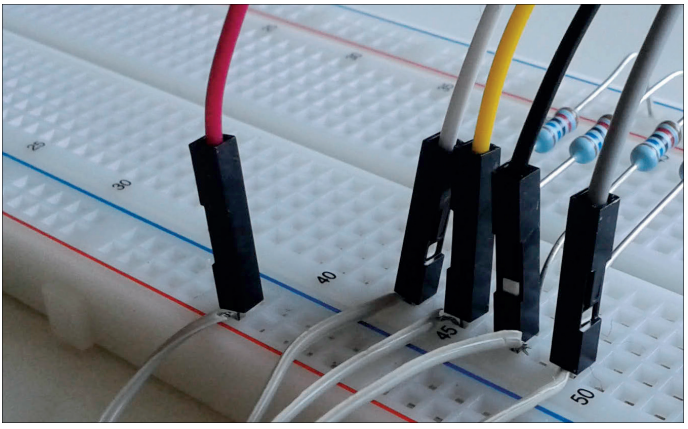


Рис. 10.18. *Изоляция на белых проводах зачищена, оголённые провода вставлены в контактные отверстия и зафиксированы разъёмами типа «папа». К белым проводам подключены контакты считывателя перфокарты*

Программа испытаний

Будет лучше, если ты сначала напишешь маленькую программу для испытаний своего считывателя перфокарты. Следующая программа каждую секунду считывает данные, полученные с контактов считывателя, и отображает их на экране. В следующем примере программы перфокарта не

вставлена. Поэтому все четыре контакта замкнуты (скрепки соприкасаются с фольгой) на общий контакт. Далее вставляется перфокарта.

Ход программы (образец)

```
1111
1100
1100
1111
```

Пишем программу

```
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(8, GPIO.IN)          #1
GPIO.setup(10, GPIO.IN)
GPIO.setup(12, GPIO.IN)
GPIO.setup(16, GPIO.IN)
while True:
    print(GPIO.input(8),
          GPIO.input(10),
          GPIO.input(12),
          GPIO.input(16))       #2
    sleep(1)                    #3
```

Как это работает?

`#1` Контакты 8, 10, 12, 16 настраиваем в качестве входного канала.

`#2` В инструкции `print()` на экране отображается состояние четырёх входных каналов в виде `0` или `1`, а не `False` или `True`. Но для Python-интерпретатора `False` то же, что и `0`, а `True` то же, что и `1`.

`#3` Время ожидания 1 с. Здесь блок завершается.

Вероятно, ты заметишь, что программа в нескольких местах показывает `0`, хотя ты ожидаешь `1`. Кроме того, некоторые скрепки плохо контактируют с фольгой. Чтобы улучшить контакт, нагрузи картонные полоски с клейкой лентой каким-то грузом (например, винтами или 2-рублёвыми монетами).

Электронный ключ

Программа для электронного ключа довольно длинная, но вместе с тем очень простая. Чтобы ознакомиться с ней, не мешало бы взглянуть на диаграмму переходов состояния

(рис. 10.8). Программа содержит бесконечный цикл (#2). Прервать его ты можешь с помощью комбинации клавиш **Ctrl**+**C**.

```
from RPi import GPIO
from time import sleep

GPIO.setmode(GPIO.BOARD)                          #1
GPIO.setup(3, GPIO.OUT)
GPIO.setup(5, GPIO.OUT)
GPIO.setup(8, GPIO.IN)
GPIO.setup(10, GPIO.IN)
GPIO.setup(12, GPIO.IN)
GPIO.setup(16, GPIO.IN)

while True:                                       #2
    GPIO.output(3, 1)                             #3
    GPIO.output(5, 1)
    a0 = GPIO.input(8)                            #4
    a1 = GPIO.input(10)
    a2 = GPIO.input(12)
    a3 = GPIO.input(16)

    while (a0, a1, a2, a3) == (1, 1, 1, 1):       #5
        # ожидание, пока карта будет вставлена
        a0 = GPIO.input(8)                        #6
        a1 = GPIO.input(10)
        a2 = GPIO.input(12)
        a3 = GPIO.input(16)
        # мигает красный светодиод
        GPIO.output(3, 0)                         #7
        sleep(0.2)
        GPIO.output(3, 1)                         #8
        sleep(0.5)                                #9
    sleep(2)                                      #10
    a0 = GPIO.input(8)
    a1 = GPIO.input(10)
    a2 = GPIO.input(12)
    a3 = GPIO.input(16)
    print(a, a1, a3)                              #11
    if (a0, a1, a3) == (1, 1, 0, 0):              #12
        GPIO.output(5, 0)                         #13
    else:
        GPIO.output(3, 0)                         #14

    while (a0, a1, a2, a3) != (1, 1, 1, 1):       #15
        # ожидание, когда карта будет извлечена
        a0 = GPIO.input(8)
        a1 = GPIO.input(10)
        a2 = GPIO.input(12)
        a3 = GPIO.input(16)
        sleep(0.5)
```

Как это работает?

#1 Сначала мы настраиваем GPIO и используем порядок нумерации контактов BOARD. Контакты 3 и 5 назначаются выходными каналами, а контакты 8, 10, 12 и 16 – входными.

#2 Используем бесконечный цикл. Команды для блоков с отступом от левого края должны постоянно повторяться.

#3 На контактах 3 и 5 устанавливается напряжение 3,3 В. Так как на контакте 1, к которому через резисторы номиналом 130 Ом подключены светодиоды, присутствует напряжение 3,3 В, с обеих сторон светодиодов одинаковое напряжение. Поэтому светодиоды не светятся.

#4 В этом шаге запрашивается состояние четырёх входных каналов и сохраняется в переменных `a0`, `a1`, `a2` и `a3`.

#5 Далее проверяется, все ли входные каналы находятся в состоянии `1`. Если ты увидишь все единицы, значит, скользящие контакты (изготовленные из скрепок) хорошо соприкасаются с общим контактом (фольга на основании картоприёмника). Если вместо `1` ты увидишь `0`, проверь, у всех ли скрепок хороший контакт. То есть на всех четырёх входных каналах (контакты 8, 10, 12 и 16 GPIO) должно быть напряжение 3,3 В (состояние `1` или `True`).

#6 Вновь опрашиваются все четыре входных канала.

#7 На контакте 3 напряжение устанавливается в ноль. Ток течет через контакт 1, резистор, красный светодиод и контакт 3, и светодиод светится.

#8 Через 0,2 с напряжение снова устанавливается на уровне 3,3 В. Ток течь прекращает, и светодиод гаснет. То есть это был один цикл мигания светодиода.

#9 Цикл завершается, и через 0,5 с со строки `#5` начинается новый цикл.

#10 Цикл приостановлен на 2 с. Это произойдёт, если кто-то вставил перфокарту в считыватель, и контакт разомкнулся. В этом случае программа ждёт 2 с, за которые перфокарта должна быть вставлена до упора.

#11 Для контроля состояние всех четырёх входных каналов выводится на экран. Эту строку ты можешь удалить, если программа работает хорошо.

#12 Здесь состояние четырёх входных каналов сравнивается с кодом `(1, 1, 0, 0)`. Это и есть секретный сохранённый код. Разумеется, ты можешь использовать совершенно другой электронный код.

#13 Если перфокарта имеет правильный код, жёлтый светодиод загорится. (Вместо светодиода ты можешь подключить исполнительное устройство, например мощный электромагнит (через соответствующую схему управления мощностью), который будет открывать или закрывать дверь или замок, засов которого будет двигать электродвигатель (естественно, через схему управления мощностью).)

#14 В остальных случаях загорается красный светодиод.

#15 Этот цикл выполняется до тех пор, пока карта не будет извлечена и все сенсоры (скрепки) снова не соприкоснутся с фольгой. Затем бесконечный цикл начинается сначала со строки (#2). Считыватель ожидает следующую карту.

Вопросы

1. Какова задача нагрузочного резистора у входного канала GPIO?
2. Какие два контакта (пина) имеют встроенный нагрузочный резистор?
3. Сколько различных шаблонов ты можешь запрограммировать с помощью комбинации замкнутых и разомкнутых четырёх контактов, если перфокарта должна использоваться в качестве «электронного ключа»?
4. Какова сила проходящего тока, если контакт 10 GPIO сконфигурирован в качестве входного канала и его соединяют с контактом 1 (с напряжением 3,3 В)? Измерь ток с помощью мультиметра.

Задания

Задание 1. Управляемый световой сигнал

Дверь твоей комнаты должна быть оснащена аварийным световым сигналом. Всякий раз, когда она открыта, светодиод мигает. Используй переключатель двери как схему из рис. 10.19 и напиши подходящую программу Python. Только если переключатель открыт, светодиод должен мигать. В остальных случаях нет.

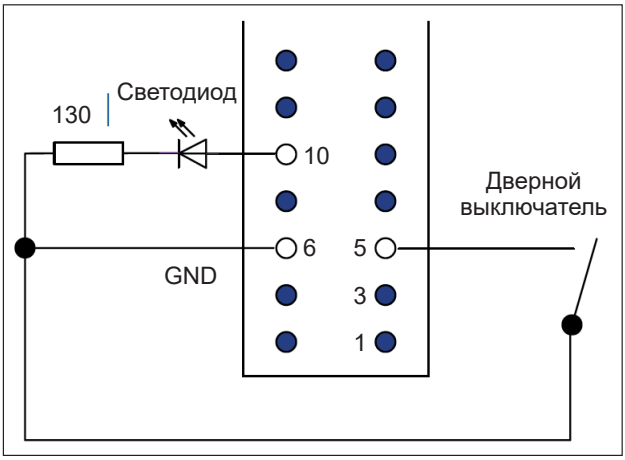


Рис. 10.19. *Принципиальная схема для управляемого светового сигнала. Светодиод горит, если на контакте 10 положительное напряжение*

Задание 2. Музыкальная шкатулка (автомат)

Музыкальный автомат – это машина, проигрывающая музыку, которую можно выбрать из предлагаемого меню. Напиши интерактивную программу, которая реализует функции музыкального автомата в плане звуковых эффектов.

Ход программы

```
Выбрать музыкальный файл. Заверши выбор, нажав клавишу <ENTER>
(А)плодисменты
(В)осторг
(Л)ай
(Ф)анфары
Выбор: а
```

Ответы на вопросы

1. Нагрузочный резистор находится между входным портом и корпусом (GND, ground, или земля). Он устанавливает напряжение на входном порту 0 В (`False`), если положительное напряжение не подаётся извне (через выключатель).
2. Контакты (пины) 3 и 5 имеют встроенный нагрузочный резистор, который обеспечивает положительный уровень напряжения (истинное значение, или `True`), если

контакт (пин) подключён в качестве входного порта и не соединён через провод с корпусом.

3. В принципе, возможны 16 шаблонов (2*2*2*2). Однако шаблон 1111 (четыре отверстия) по причине того, что это тестовый шаблон, проверяющий, все ли скользящие контакты хорошо соприкасаются с общим контактом (фольгой), использован быть не может. В режиме ожидания перфокарты все четыре контакта замкнуты на общий контакт. Таким образом, для применения доступны 15 шаблонов.
4. Сила тока около 2 мА.

Решение задач

Решение 1. Управляемый световой сигнал

Пишем программу

```
# световой сигнал двери.py
import os
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(5,GPIO.IN)
GPIO.setup(10,GPIO.OUT)

while True:                    #1
    while GPIO.input(5):       #2
        GPIO.output(10,True)   #3
        sleep(0.5)
        GPIO.output(10,False)  #4
        sleep(0.5)
```

Как это работает?

#1 Бесконечный цикл.

#2 Когда дверь открыта, контакты выключателя разомкнуты. Контакт 5 из-за встроенного нагрузочного резистора имеет высокий уровень напряжения (истинное значение `True`). Поэтому условие выполняется, когда дверь открыта. Значит, выполняется внутренняя часть цикла (вложенный блок).

#3 Светодиод включается.

#4 Светодиод выключается.

Решение 2. Музыкальный автомат

Пишем программу

Прежде всего найди в интернете нужные звуки в формате *.wav* и сохрани их в созданной ранее папке *sound*. Переименуй файлы так, чтобы их имена обязательно совпадали с именами файлов в строках #1а, #2а, #3а и #7.

Введи код программы, обращая внимание на отступы от левого края.

```
import os
выбор = 'а'                                              #1
while выбор != 'q':                                      #2
    print()                                              #3
    print('Выбери аудиофайл, нажав нужную клавишу,
          и нажми клавишу <Enter>.')                     #4
    print('(А)плодисменты')
    print('(В)осторг')
    print('(Л)ай')
    print('(Ф)анфары')
    выбор = input('выбор: ')                             #5
    if выбор == 'а':                                     #6
        os.system('aplay sounds/аплодисменты.wav')       #1а
    if выбор == 'в':
        os.system('aplay sounds/восторг.wav')            #2а
    if выбор == 'л':
        os.system('aplay sounds/лай.wav')                #3а
    if выбор == 'ф':
        os.system('aplay sounds/фанфары.wav')            #7
    print('До свидания.')                                #8
```

Как это работает?

#1 В переменной `выбор` позднее сохраняются введённые буквы. Но пустая переменная работать не будет, поэтому изначально мы вводим любое значение, например `а`. Иначе цикл `while` не начнёт свою работу.

#2 Начало цикла `while`. Он повторяется до тех пор, пока пользователь не введёт что-то, отличное от `q`. Если нажать клавишу **q**, работа программы завершится.

#3 Ввод пустой строки.

#4 Отображается меню, в котором объясняется ход твоих действий.

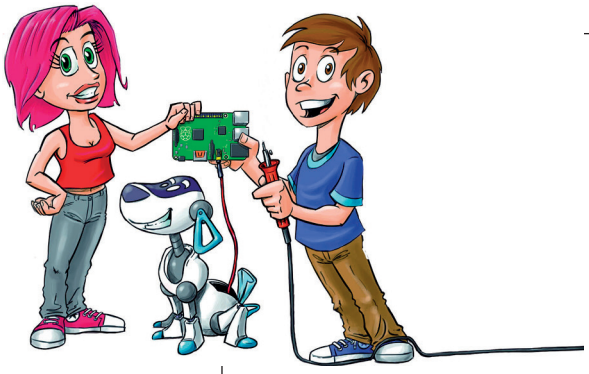
#5 Ожидание ввода пользователя. Важно, чтобы данные вводились прописными (маленькими) буквами, иначе условие не будет выполнено.

`#6` Если буква `a` введена, звуковой файл начнёт воспроизведение файла, связанного с этой кнопкой.

`#7` Это последняя строчка внутренней части цикла.

`#8` Завершение работы программы. Если пользовать нажал клавишу **q** и клавишу ↵, программа прощается с пользователем и завершает свою работу. Эта строка не относится больше к внутренней части цикла.

Ещё раз обрати внимание на имена аудиофайлов, которые должны воспроизводиться при нажатии нужной клавиши. Ты файлы можешь переименовать, присвоив им имена с цифрами. Например, назвать *01.wav*, *02.wav*, *03.wav* и *04.wav*. В этом случае ты в строке `#1a` вводишь имя не *аплодисменты.wav*, а *01.wav*. Кстати, ты можешь скачать четыре песни в формате *.wav*, переименовать их *01.wav*, *02.wav*, *03.wav* и *04.wav*. В соответствующих строках кода программы введи имена этих файлов. В этом случае при нажатии нужной клавиши будет воспроизводиться определённая песня.



11
Светодиодные дисплеи

Большие и яркие дисплеи состоят из большого количества светодиодов. Медиафасад здания Музея современного искусства в Дортмунде (рис. 11.1) – это один из ярких примеров такого светодиодного панно.

В этой главе мы создадим светодиодный индикатор, используя проволоку, картон и светодиоды. На этом индикаторе мы отобразим сначала статическое (неподвижное), а затем и динамическое (подвижное) изображение. Если ты не хочешь или не умеешь паять, можно использовать светодиодный точечный матричный дисплей.

Рис. 11.1. *Огромный светодиодный дисплей: медиафасад, венчающий здание Дортмундер U*

Проект 27. Светодиодная матрица

Если ты отправишься в город Дортмунд на поезде, то на крыше здания Дортмундер U увидишь гигантский дисплей. Он состоит из 1 700 000 очень ярких светодиодов и имеет общую площадь размером 625 м^2. Этот гигантский светодиодный матричный дисплей работает в открытой окружающей среде. На дисплее днём и ночью транслируют цветные анимации: пламя, аквариум с плавающими рыбками, летающих голубей.

При выключенных светодиодах дисплей становится невидимым, и можно увидеть конструкции крыши, расположенные позади светодиодного матричного дисплея. С большого расстояния тёмные светодиоды представляют собой маленькие неприметные точки, между которыми много места.

В этом проекте мы построим миниатюрную версию такого светодиодного фасада, состоящего из 16 светодиодов. Но принцип управления светодиодной матрицей такой же, как и в случае с крупным объектом. На рис. 11.2 показан принцип работы светодиодной матрицы. Матрица состоит из горизонтальных контактных линий Z0–Z3 и вертикальных контактных линий S0–S3. Эти контактные линии между собой напрямую не соединены, но на их пересечении припаяны светодиоды. Анод (вывод светодиода, на который подаётся положительное напряжение) припаивается к вертикальной контактной линии. Катод (отрицательный (минусовой) вывод светодиода) припаивается к горизонтальной контактной линии. Направление движения электрического тока – положительный контакт источника питания – анод (положительный вывод светодиода) – катод (отрицательный вывод светодиода) – отрицательный контакт источника питания. В нашем случае, если на контакте разъёма расширения ввода-вывода, подключённого к вертикальной контактной линии, присутствует положительное напряжение (`True`), а контакт разъёма расширения ввода-вывода, подсоединённый к горизонтальной контактной линии, подключен к массе (`False`) через светодиод, подсоединённый к этим двум контактным линиям, потечёт электрический ток, и этот светодиод будет светиться. На рис. 11.2 вертикальная контактная линия, на которую подано положительное напряжение (состояние вывода – `True`), окрашена в красный цвет. Горизонтальная контактная линия, подсоединённая к контакту разъёма расширения ввода-вывода, который подключён на корпус (масса), окра-

шена в голубой цвет. На пресечении этих двух контактных линий будет светиться подключённый к ним светодиод (на рис. 11.2 этот светодиод обозначен зелёным цветом). То есть светодиод светит только тогда, когда на его аноде установлен высокий уровень напряжения (`True`), а на катоде – низкий уровень напряжения (`False`).

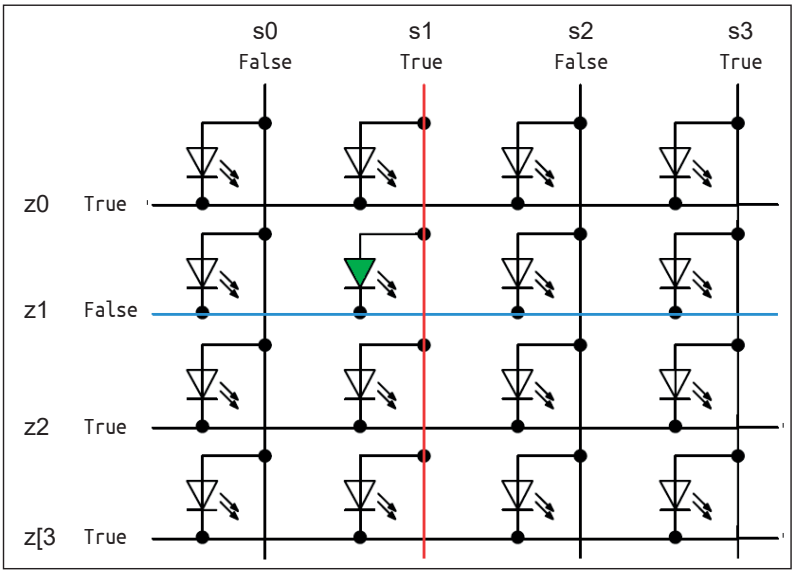


Рис. 11.2. *Принцип работы светодиодной матрицы*

Аппаратное обеспечение

В этом разделе описание того, как самостоятельно собрать светодиодную матрицу размером 4×4. Если желания возиться с паяльником и ножницами у тебя нет, ты можешь купить готовую матрицу типа TA07. В разделе «Светодиодный модуль», который ты прочитаешь далее, ты узнаешь, как подключить эту светодиодную сборку. Но программный код для этого раздела прекрасно работает и со светодиодной сборкой модели TA07. Если ты будешь использовать эту деталь, тебе понадобится лишь изменить пару чисел.

Итак, для самостоятельной сборки светодиодной матрицы тебе потребуется:

- 16 светодиодов;
- 4 резистора номиналом 100 Ом;
- один плоский шлейф с 40 жилами (джампер-кабель (female-female)) и 40-контактный разъём подключения к разъёму расширения ввода-вывода RPi или 26-контактный разъём для старой модели RPi;

- примерно 2 м одножильного медного провода для дверного звонка или одножильного монтажного провода;
- толстый гофрированный картон и клейкая лента;
- ножницы;
- олово и канифоль (вместо канифоли можно использовать флюс (размельчённая в ступке канифоль, разведённая спиртом);
- паяльник;
- деревянная доска для подставки;
- плоский кусок пенопласта (10×10 см);
- дополнительно провода с разъёмами «папа-мама».

Для начала вырезаешь из картона квадратную рамку размером 20×20 см (рис. 11.3).

На одной стороне предварительно проткни шилом или другим острым предметом маленькие отверстия и вставь в них резисторы (рис. 11.3). Несмотря на то что у нас 16 светодиодов, нам для строчек матрицы вполне достаточно четырёх резисторов. К резисторам мы подключим горизонтальные контактные линии. Ток, протекающий через светодиоды, будет ограничиваться этими резисторами.

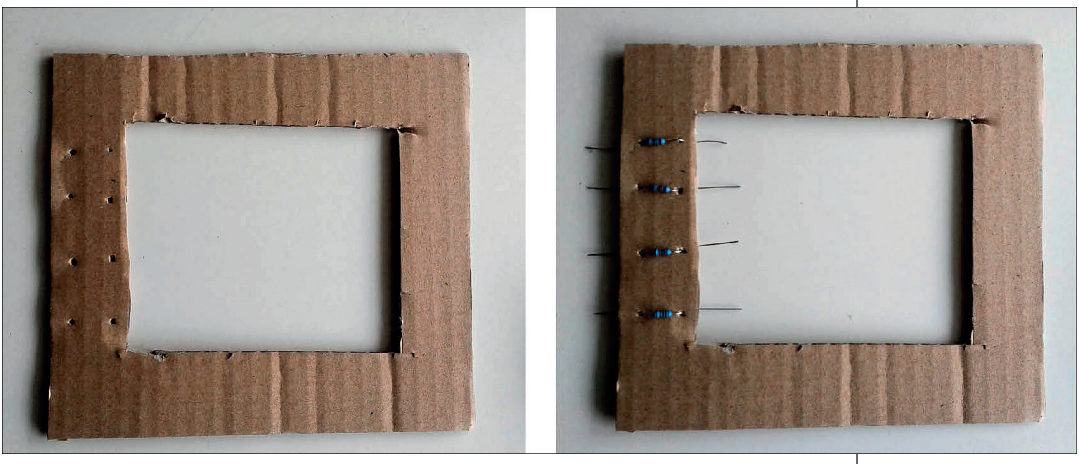


Рис. 11.3. *Рамка для светодиодной матрицы*

Отрежь от одножильного медного монтажного провода (или провода для дверного звонка) восемь кусков длиной примерно 25 см. Они должны быть чуть длиннее картонной рамки. Аккуратно удали изоляцию с этих проводов. Тебе понадобится сердцевина этого провода, очищенная от изоляции (блестящий медный провод).

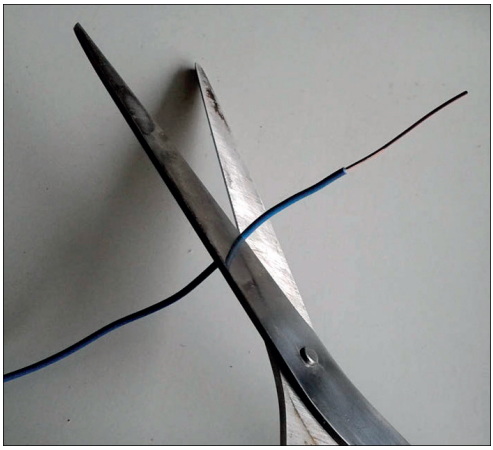


Рис. 11.4. *Удаление изоляции по частям с помощью ножниц*

Проделай отверстия в рамке, через которые ты вставишь медный провод (рис. 11.5). Провода для горизонтальных контактных линий расположены в плоскости картонной рамки и соединены с резисторами. Провода для вертикальных контактных линий приподняты над плоскостью рамки и как бы образуют мосты. Они не должны касаться проводов горизонтальных линий.

Наклей по краям клейкую ленту, чтобы закрепить провода и защитить их.

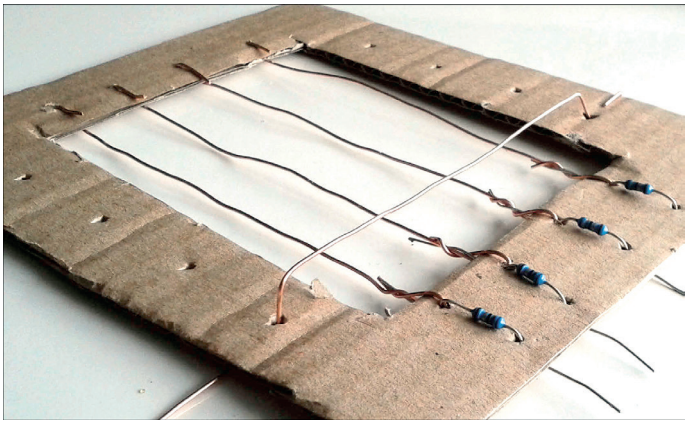


Рис. 11.5. *Медные провода для строк и столбцов матрицы*

Теперь будем паять. Положи на деревянную доску квадратный кусок пенопласта, а поверх него матрицу. Возьми один

светодиод и держи его головкой вверх таким образом, чтобы слева находился короткий соединительный вывод (катод), а справа – более длинный вывод (анод). Согни конец длинного вывода (анода) примерно на расстоянии 5 мм от основания светодиода под прямым углом к себе. Вставь светодиод коротким концом (катодом) в пенопласт так, чтобы короткий вывод (катод) касался провода матрицы, а изогнутый вывод (анод) касался вертикальной контактной линии, поднятой над плоскостью картонной рамки (не подключённой к резисторам). Точно так же подготовь и остальные три светодиода и вставь их в матрицу. Всё должно выглядеть как на рис. 11.6. Чтобы не ошибиться полярностью светодиода, следи за тем, чтобы катод (короткий вывод) всегда находился слева. Иначе неправильно впаянный светодиод светиться не будет. Когда все четыре диода будут вставлены, подпаяй анодные выводы к вертикальным контактным линиям.

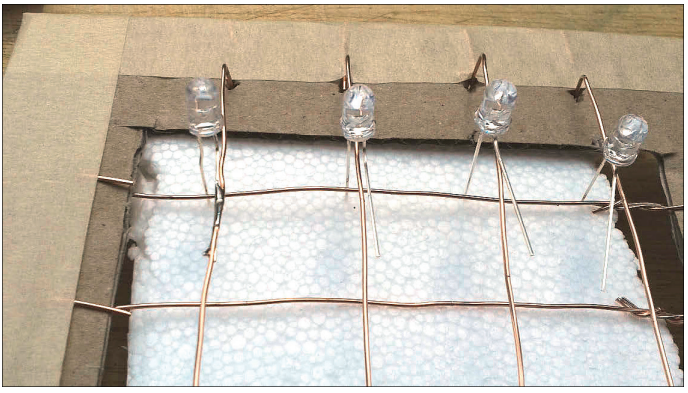


Рис. 11.6. *Впаиваем светодиоды. Катоды (короткие выводы) находятся слева и вставлены в пенопласт*

Подпаяй таким образом все 16 светодиодов, распределив их на пересечениях вертикальных и горизонтальных контактных линий. Когда аноды всех 16 светодиодов будут подпаяны, переверни матрицу, согни концы катодов и припаяй их к проводам горизонтальных контактных линий.

Горизонтальные и вертикальные контактные линии необходимо соединить с контактами разъёма расширения ввода-вывода GPIO. Для этого тебе понадобится джампер-кабель (female-female). Ввиду размеров конструкции, возможно, гораздо практичнее будет использовать плоский кабель с разъёмом.

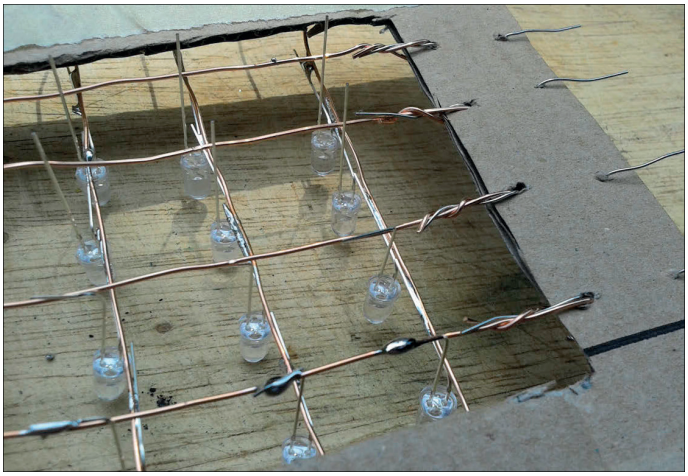


Рис. 11.7. *На обратной стороне катодные выводы припаяны к проводам матрицы*

Соединение с плоским ленточным кабелем

Для модели RPi B+ тебе нужен 40-контактный штырьковый разъём. Для более старой модели RPi с меньшим количеством контактов разъёма расширения ввода-вывода GPIO ты можешь использовать 26-полюсный штырьковый разъём. Такой разъём состоит из двух частей: основной части с контактными отверстиями, которые являются ответной частью GPIO, прижима с рифлёной поверхностью и второго прижима для фиксации.

- Вытяни вверх оба прижима так, как показано на рис. 11.8 слева. Верхний прижим разгрузки натяжения сними.
- Посмотри на этот штырьковый соединитель внимательнее. В одном месте ты увидишь маленький треугольник. Пометь его как Pin 1.
- Уложи плоский кабель на иголки разъёмов, как показано на рис. 11.8 втором слева. Проследи, чтобы крайняя жила шлейфа, окрашенная в красный цвет, оказалась на одной стороне с изображением маленького треугольника, выбитого на разъёме. Затем надави на прижим так, чтобы иглы разъёмов вонзились в изоляцию каждой жилы шлейфа. Теперь тебе необходимы тиски. Если у тебя их нет, спроси у соседей или поезжай на строительный рынок. Там, чаще всего рядом с кассой, находятся тиски для клиентов, которые собираются пилить длинные бруски. В крайнем случае, попроси помощи у сотрудников! Обычно к юным умельцам

они настроены очень дружелюбно. Обожми разъём со вставленным шлейфом, аккуратно зажав конструкцию в губках тисков. Обрати внимание: детали конструкции пластиковые, и их можно раздавить. Сжимай разъём с кабелем в тисках, пока иглы полностью не зайдут в шлейф. После обжатия кабель должен выглядеть как на второй слева фотографии рис. 11.8.

- Установи прижим разгрузки натяжения как на третьей слева фотографии рис. 11.8.
- Протолкни длинный конец шлейфа через образовавшуюся щель между прижимом, держащим шлейф на иглах контактов, и прижимом разгрузки и зажми прижим разгрузки. Собранная конструкция должна выглядеть как на рис. 11.8 справа.

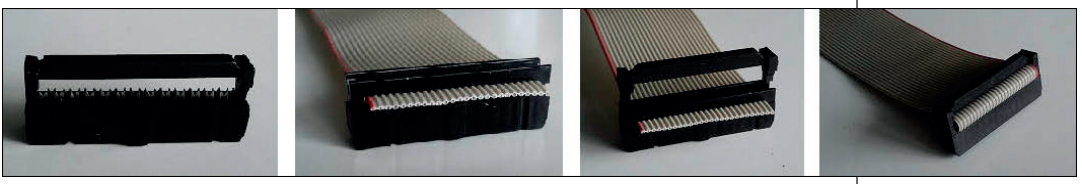


Рис. 11.8. *Закрепление штырькового соединителя*

- Запиши на картонной рамке матрицы нумерацию контактов GPIO, к которым ты хочешь подключить светодиоды. Я использовал для горизонтальных контактных линий контакты 8, 10, 11, 12, а для вертикальных контактных линий – контакты 19, 16, 15 и 13.

Эти данные очень важны, так как при написании программы нам будет нужно правильно запрограммировать каждый контакт.

- Отсчитай 7 жил, взяв за начало отсчёта красную жилу, обозначающую контакт 1 разъёма GPIO, и отдели восьмую жилу от шлейфа так, чтобы отдельная жила была длиной около 20 см (рис. 11.9). Зачисти конец этой жилы примерно на 1 см от изоляции. Изоляция очень мягкая, и ты можешь зачистить конец отделённого провода просто ногтями большого и указательного пальцев.
- Залуди и припаяй очищенный конец этой жилы к свободному выводу резистора первой сверху горизонтальной контактной линии (помеченной на картонке цифрой 8) (рис. 11.9).
- Отдели десятый провод от шлейфа и припаяй его ко второй сверху контактной горизонтальной линии, отмеченной цифрой 10.

Обрати внимание: мы выбрали для горизонтальных контактных линий контакты 8, 10, 11 и 12. Так как жилы шлейфа уже подключены к этим контактам на разъёме, нам нужно выбирать именно эти номера жил.

- Таким же способом подпаяй вертикальные контактные линии к проводам шлейфа, учитывая, что вертикальные линии подключаются к контактам 13, 15, 16 и 19.
- Места пайки заизолируй изолентой или бумажной клейкой лентой.
- Остальные 32 жилы загни под шлейф и приклей клейкой лентой так, чтобы они тебе не мешали (рис. 11.9).

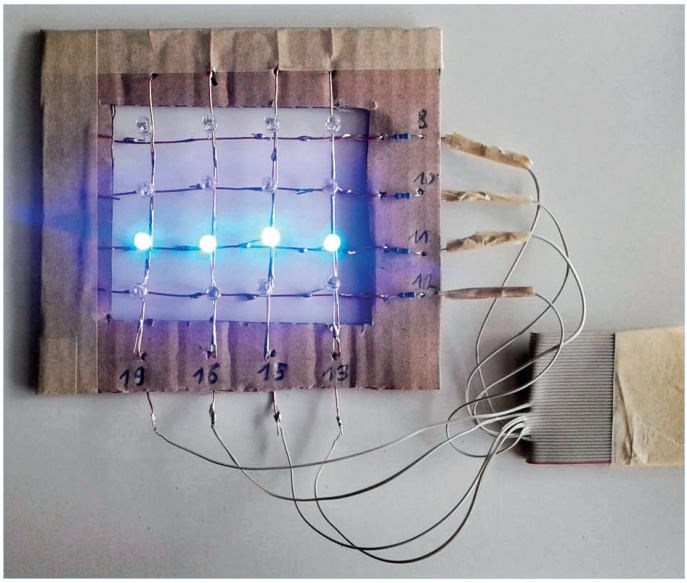


Рис. 11.9. *Готовая модель светодиодной матрицы в действии. На картонной рамке указана нумерация контактов. Горизонтальные контактные линии: 8, 10, 11, 12. Вертикальные контактные линии: 19, 16, 15, 13*

Проект 28. Перемещающиеся светящиеся линии

После того как светодиодная матрица собрана, нам нужно проверить работу всех светодиодов. При присутствии положительного напряжения на одной из вертикальных контактных линий эта линия, состоящая из четырёх светодиодов, должна светиться. То есть светодиодная линия све-

тится, когда на контакте GPIO, к которому она подключена, присутствует положительное напряжение (`True`), а на контактах 8, 10, 11, 12, к которым подключены горизонтальные контактные линии, присутствует состояние `False`, и эти контакты соединены с массой. Сперва по циклу светится верхняя линия, далее – вторая, затем – третья, после – четвёртая, после чего цикл повторяется, и все начинается сначала. То есть всё должно выглядеть так, словно светящиеся линии передвигаются сверху вниз (рис. 11.9).

Итак, чтобы написать программу, сначала запусти оболочку IDLE 3 под правами администратора:

```
$ sudo idle3
```

Пишем программу

Большинство инструкций похожи друг на друга. В основном они содержат вызов функций `GPIO.setup()` и `GPIO.output()`. Соответственно, ты напишешь такие инструкции всего один раз, а потом просто скопируешь. Копии потом можно легко преобразовать.

```
# moving_luminous_lines.py
from time import sleep
from RPi import GPIO
GPIO.setmode(GPIO.BOARD)          #1

# Горизонтальные контактные линии
GPIO.setup(8, GPIO.OUT)           #2
GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
# Вертикальные контактныне линии
GPIO.setup(19, GPIO.OUT
GPIO.setup(16, GPIO.OUT)
GPIO,setup(15, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)

# Светодиоды отключить            #3
GPIO.output(8, True)
GPIO.output(10, True)
GPIO.output(11, True)
GPIO.output(12, True)
GPIO.output(19, True)
GPIO.output(16, True)
GPIO.output(15, True)
GPIO.output(13, True)
```

```
while True:                          #4
    GPIO.output(8, False)            #5
    sleep(0.2)                       #6
    GPIO.output(8, True)             #7
    GPIO.output(10, False)           #8
    sleep(0.2)
    GPIO.output(10, True)
    GPIO.output(11, False)
    sleep(0.2)
    GPIO.output(11, True)
    GPIO.output(12, False)
    sleep(0.2)
    GPIO.output(12, True)
```

Как это работает?

#1 С помощью модуля BOARD мы назначаем контакты GPIO согласно его позиции.

#2 Все подключённые контакты GPIO назначаются как входные контакты.

#3 Для всех горизонтальных и вертикальных контактных линий и устанавливается высокий уровень напряжения (+3,3 В), или состояние `True`. Таким образом, на каждом выводе всех светодиодов присутствует одинаковое положительное напряжение, разницы в напряжении нет, и все светодиоды не светятся.

#4 Начало бесконечного цикла.

#5 На горизонтальной контактной строке (подключённой к контакту 8) уровень напряжения установлен на 0 В (состояние контакта 8 – `False`). Поскольку на этом контакте, к которому подключена вертикальная контактная линия, присутствует состояние True, ток течёт через светодиоды этой линии, и они светятся (рис. 11.10).

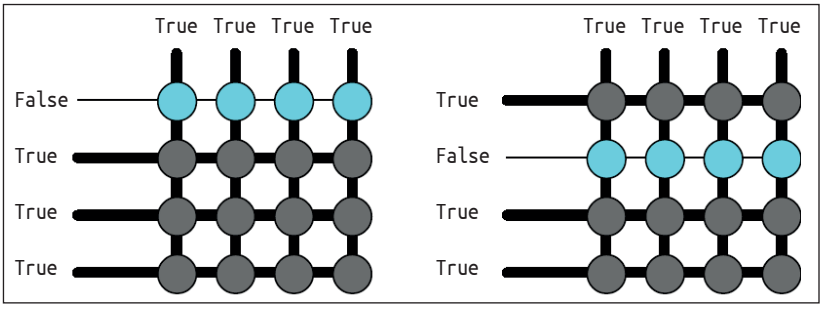


Рис. 11.10. *Так появляются перемещающиеся светящиеся линии. Контакты, к которым подключены горизонтальные контактные линии, попеременно переходят в состояние `False`*

#6 Время ожидания 0,2 с.

#7 Состояние контакта 8, к которому подключена горизонтальная контактная линия, снова переходит в состояние `True`, и на нём устанавливается положительное напряжение 3,3 В.

#8 Светодиоды, подключенные ко второй горизонтальной линии, выключены. Далее включаются светодиоды, подключённые к третьей горизонтальной контактной линии, и т. д.

Эта программа работает в режиме бесконечного цикла. Чтобы завершить работу программы, нажми комбинацию клавиш **Ctrl**+**C**.

Блок со светодиодным матричным индикатором

Матричная светодиодная сборка модели TA07-11HWA фирмы «Кингбрайт» (Kingbright) очень популярна в среде людей, которые любят что-то делать своими руками. Сборка содержит семь строчек, в каждой из которых находится пять красных светодиодов. То есть в матрице семь горизонтальных строк и пять вертикальных столбцов:

- матричная светодиодная сборка (TA07-11HWA);
- пять резисторов номиналом 130 Ом;
- монтажная плата и джампер-кабель.

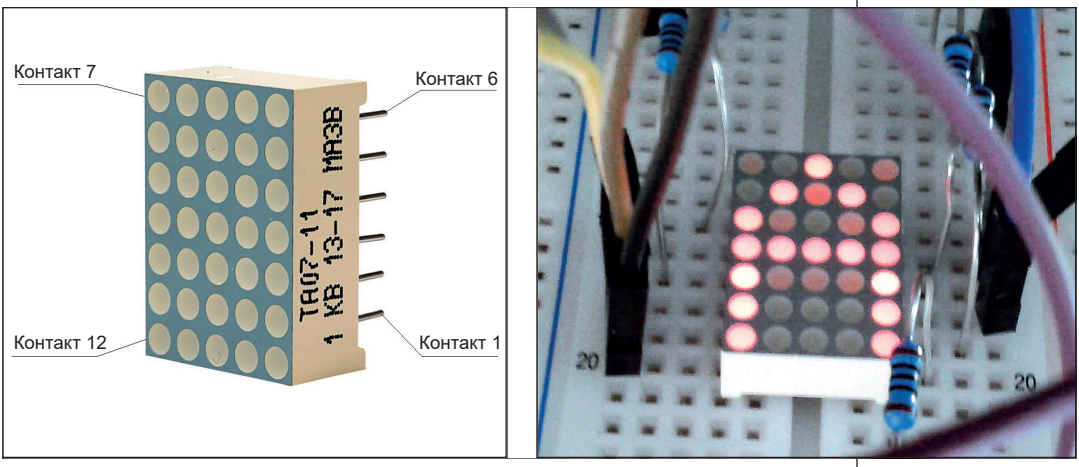


Рис. 11.11. *Промышленный функциональный матричный дисплей*

- Установи матричную светодиодную сборку в центре монтажной платы (рис. 11.11 справа) и подключи через резистор номиналом 130 Ом проводом с разъёмами «папа-мама» первый контакт матрицы с контактом 3 разъёма GPIO (рис. 11.12).

Контакт 1 на светодиодной матрице отмечен цифрой «1» на одной из сторон детали (рис. 11.11 слева).

- Собери остальную схему согласно рис. 11.12. Строго придерживайся номеров контактов.

Обрати внимание: расположение контактов в шахматном порядке не отражает структуру из вертикальных и горизонтальных линий светодиодов.

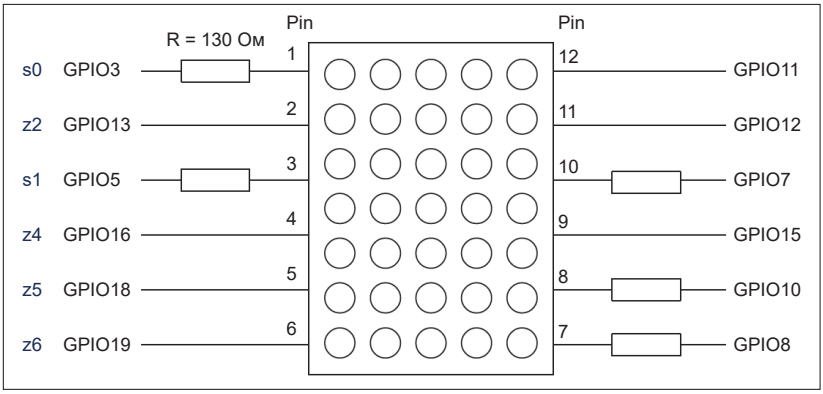


Рис. 11.12. *Подключение матричного индикатора TA07-11HWA к GPIO (модуль BOARD)*

Проект 29. Управление отдельными светодиодами

Сначала программа выводит графику в виде текста на экран, в которой представлены горизонтальные и вертикальные строки матричной светодиодной сборки.

С клавиатуры можно ввести только два номера контактов GPIO. Первый номер относится к горизонтальной строке, а второй – к вертикальному столбцу матричной светодиодной сборки. После ввода номера контакта строки и столбца на одну секунду загорается светодиод, находящийся на пересечении этой строки и столбца. Затем программа запрашивает, нужно ли тестировать остальные светодиоды.

Ход программы (образец)

```
    3  5  7  8  10
11 -+--+--+--+--+
12 -+--+--+--+--+
13 -+--+--+--+--+
15 -+--+--+--+--+
16 -+--+--+--+--+
18 -+--+--+--+--+
19 -+--+--+--+--+
Строчки: 13
Столбцы: 5
Еще раз? (д/н): н
До свидания…
```

Пишем программу

Идея программы состоит в том, чтобы на контакте GPIO, к которому подключена выбранная строка светодиодной матрицы, установить значение `False`. То есть этот контакт подключается к массе, а контакты, к которым подключены остальные строки матрицы, имеют значение `True` (+ 3,3 В). Также контакт, к которому подключён выбранный столбец, будет единственным со значением `True`. Тогда будет светиться только один светодиод, расположенный в точке пересечения выбранной строки и столбца (рис. 11.13).

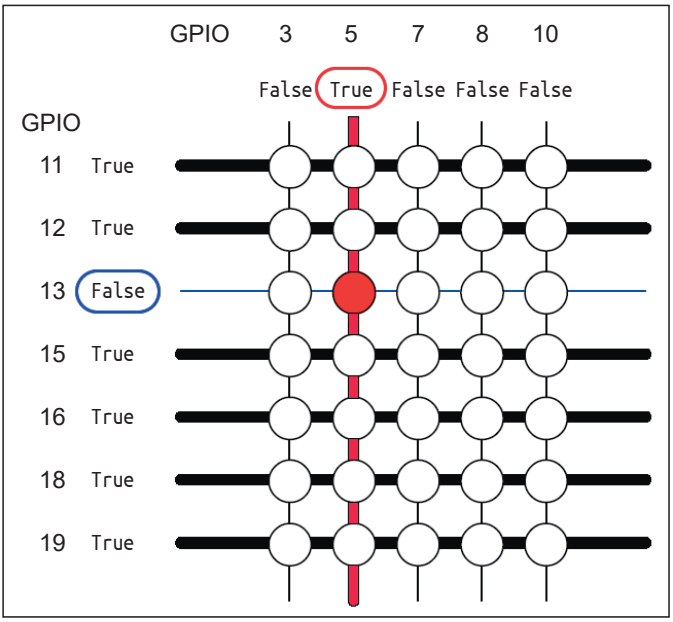


Рис. 11.13. *Включаем светодиод*

Ниже приведён код программы для тестирования светодиода, которую следует сохранить под именем *testing_the_led.py*.

```
# testing_the_led.py
from time import sleep
from RPi import GPIO
GPIO.setmode(GPIO.BOARD)

#lines
GPIO.setup(11, GPIO.OUT)                #1
GPIO.setup(12, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(19, GPIO.OUT)

#columns
GPIO.setup(3, GPIO.OUT)
GPIO.setup(5, GPIO.OUT)
GPIO.setup(7, GPIO.OUT)
GPIO.setup(8, GPIO.OUT)
GPIO.setup(10, GPIO.OUT)

# Светодиоды выключить
GPIO.setup(11, True)                    #2
GPIO.setup(12, True)
GPIO.setup(13, True)
GPIO.setup(15, True)
GPIO.setup(16, True)
GPIO.setup(18, True)
GPIO.setup(19, True)                    #3

GPIO.setup(3, False)
GPIO.setup(5, False)
GPIO.setup(7, False)
GPIO.setup(8, False)
GPIO.setup(10, False)

ввод = 'д'                              #4
while ввод == 'д':                      #5
    print('''
    3  5  7  8  10
11 -+--+--+--+--+
12 -+--+--+--+--+
13 -+--+--+--+--+
15 -+--+--+--+--+
16 -+--+--+--+--+
18 -+--+--+--+--+
19 -+--+--+--+--+''')                   #6
```

```
    lines = int(input('Строчки: '))
    columns = int(input('Столбцы: '))
    GPIO.output(lines, False)          #7
    GPIO.output(columns, True)
    sleep(1)
    GPIO.output(lines, True)           #8
    GPIO.output(columns, False)
    ввод = input('Еще раз? (д/н): ')
print('До свидания…')                  #9
```

Как это работает?

#1 Все контакты GPIO подключаются в качестве выходных каналов.

#2 Все контакты, к которым подключены строки светодиодной матрицы, устанавливаются в значение `True` (+3,3 В)…

#3 … а все контакты, к которым подключены столбцы светодиодной матрицы, устанавливаются в значение `False` (0 В). В этом случае все светодиоды выключены.

#4 Для того чтобы следующий цикл выполнялся хотя бы один раз, переменная `ввод` должна содержать значение `'д'`.

#5 Следующий блок повторяется, пока у переменной `ввод` имеется значение `'д'`.

#6 С помощью этой команды отображается длинный текст, который может быть разбит на несколько строк. Разбиение длинного текста на несколько строк задаётся тремя кавычками в начале и конце текста.

#7 Здесь с помощью команд `lines = int(input('Строчки: '))` и `columns = int(input('Столбцы: '))` выбирается нужная строка и столбец. То есть переменными `lines` и `columns` назначается значение, определяющее номер контактов, к которым подключены нужная строка и столбец. Далее с помощью команд `GPIO.output(lines, False)` и `GPIO.output(columns, True)` контакт, к которому подключена выбранная строка, устанавливается в состояние `False`, а контакт с подключённым нужным столбцом переходит в состояние `True`. На пересечении выбранной строки и столбца должен засветиться светодиод. Команда `sleep(1)` задаёт время, в течение которого выбранный светодиод будет светиться. В данном случае время свечения светодиода равно одной секунде.

#8 Через секунду светодиод вновь отключается. Выбранные контакты GPIO возвращаются в исходное состояние.

#9 По завершении цикла `while` программа прощается.

Вечно одно и то же!

Ты что-нибудь заметил? В этих программах очень часто повторяется несколько видов инструкций с небольшими изменениями. До чего же скучно! Можно было сделать гораздо изысканнее, используя коллекции и итерации. Как это выполнить, ты узнаешь из следующей главы.

Вопросы

1. Светодиод подключён катодом к контакту 8, а анодом – к контакту 10. В каких логических состояниях должны находиться оба контакта GPIO, чтобы светодиод начал светиться?
2. Если все контакты светодиодной матрицы имеют логическое состояние `True` (+3,3 В), все светодиоды выключены. Что произойдёт, если все разъёмы будут иметь логическое состояние `False`?
3. Сколько загорится светодиодов, если контакты, к которым подключены два столбца, находятся в состоянии `True`, а контакты, к которым подключены две строки, находятся в состоянии `False`?
4. У тебя есть светодиодная матрица, состоящая из 4000 светодиодов, расположенных в 50 строках (линиях) и 80 столбцах. Сколько резисторов тебе понадобится для работы?

Задания

Задание 1. Затемнение (диммирование) светодиода

Как вообще можно изменить яркость свечения светодиодов? Очень просто! Для этого их нужно очень быстро включить и выключить. Чем дольше время ожидания между вспышками, тем меньше яркость свечения светодиодов. Взгляни на рис. 11.14. Если светодиод включать с частотой, показанной на осциллограмме слева, яркость свечения светодиода будет меньше, чем согласно правой осциллограмме.

Рис. 11.14. *Яркость свечения светодиода регулируется длиной импульса, включающего светодиод, и временем между этими импульсами, когда светодиод не светится*

Теперь задание. Подключи светодиод через резистор в 130 Ом анодом к контакту GPIO 10, а катодом – к контакту GPIO 6 (корпус). Выключатель соединяется с контактами GPIO 5 и 6 (корпусом).

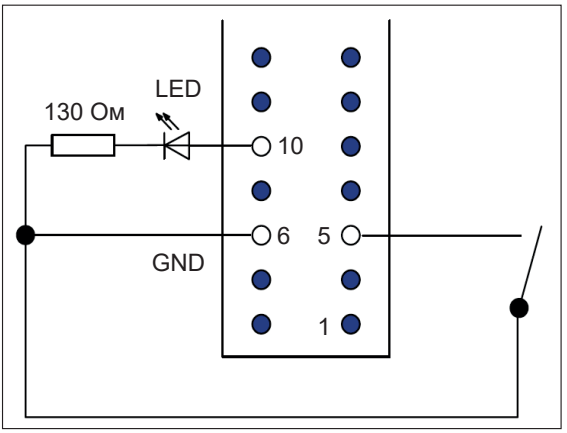


Рис. 11.15. *Диммирование светодиода с помощью выключателя*

Напиши программу, которая будет делать следующее. Всякий раз, когда контакты выключателя разомкнуты, яркость свечения светодиода уменьшается. Как только контакты выключателя замыкаются, светодиод вновь ярко светится.

Совет. Используй эту структуру программы и дополни недостающими инструкциями.

```
from RPi import GPIO
from time import sleep
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.OUT)
GPIO. setup(5, GPIO.IN)
while True:
    ...
```

Задание 2. Правой, левой

На светодиодной матрице поочерёдно должны светиться левая и правая стороны светодиодов (рис. 11.16). Напиши подходящую для этого программу.

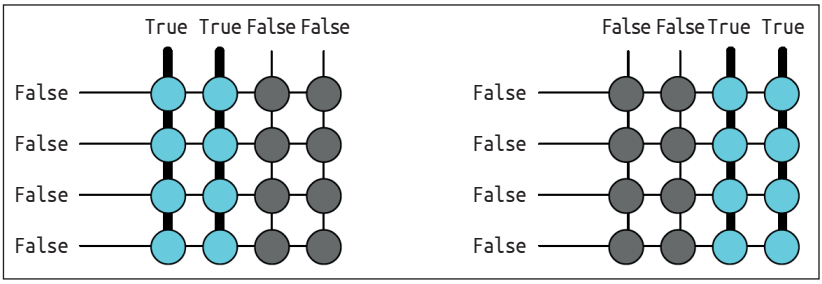


Рис. 11.16. *Левая и правая стороны матрицы светятся по очереди*

Ответы на вопросы

1. Контакт 8 (катод) находится в состоянии `False` (0 В), а контакт 10 (анод) – в состоянии `True` (+3,3 В). Только в этом случае ток может протекать через светодиод.
2. Если все разъёмы имеют логическое состояние `False`, то разницы в напряжении нет ни на одном выводе светодиодов. В этом случае ток течь не будет. Все светодиоды выключены.
3. Четыре светодиода загораются, потому что оба столбца и строчки (линии) имеют четыре точки пересечения.
4. Если управлять индикатором целыми столбцами, загорятся одновременно максимум 50 столбцов. Тогда потребуется 50 резисторов, по одному на каждую строку (линию).

Решение заданий

Решение 1. Затемнение (диммирование) светодиода

Пишем программу

```
# darken.py
from RPi import GPIO
from time import sleep
```

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.OUT)           #1

GPIO.setup(5, GPIO.IN)             #2
GPIO.output(10, True)              #3
while True:                        #4
    if GPIO.input(5) == False:     #5
       GPIO.output(10, False)      #6
       sleep(0.01)                 #7
       GPIO.output(10, True)       #8
       sleep(0.001)                #9
```

Как это работает?

#1 Контакт 10 GPIO назначается выходным каналом.

#2 Контакт 5 – это входной канал.

#3 Светодиод загорается.

#4 Запускается бесконечный цикл. Следующий блок беспрестанно повторяется.

#5 Во время замыкания контактов переключателя контакт 5 соединяется с корпусом. Контакт 5 переходит в состояние `False`, и яркость свечения светодиода снижается. То есть светодиод постоянно включается и выключается. Это происходит настолько быстро, что это мигание заметить невозможно, и нам кажется, что яркость свечения светодиода снижена.

#6 Светодиод выключается.

#7 Ожидание длительностью в 0,01 с. Время ожидания в десять раз дольше, чем время свечения.

#8 Светодиод вновь включается.

#9 Ждём одну миллисекунду.

Решение 2. Правой, левой

Пишем программу

```
# right_left.py
from time import sleep
from RPi import GPIO

GPIO.setmode(GPIO.BOARD)
#Строки
GPIO.setup(8, GPIO.OUT)        #1
GPIO.setup(10, GPIO.OUT)
GPIO.setup(11, GPIO.OUT)
GPIO.setup(12, GPIO.OUT)
```

```
#Столбцы
GPIO.setup(19, GPIO.OUT)
GPIO.setup(16, GPIO.OUT)
GPIO.setup(15, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)

GPIO.output(8, False)          #2
GPIO.output(10, False)
GPIO.output(11, False)
GPIO.output(12, False)
GPIO.output(19, False)
GPIO.output(16, False)
GPIO.output(15, False)
GPIO.output(13, False)

while True:
    GPIO.output(19, True)      #3
    GPIO.output(16, True)
    sleep(0.2)
    GPIO.output(19, False)     #4
    GPIO.output(16, False)
    GPIO.output(15, False)     #5
    GPIO.output(13, False)
    sleep(0.2)
    GPIO.output(15, False)     #6
    GPIO.output(13, False)
```

Как это работает?

#1 Все контакты, подключённые к матрице, назначаются как выходные каналы.

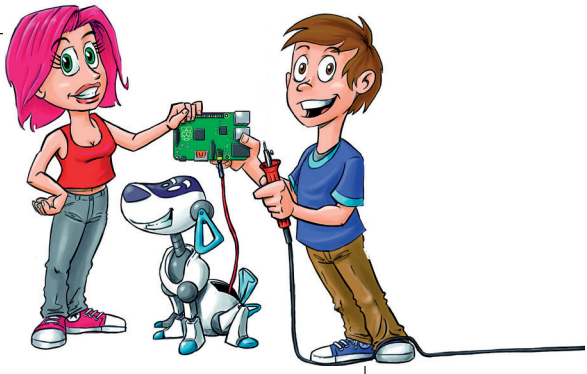
#2 Все контакты устанавливаются в логическое состояние `False`. При этом светодиоды выключены.

#3 Загораются оба *левых* столбца светодиодов.

#4 … и спустя 0,2 с они гаснут.

#5 Загораются оба *правых* столбца светодиода…

#6 … и гаснут спустя 0,2 с.



12
Сбор данных и их обработка

Картинка на светодиодном индикаторе – это рисунок, состоящий из множества световых точек. Сбор общих данных называют коллекцией. Примерами таких коллекций являются списки, кортеж (в базе данных), цепь символов и словари. С помощью всего этого ты можешь моделировать карточные игры, программировать тренажёр для изучения лексики и управлять светодиодным индикатором.

Коллекции

Игрушечная машинка – это модель настоящего автомобиля. Пластилиновый слон – это модель настоящего слона. Модели представляют собой вещи из реальности, но они *проще*, чем в реальности.

Коллекции – это модели предметов или людей, собранные вместе:

- люди в очереди перед кассой в супермаркете;
- планеты Солнечной системы;
- участники теннисного клуба;
- пиксели на мониторе компьютера;
- сладости в пакете.

Для некоторых вещей подобной коллекции существует очерёдность. Вспомни очередь перед кассой! Всегда есть кто-то, кто находится в самом начале. За ним стоит следующий покупатель и т. д. И горе тому, кто лезет вперёд других! Такие коллекции, в которых важна очерёдность, называют *секвенциями*. Термин «секвенция» происходит из латинского языка и означает *последовательность*.

Для сладостей в пакете особой упорядоченности нет. Если кто-то, не глядя, полезет в пакет, то не узнает, что он схватил, пока не достанет это.

Секвенции

Секвенции – это последовательность элементов, расположенных в особом порядке. Python обладает множеством видов секвенций, которые различаются по своим характеристикам.

Списки являются изменяемыми секвенциями любых элементов. Изменяемые – это значит, что элементы можно изменить, дополнить или удалить. С таким списком можно, например, очень хорошо моделировать очередь у кассы, потому что люди то и дело прибывают. Тот, кто оплатил покупки, покидает очередь. И хотя состав очереди постоянно меняется, всё же это по-прежнему одна и та же очередь. Списки записываются с помощью угловых скобок [и]. В них перечисляются элементы, которые разделяются запятой:

```
turn = ['Майке', 'Юлиан', 'Феликс']
```

Список называется `turn` (очередь) и содержит три элемента.

Кортеж является неизменяемой секвенцией. То есть кортеж остаётся таким, какой он есть. Нельзя изменить в нём ни одного элемента. Типичным примером кортежа является дата, ну скажем, 24 декабря 2015. Дата состоит из указания дня, месяца и года. Именно в таком порядке. Если что-то здесь изменить, это уже будет другая дата. В отличие от очереди, при написании кортежа используются круглые скобки:

```
date = (24, 'декабрь', 2015)
```

Strings (строчки) – это неизменяемые секвенции, состоящие из букв (и из других символов), которые заключаются в верхние одиночные запятые или кавычки, например:

'Земля' или "Земля". Слово *string* в переводе с английского означает дословно «нить» или «цепь». В русском этот термин называют *строкой символов*.

Множество

Термин «множество» тебе, вероятно, знаком из уроков математики. *Множество* – это набор любых объектов, собранных без определённого порядка, но обладающих общим определённым признаком. Кроме того, каждый элемент такой структуры является единичным. Как и в математике, мы будем записывать множество с помощью фигурных скобок { и }.

Невозможно получить непосредственный доступ к определённому элементу множества. Но ты можешь проверить, содержится ли в нём какой-либо объект:

```
>>> vokale = {'a', 'e', 'i', 'o', 'u'}
>>> 'e' in vokale
True
>>> 't' in vokale
False
```

Обработка секвенций

Секвенции имеют множество общих характеристик и общих функций. В этом разделе ты познакомишься с самыми важными из них.

Всё имеет свой номер. Изучаем индексы

Элементы одной секвенции пронумерованы, так как речь здесь идёт о последовательности. Каждому элементу присвоен номер, который называется *индексом*. Нумерация начинается с нуля, а не с единицы. Такой порядок нумерации немного отличается от правил в обычной жизни.

Возьмём планеты нашей Солнечной системы. По отношению к Солнцу они все находятся в определённом порядке. В центре расположено Солнце, далее идёт Меркурий. За Меркурием следует Венера. Далее – Земля и т. д.

Список названий планет можно создать с помощью такого распределения.

```
планеты = ['Меркурий', 'Венера', 'Земля', 'Марс', 'Юпитер',
      'Сатурн', 'Уран', 'Нептун']
```

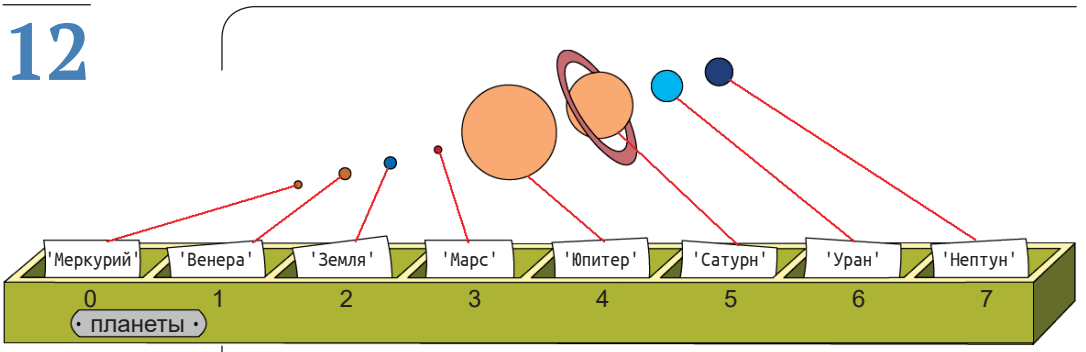


Рис. 12.1. *Планеты нашей Солнечной системы можно представить в виде этой секвенции*

Этот список ты можешь себе представить в качестве контейнера с предметами как на рис. 12.1. Первая планета (Меркурий) имеет индекс `0`. У Венеры индекс `1` и т. д.

Доступ к элементам

Через какой индекс ты сможешь найти каждый элемент секвенции? После названия секвенции ты пишешь индекс в угловых скобках. Если хочется узнать первый элемент списка `планеты`, нужно писать `планеты[0]`.

Запусти оболочку IDLE 3 и попробуй несколько примеров в Python Shell.

```
>>> s = [12, 3, 7]
```

Здесь у нас список из трёх цифр – `12`, `3` и `7`. Список имеет имя `s`. Допустим, ты хочешь узнать, какой элемент списка является первым. Тогда введи в командную строку 0 и после ввода нажми клавишу ↵.

```
>>> s[0]
12
```

В следующей строке появится результат `12`.

Это работает со всеми секвенциями, а также со строчками.

```
>>> слово = 'День'
>>> слово[0]
 'Д'
>>> слово[1]
 'е'
>>> слово[2]
 'н'
```

```
>>> слово[3]
 'ь'
```

С индексом -1 ты можешь получить доступ к *последнему* элементу секвенции. Это удобно, когда не знаешь, насколько длинной является последовательность.

```
>>> слово = 'День'
>>> слово[-1]
 'ь'
```

Длина секвенции

Каждая секвенция имеет свою длину. Это количество содержащихся в ней элементов. С помощью функции len() ты можешь вычислить длину секвенции.

Слово состоит из четырёх букв.

```
>>> слово = 'День'
>>> len(слово)
4
```

Следующий список содержит два элемента. В квадратных скобках ты увидишь, что это за список.

```
>>> выходные = ['Суббота', 'Воскресенье']
>>> len(выходные)
2
```

Существуют также пустые секвенции, которые не содержат никаких элементов. Пустая секвенция имеет нулевую длину. Вот здесь мы видим пустой список:

```
>>> пусто = []
>>> len(пусто)
0
```

Пустая строка не содержит никаких знаков. Она состоит лишь из двух верхних запятых или двух кавычек подряд.

```
>>> len('')
0
```

Строка, состоящая из пробела, не считается пустой, поскольку пробел – это тоже знак:

```
>>> len(' ')
3
```

Объединяем секвенции

С помощью знака «+» вы можете соединить несколько последовательностей в более длинную последовательность. Из двух коротких слов составится длинное слово:

```
>>> 'Raspberry' + 'Pi'
'RaspberryPi'
```

То же самое произойдёт и со списками. Представь себе очередь. Макс и Лукас стоят у кассы. И тут подходит Тина. Сейчас список `очередь` пополнился одним новым элементом.

```
>>> очередь = ['Макс', 'Лукас']
>>> очередь = очередь + ['Тина']
```

Если ты указал название списка, то в следующей строке он выдаст результат:

```
>>> очередь
['Макс', 'Лукас', 'Тина']
```

Но ты можешь связать друг с другом только секвенции *одного типа*. То есть списки можно соединять только со списками, строки только со строками и кортеж только с кортежами. Смешивать между собой различные типы нельзя.

Найти элемент

С помощью оператора `in` ты можешь проверить, содержится ли элемент в секвенции:

```
>>> 15 in [1, 5, 10, 15, 20]
True
>>> 'I' in 'Team'
False
```

При введении строчек существует одна особенность. Здесь оператор `in` также может проверить, является ли строчка частью длинной строки:

```
>>> 'aus' in 'Maus'
True
```

Техники, с которыми ты сейчас познакомился, мы используем в проекте.

Проект 30. Планеты

Это своего рода игра на знания. Игроку нужно перечислить вещи одной категории. В нашем примере будут участвовать планеты Солнечной системы. Но это могут быть также знаки Зодиака, столицы Европы, тяжёлые металлы или что-то другое. Придумай что-нибудь своё!

Фрагмент запуска программы (пример)

```
Назови планету!
Планета: Земля
Верно!
Планета: Марс
Верно!
Планета: Земля
Эта планета уже была!
Планета: Плуто
Это не планета!
...
Ты перечислил все планеты за 135 секунд.
```

Пишем программу

```
# planet.py
import time
планеты = ['Меркурий', 'Венера', 'Земля', 'Марс',
      'Юпитер', 'Сатурн', 'Уран', 'Нептун']        #1
уже_названы = []                                   #2
start = time.time()                                #3
print('Назови планету!: ')

while len(уже_названы) < len(планеты):             #4
  ответ = input('Планета: ')                       #5
  if ответ in планеты:
    if ответ not in уже_названы:
      print('Верно!')
      уже_названы = уже_названы + [ответ]          #6
    else:
      print('Эта планета уже была!')
  else:
    print('Это не планета!')                       #конец while

print('Ты перечислил все планеты за',             #7
      round(time.time() - start),
      'секунд.')
```

Как это работает?

#1 Список с восемью строчками.

#2 Пустой список, не содержащий ни единого элемента. В этом списке собраны названия планет, которые игрок уже назвал. Один список – это одна секвенция, которую можно изменить. Хотя речь здесь не идёт об очерёдности, мы всё же берём список, потому что его можно легко обработать.

#3 Здесь мы начинаем отсчёт времени, поскольку в конце должен отобразиться тот временной промежуток, который потребовался для перечисления всех планет.

#4 Начало цикла, смысл которого в следующем: пока список `уже названы` содержит меньшее число элементов, чем список планет, выполни следующее…

#5 Всё, что игрок вводит с клавиатуры, сохраняется в переменной `Ответ`.

#6 Список `уже названы` добавит планету, которую игрок только что ввёл (содержимое переменной `ответ`). При каждом цикле содержание переменной `ответ` изменяется, потому что игрок каждый раз вводит новое слово. Список `[ответ]` – это список с одним-единственным элементом `ответ`. И этот список, содержащий один элемент, добавляется знаком `+` к списку `уже_названы`. При этом список `уже_названы` становится немного длиннее.

#7 Результат состоит из трёх частей. В центре результата выводится время, которое игроку требуется для решения задачи. Вторая часть рассчитывается так. Сначала стартовое время вычитается из текущего времени: `time.time()` – `start`. Число, которое мы здесь получаем, – это количество секунд, прошедших с момента старта. Это число округляется с помощью функции `round()` до целого числа.

Итерация – инструкция for

При итерации (лат. *iter* – «путешествие», «поход») для каждого элемента выполняется одно и то же действие. Итерации вы знаете из повседневной жизни.

- Съешь всё печенье из банки.
- Поприветствуй каждого гостя на вечеринке.
- Почисти всю свою обувь.

Итерацию ты будешь программировать с помощью команды `for`. Данная команда имеет следующую структуру:

```
for item in kollektion:
    блок инструкций
```

Пример: ты хочешь поприветствовать всех гостей своей вечеринки.

```
for гость in ['Том', 'Лукас', 'Мари']:
    print('Привет ' + гость + '!')
```

Обрати внимание на отступ второй строки. Отступ должен быть обязательно!

Если ты вводишь это в Python Shell, то в строчках снизу появится следующий результат:

```
Привет, Том!
Привет, Лукас!
Привет, Мари!
```

Как это работает? Взгляни, как построена инструкция `for`.

После ключевой команды `for` следует любое имя, которое назовёт управляющая переменная. В нашем случае это переменная `гость`. Но можно взять другое слово или одну букву. Управляющая переменная принимает поочерёдно те значения, которые стоят в коллекции.

Затем следует ключевое слово `in` и коллекция, которая должна поочерёдно выводиться на экран. В нашем случае это список `['Том', 'Лукас', 'Мари']`.

В конце первой строки после списка ставится двоеточие.

После первой строки с командой `for` и списком следует блок инструкций, который должен проводиться для каждого элемента. Наш блок состоит из одной-единственной инструкции:

```
    print('Привет ' + гость + '!')
```

Сначала составляется текст из трёх частей.

- В начале находится слово `'Привет '` (обрати внимание на пробел в конце слова).
- Затем идёт содержимое управляющей переменной `гость`, т. е. *тот* элемент списка, который сейчас в очереди. При первом прохождении это будут `'Том'`, потом `'Лукас'` и затем `'Мари'`.
- По окончании ещё вводятся кавычки `'!'`.

Этот комбинированный текст отображается на экране.

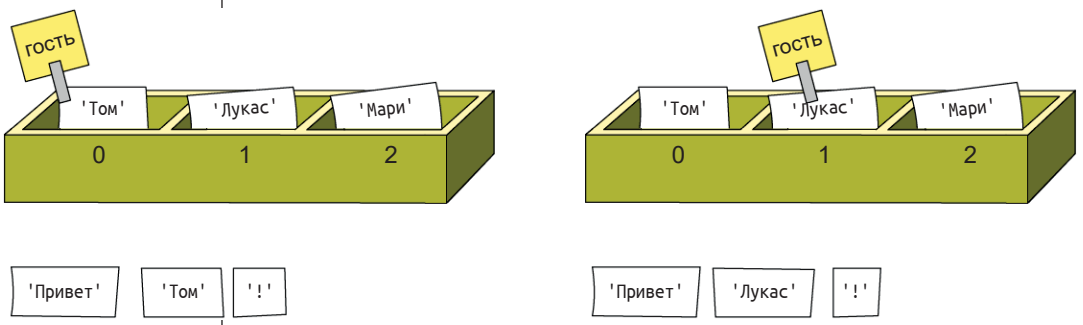


Рис. 12.2. *Управляющую переменную* `гость` *можно представить в виде ярлыка, который прикреплён к элементу списка*

Создать итерацию ты можешь с любой коллекцией, например со строчкой. Мы собираемся записать буквы слова отдельно друг от друга. Выглядит это так:

```
for буква in 'SOS':
    print(буква)
```

Результат, который появится на экране:

```
S
O
S
```

Как это работает?

Управляющая переменная называется `буква`. Коллекция – это строчка `'SOS'`. Строчка – это секвенция из букв. Управляющая переменная поочерёдно принимает значение `'S'`, `'O'` и `'S'`. В команде `print()`, которую снова нужно вложить (записать с отступом), текущее содержимое переменной `буква` всегда при прохождении выводится в новой строке. (То есть после введения команды `print()` всегда идёт новая строка.)

Изменяем списки

Списки являются изменяемыми. Поэтому для них существуют операции, которые влияют на переменные, например удаление, добавление или изменение элементов. Подумай об очереди. Кто-то пропустит человека вперёд, и порядок изменится. Новые люди добавятся в очередь, или кто-то покинет её. Очередь меняется постоянно. Тем не менее это по-прежнему одна и та же очередь.

Попробуй ввести пример в Python Shell.

Пример 1. К списку имён необходимо добавить ещё одно имя.

```
>>> персоны = ['Бенни', 'Басти']
```

Список `персоны` содержит два имени. Добавим туда ещё одно:

```
>>> персоны.append('Флориан')
>>>
```

На экране это может быть не указано, но состояние списка изменилось. Чтобы перепроверить это, просто напиши имя списка и нажми ↵:

```
>>> персоны
['Бенни', 'Басти', 'Флориан']
```

Ага. Получилось! Список стал длиннее.

Операция `append()` называется *методом*. Метод – это функция, которая относится к одному объекту, например к списку. Вызвать метод можно по такому образцу:

```
объект.метод(аргументы)
```

Сначала вводится имя объекта (здесь это список `персоны`), затем идёт точка, а потом имя метода и в скобках аргументы. При использовании метода `append()` аргументом является новый элемент, который должен быть добавлен к списку. В нашем примере это `('Флориан')`.

Пример 2. Элементы должны быть рассортированы в порядке возрастания.

```
>>> числа = [17, 3, 9]
```

У нас есть список `числа`, который содержит три цифры. Теперь мы их сортируем:

```
>>> числа.sort()
```

Метод `sort()` может быть вызван без аргументов. Убедись, что список действительно отсортирован:

```
>>> числа
[3. 9. 17]
```

В объектно-ориентированном программировании представляется, что объекты являются «живыми». Они принимают сообщения и реагируют на них. В нашем примере объект числа получает задание отсортировать свои элементы, а потом выполняет это задание самостоятельно. Как это работает, показано на рис. 12.3. Это похоже на хорошо обученную собаку, способную реагировать на определённые команды.

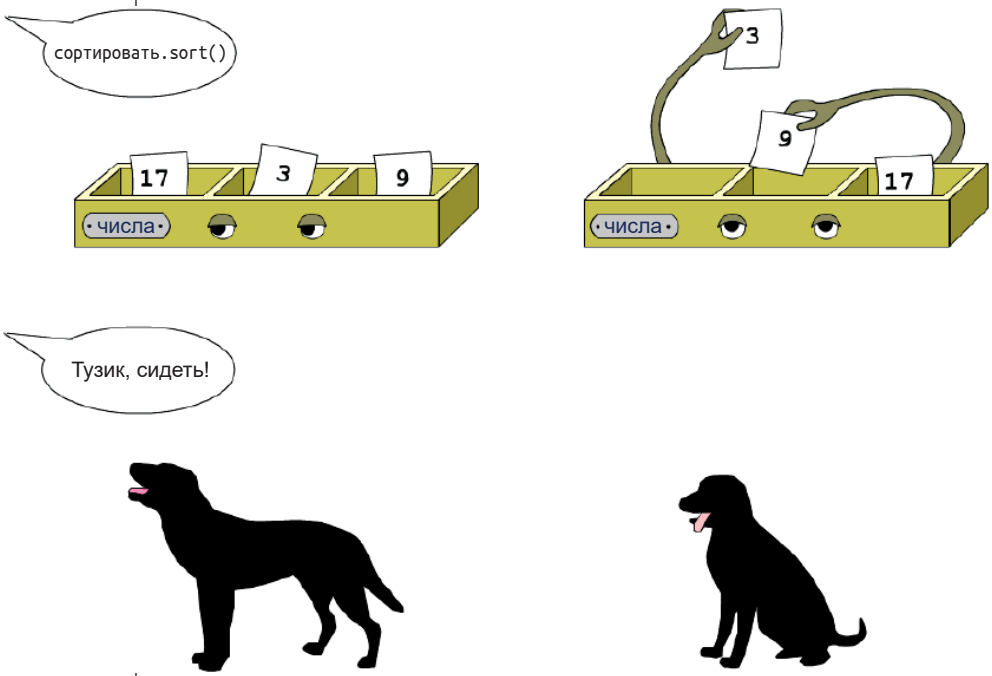


Рис. 12.3. *Идея объектно-ориентированного программирования: объекты реагируют на сообщения*

Таблица 12.1 описывает дальнейшие операции для списков. Попробуй некоторые из них!

Таблица 12.1. Операции для списков

Операция	Результат

`s[i] = x`	Элемент с индексом `i` заменяется на `x`.
	`s = [1, 10, 12]`
	`s[0] = 2`
	Результат: `[2, 10, 12]`

Таблица 12.1 (окончание)

Операция	Результат

`s.append(x)`	К списку s добавлен новый элемент *x*.
	`s = ['Том', 'Тина']`
	`s. append('Бен')`
	Результат: `['Том', 'Тина', 'Бен']`

`del s[i]`	Элемент с индексом *i* удалён.
	`s = ['Май', 'Июнь']`
	`del s[0]`
	Результат: `['Июнь']`

`s.index(x)`	Возвращается наименьший индекс *i* с `s[i] == x`.
	`s = ['Том', 'Тина', 'Бен']`
	`s['Бен']`
	Результат: 2

`s.insert(i.x)`	В случае если `i>=0`, объект *x* ставится перед элементом с индексом *i*.
	`s = ['Май', 'Июль']`
	`s. insert(1, 'Июнь']`
	Результат: `[['Май', 'Июнь', 'Июль']`

`s.remove(x)`	Первый элемент со значением *x* удаляется из списка.
	`s = ['Железо', 'Медь', 'Сера']`
	`s. remove('Сера')`
	Результат: `['Железо', 'Медь']`

`s.reverse()`	Последовательность элементов меняется на обратную.
	`s = ['A', 'B', 'C']`
	`s. reverse()`
	Результат: `['C', 'B', 'A']`

`s.sort()`	Элементы списка сортируются в порядке возрастания.
	`s = [17, 3, 9]`
	`s. sort()`
	Результат: `[3, 9, 16]`

Случайные функции секвенций

Очень удобны случайные функции секвенций, которые определены в модуле `random`. Оператор `choice(s)` выдаёт случайный элемент секвенции `s`. Такая функция работает со всеми видами секвенций.

```
>>> from random import *
>>> choice('aeiou')
'i'
```

Оператор `shuffle()` (англ. *to shuffle* – «смешивать») смешивает элементы одного списка и выдаёт их в случайном порядке:

```
>>> s = [1, 2, 3, 4]
>>> shuffle(s)
>>> s
[4, 2, 1, 3]
```

Проект 31. Вытяни карту

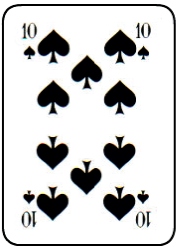
Игрок может одну за другой вытащить карты из колоды (32 карты).

Каждую карту мы смоделируем через кортеж. Ты помнишь, как это делается? Кортеж – это секвенция с круглыми скобками. Менять его нельзя. Это подходит к картам. Ведь ты не можешь просто взять и изменить значение карты. Каждый карточный кортеж имеет форму `(масть, достоинство)`.

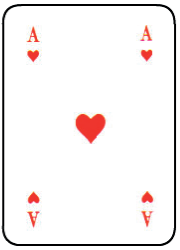
Масти у нас такие: трефы, пики, червы, бубны.

Достоинства карт: 7, 8, 9, 10, валет, дама, король, туз.

Например, карта 10 пик представлена через кортеж `('Пики', 10)`.



`('Пики', 10)`



`('Черви', 'Туз')`

Рис. 12.4. *Игральные карты можно смоделировать через кортеж*

Почему кортеж, а не список? Кортеж является *неизменяемым*. И у игровой карты тоже ничего нельзя менять, потому что иначе это была бы уже совсем другая карта. Поэтому кортеж здесь – лучшая модель.

Ход программы (образец)

```
Вытащить карту? (д/н) д
Ты вытащил Пиковую Даму.
Ещё одну карту? (д/н) д
```

```
Ты вытащил Крестовую Девятку.
Ещё карту? (д/н) н
До свидания.
```

Пишем программу

```
# playing_card.py
from random import *
масти = ['Бубны', 'Черви', 'Пики', 'Трефы']
достоинства = [7, 8, 9, 10,
               'Валет', 'Дама', 'Король', 'Туз']
игра = []                                          #1
for м in масти:                                    #2
    for д in достоинства:
        игра.append((м, д))                        #3

ответ = input('Вытащить карту? (д/н): ')
while (ответ == 'д') and (len(игра) > 0):          #4
    карта = choice(игра)                           #5
    игра.remove(карта)                             #6
    print('Ты вытянул', карту[0], карту [1],'.')
    ответ = input('Еще одну карту? (д/н): ')
if игра == []:                                     #7
    print('Карт больше нет.')
print('До свидания.')
```

Как это работает?

#1 Сначала мы создаём пустой список `игра`, в котором не содержится ни одного элемента.

#2 В следующих строчках мы строим список `игра`, который моделирует карточную игру. Он содержит 32 элемента и имеет строение `[('Бубны', 7), ('Бубны', 8)` и `т. д.]`.

Управляющей переменной для списка `масти` является `м`. Управляющей переменной для списка `достоинства` является `д`. Рисунок 12.5 иллюстрирует этот процесс. Сначала `м` получает достоинство `'Бубны'`. Затем `д` проходит через полный список `достоинства`. После этого `м` получает достоинство `'Черви'`. И снова `м` проходит через полный список `достоинства`. И так далее.

#3 К списку `игра` добавляется новый элемент формы (`м` – (масть), `д` – (достоинство)).

#4 До тех пор пока игрок запрашивает остальные карты и карты имеются в наличии, выполняется следующий блок инструкций: #5, #6.

#5 Из списка `игра` выбирается случайный элемент.

#6 Выбранная карта выводится из списка.

#7 Если список пуст, то приходит сообщение, что карт больше нет. Можно было также написать:

```
if not список: …
```

так как пустой список имеет истинное значение `False`.

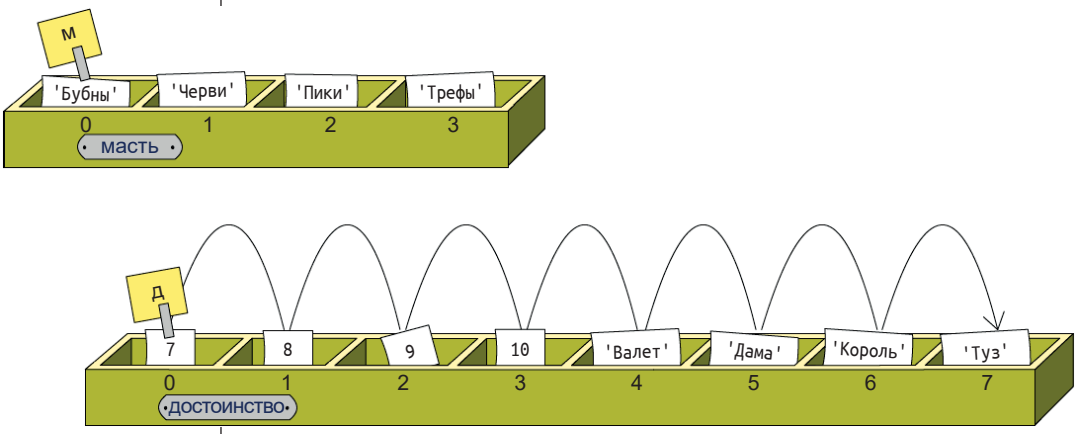


Рис. 12.5. *Два списка с управляющими переменными*

Последовательность чисел – функция range()

С помощью функции `range()` ты можешь создавать последовательность чисел. Например, вызвав

```
range(3)
```

ты создашь последовательность 0, 1, 2.

Попробуй следующий пример:

```
>>> for i in range(3):
        print(i)

0
1
2
```

Здесь `i` является управляющей переменной. Число в скобках, которое стоит после `range`, называется аргументом. В вышеприведённом примере аргументом у нас была цифра 3. Аргумент указывает, сколько элементов содержится в числовой последовательности. Серия 0, 1, 2 содержит три элемента. Поскольку первый из них равен нулю, то послед-

ний будет на единицу меньше, чем аргумент. Аргумент функции `range()` – это тип границы, *до* которой должна заканчиваться последовательность чисел.

Второй пример: вызов функции

```
range(100)
```

создаёт все целые числа от 0 до 99. То есть это 100 чисел.

Последовательность чисел может быть хорошо использована для повторений. Аргумент в скобках отображает, как часто должен выполняться блок инструкций (количество повторов):

```
>>> for I range(3):
        print('Help!')

Help!
Help!
Help!
```

Функция `range()` возвращает объект `range`. Для этого компьютеру потребуется меньше свободного пространства, чем, например, для списка. Если ты хочешь сделать числа визуальными, то должен получить список из объекта `range` с помощью функции `list()`:

```
>>> числа = range(10)
```

Здесь мы создали range-объект с именем `числа`. Объект `range` выводится при вводе его имени и нажатии ↵.

```
>>> числа
range(0, 10)
```

Этот `range`-объект мы можешь преобразовать в списке:

```
>>> list(числа)
[0. 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

В списке ты можешь узнать все числа от 0 до 9.

Dictionarys (словари)

Dictionary (англ. *словарь*) – это распределение ключей по значениям. Верно подобранный ключ, который подходит

к двери, позволяет получить доступ в помещение. Точно так же ключ в словаре позволяет получить прямой доступ к связанным данным.

Словари знакомы тебе из обычной жизни.

- В адресной книге ты находишь номера телефонов людей (значений) через имена (ключи).
- В немецко-русском словаре ты к немецкому слову (ключу) находишь список русских слов с похожим значением.

В Python словарь представлен последовательностью пар формы `ключ: значение` в фигурных скобках. Пример:

```
>>> english = {'Sun': 'Солнце', 'Moon': 'Луна'}
```

Ключами здесь являются английские слова, а подобранные значения относятся к русским словам. Используя английские слова как ключи, ты можешь подобрать подходящие к ним русские слова. Для этого напиши название словаря, например `english`, и поставь после него в квадратных скобках ключевое слово `[]`, заключённое в верхние одинарные кавычки (например, `'Sun'`).

Пример:

```
>>> english['Sun']
'Солнце'
```

Словарь `english` ты можешь расширить через инструкцию формы

```
e[key] = значение
```

Пример:

```
>>> english['star' ] = 'звезда'
>>> engkish
{'Moon' : 'Луна', 'star': 'звезда', 'Sun': 'Солнце'}
```

Здесь добавлена пара `'star': 'звезда'`.

Ключами словаря должны быть единичные и неизменяемые объекты.

У словарей тоже есть методы. С помощью метода `keys()` ты получишь коллекцию всех ключей:

```
>>> english = {'Moon' : 'Луна', 'star': 'звезда', 'Sun': 'Солнце'}
>>> list(english.keys())
['Moon', 'star', 'Sun']
```

Проект 32. Учим лексику

Мы разработаем программу для изучения английских слов. Вместо слов ты можешь, конечно же, запрашивать термины или другие вещи.

Программа называет английское слово и спрашивает его русский перевод. Но таким образом, чтобы к одному английскому слову было подобрано несколько русских слов с похожими значениями. Поэтому в программе мы будем использовать словарь, который сохраняет список русских слов в виде значения для английского слова (ключа).

Ход программы (образец)

```
Как перевести Key на русский?
- - > Ключ
Верно!
Как перевести Tree на русский?
 - - > Куст
Неправильно!
Tree означает:
Дерево

Как перевести Flower на русский?
- - > Цветок
Верно!
Как перевести Tree на русский?
- - > Дерево
Верно!
Выучены все слова!
Ты допустил одну ошибку.
```

Пишем программу

```
from random import *
english={'Key':['Ключ', 'Клавиша'],                #1
         'Tree':['Дерево'],
         'Flower': ['Цветок']}
ошибка = 0
слова = list(english.keys())                       #2
while слова:                                       #3
    слово = choice(слова)                          #4
    print('Как перевести', слово, 'на русский?')   #5
```

```
    ответ=input('- - > ')
    if ответ in english[слово]:                    #6
        print ('Верно!')
        слова.remove(слово)                        #7
    else:
        print('Неправильно!')                      #8
        print(слово.capitalize(), 'означает: ')    #9
        for w in english[слово]:                   #10
            print(w)
        print()                                    #11
        ошибка += 1                                #12
print('Все слова выучены!')                        #13
print('Ты допустил', ошибок, 'ошибку.')            #14
```

Как это работает?

#1 Словарь, в котором каждое английское слово (ключ) относится к списку с русским переводом (значением).

#2 Ключи, найденные в словаре английского языка (`'Key'`, `'Tree'`, `'Flower'`), группируются в список. Другими словами, формируется список слов, соответствующих значениям `'key'`, `'Tree'`, `'Flower'`.

#3 Пока список слов не опустеет, цикл, состоящий из строк #4–#13, повторяется.

#4 Переменная `слово` получает значение выбранного случайным образом слова из списка `слова`.

#5 Отображается текст, который состоит из трёх частей: сначала идёт `Как перевести`, потом содержание переменной `слово` (случайно выбранное английское слово), после чего идёт строка `на русский?`.

#6 Если ответ пользователя соответствует переводу английского слова из списка, выводится ответ `Верно!`. Обрати внимание, если ты вводишь правильное слово, но регистр первой буквы не соответствует, компьютер скажет, что это слово неправильное.

#7 Правильно названное слово удаляется из списка слов. Его уже запомнили.

#8 Если ответ пользователя неверный, появляется сообщение: `Неправильно!`.

#9 Если ответ неправильный, в этой строке сначала с большой буквы (заглавную букву обеспечивает метод `capitalize()`) выводится английское слово, которое было предложено к переводу, далее идёт слово `означает`, после чего выводится правильный перевод этого слова.

#10 Команда `for w in english[слово]`: присваивает управляющей переменной `w` все сохранённые переводы англий-

ского слова, предложенного к переводу, на которое был дан неправильный ответ. Команда `print(w)` выводит эти слова на экран:

```
Ключ
Клавиша
```

#11 Эта инструкция создаёт пустую строку.

#12 Значение переменной `ошибка` увеличивается на `1`. Можно записать и так:

```
ошибка = ошибка + 1
```

На этом цикл `while` завершается.

#13 Цикл завершён. Эта командная строка без отступа. Компьютер выдаёт текст `Все слова выучены!`.

#14 В этой строке выводится количество (значение переменной `ошибка`).

Проект 33. Световой сигнал

В этом проекте мы вновь обратимся к светодиодной матрице из главы 11.

На автомагистралях иногда можно увидеть матрицу, состоящую из множества огней, образующих стрелу. Эта стрела указывает водителям, с какой стороны они могут объехать препятствия. Мы разработаем мини-версию такого индикатора. С помощью клавиатуры можно указать, какой символ должен появиться: стрелка вправо, стрелка влево или крест (указывающая, что эта полоса заблокирована).

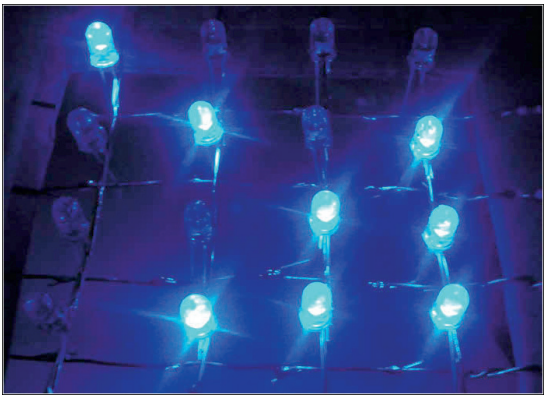


Рис. 12.6. *Светодиодная матрица указывает стрелкой направо и вниз*

Составляем список из списков

Светодиодную матрицу размером 4×4 ты можешь представить в виде списка из четырёх списков:

```
m = [[1, 0, 0, 0],[0, 1, 0, 1],[0, 0, 1, 1],[0, 1, 1, 1]]
```

Эти четыре списка можно также написать друг под другом, тогда их лучше можно распознать:

```
m = [[1, 0, 0, 0],
     [0, 1, 0, 1].
     [0, 0, 1, 1].
     [0, 1, 1, 1]]
```

Первый список `m[0]` – это первая строчка, второй список `m[1]` –вторая строчка, `m[2]` – это третья строчка, и `m[3]` –четвёртая. Единица означает «светодиод светится». Ноль означает «светодиод не светится». Если ты внимательнее присмотришься, то в этом примере сможешь из нулей и единиц разглядеть стрелку.

Как получить доступ к отдельным светодиодам? Предположим, что в следующей светодиодной матрице ты хочешь светодиод, обозначенный жирным нулём `0`, перевести в состояние `1` (другими словами, выбранный светодиод должен светиться).

```
m = [[0, 0, 0, 0],
     [0, 0, 0, 0],
     [0, 0, 0, 0],
     [0, 0, 0, 0]]
```

Искомый элемент находится во второй строчке, обозначенной `m[1]`, третьим элементом, имеющим индекс `2`. В общей сложности ноль, выделенный жирным шрифтом, является элементом `m[1][2]`.

Как изобразить любые рисунки?

Здесь у нас возникает одна проблема. Обычной коммутацией светодиодов матрицы мы не сможем сделать так, чтобы светились два светодиода, расположенных по диагонали, как показано на рис. 12.7 слева. Посмотри на правый рисунок. Когда контакты, к которым подключены две крайние вертикальные контактные линии матрицы, пере-

ведены в состояние True (истина) и на них присутствует положительное напряжение 3,3 В, а контакты, к которым подключены крайние горизонтальные контактные линии, переведены в состояние False и подключены к *массе*, на пересечении этих линий находятся четыре угловых светодиода. И все они будут светиться.

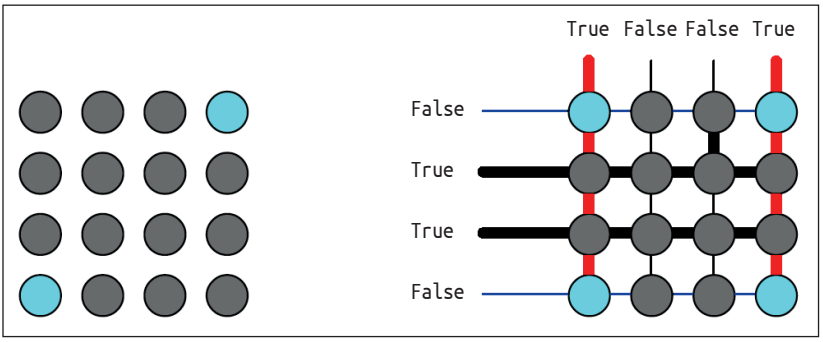


Рис. 12.7. *Это невозможно! Оба светодиода с левой картинки никогда не смогут светиться одновременно*

Как теперь изобразить на светодиодной матрице стрелку и другие рисунки? Здесь нам поможет только инерция человеческого глаза, когда человек некоторое время ещё помнит предыдущую картинку, хотя в действительности картинка изменилась. То есть светодиоды будут с определённой частотой светиться по очереди. Рисунок 12.8 показывает нам это на примере. Одновременно отображается только *одна* строка картинки. То есть контакт, к которому подключена *одна* горизонтальная контактная линия (находится в состоянии False), подключён к корпусу, а у остальных этот уровень высокий (состояние True). Если поочерёдно и очень быстро, не переставая, показывать каждую строчку со светящимся нужным светодиодом, глаз увидит только одну общую картинку, состоящую из этих строк. Это мы сейчас и попробуем.

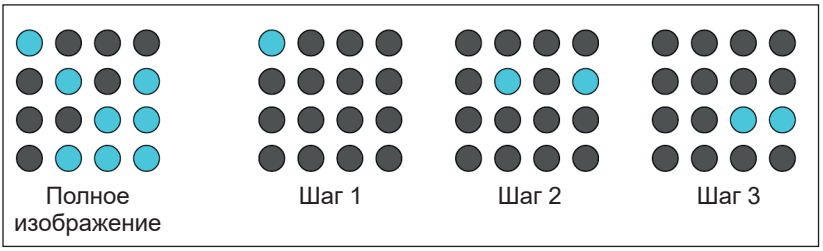


Рис. 12.8. *Изображение отображается по строчкам*

Пишем программу

```
# arrow.py
# инициализация GPIO
import time
from RPi import GPIO
GPIO.setmode(GPIO.BOARD)
s = [19, 16, 15, 13]
z = [8, 10, 11, 12]
for pin in z + s:
    GPIO.setup(pin, GPIO.OUT)

# Konstanten
LEFT =  [[0, 0, 0, 1],
         [1, 0, 1, 0],
         [1, 1, 0, 0],
         [1, 1, 1, 0]]                                  #1

RIGHT = [[1, 0, 0, 0],
         [0, 1, 0, 1]],
         [0, 0, 1, 1],
         [0, 1, 1, 1]]

INTERSECTION = [[1, 0, 0, 1],
                [0, 1, 1, 0],
                [0, 1, 1, 0],
                [1, 0, 0, 1]]

символ = {'L':LEFT, 'R':RIGHT, 'I':INTERSECTION}     #2

for pin in z:                                          #3
    GPIO.output(pin, True)
for pin in s:
    GPIO.output(pin, False)

print( 'Пожалуйста, введи L, R или I')
a = input('символ: ')
if a in 'LRI':                                         #4
    образец = символ[a]                                #5
    while True:
        for i in range(4):                             #6
            GPIO.output(z[i], False)                   #7
            for j in range(4):
                GPIO.output(s[j], образец[i][j])       #8
            time.sleep(0,001)
            GPIO.output(z[i], True)                    #9
```

Как это работает?

#1 Светодиодный рисунок моделируется с помощью списка, состоящего из четырёх списков. Согласно списку

LEFT, на светодиодной матрице появляется светящееся изображение стрелки, показывающей влево вниз. Список RIGHT включает светодиоды так, что изображение стрелки направлено вправо вниз. Список INTERSECTION коммутирует светодиоды так, что светятся угловые светодиоды, а во второй и третьей строках светятся светодиоды, расположенные ближе к центру. Каждый подсписок изображает одну строчку. Вместо значений True и False мы используем 1 и 0, так как это короче. Это работает, потому что 0 принимает значение False, а 1 – значение True. Итак, 1 у нас будет включать светодиод, а 0 – выключать светодиод.

#2 Это словарь. Светодиодный рисунок сопоставляется с буквами.

#3 Все светодиоды выключены. Далее предлагается ввести с клавиатуры нужный символ (L, R или I), и значение этого символа присваивается переменной a.

#4 Проверяется, не вводился ли один из знаков L, R, I.

#5 С помощью этой команды просматривается словарь, символ и ищется шаблон, соответствующий значению переменной a. Далее найденный шаблон присваивается переменной образец.

#6 Здесь выполняются все четыре строчки выбранного шаблона.

#7 Контакт, подключённый к строке i, переходит в состояние False. Другие контакты, к которым подключены остальные строки, остаются в состоянии True. Активирована только строка i. Только в этой строке могут загореться светодиоды.

#8 Переключаются контакты, к которым подключены вертикальные контактные линии матрицы (столбцы). Светодиод в столбце j начинает светиться, если в строке i и в столбце j образца указано значение 1.

#9 После ожидания в миллисекунду строчка деактивируется. На всех выводах светодиодов, подключённых к этой строке, напряжение становится одинаковым, и все светодиоды гаснут. На очереди следующая строка.

Проект 34. Светодиодные буквы

Самым популярным точечным индикатором является модель TA07-11HWA фирмы «Кингбрайт» (Kingbright). Он имеет семь строк, на каждой из которых по пять малень-

ких красных светодиодов. То есть матрица состоит из семи строк и пяти столбцов. Подключи светодиодную матрицу, как описано в главе 11.

Следующая программа покажет изображение буквы А, состоящее из светящихся светодиодов.

Пишем программу

```
import time
from RPi import GPIO
#GPIO инициализация
GPIO.setmode(GPIO.BOARD)
s = [3, 5, 7, 8, 10]
z = [11, 12, 13, 15, 16, 18, 19]
for pin in z + s:
    GPIO.setup(pin, GPIO.OUT)                          #1

# светодиоды выключить
for pin in z:
    GPIO.output(pin, True)
for pin in s:
    GPIO.output(pin, True)                             #2

A = [[0, 0, 1, 0, 0],
     [0, 1, 0, 1, 0],
     [1, 0, 0, 0, 1],
     [1, 1, 1, 1, 1],
     [1, 0, 0, 0, 1],
     [1, 0, 0, 0. 1],
     [1, 0, 0, 0, 1]]                                  #3

while True:
  for строка in range(7):                              #4
    GPIO.output(z[строка], False)                      #5
    for столбец in range(5):                           #6
      GPIO.output(s[столбец], A[строка][столбец])      #7
    time.sleep(0.001)
    GPIO.output(z[строка], True)                       #8
```

Как это работает?

#1 Все подключённые контакты GPIO настроены в качестве выходных каналов.

#2 На контактах, к которым подключены строки и столбцы матрицы, устанавливается положительное напряжение 3,3 В. Так как на всех выводах светодиодов, из которых состоит матрица, одно и то же напряжение, ток через светодиоды не течёт, и светодиоды не светятся.

#3 Список состоит из семи подсписков. Каждый подсписок описывает состояние светодиодов одной строки свето-

диодной матрицы. Если ты внимательно посмотришь на все эти подсписки, то увидишь букву A, состоящую из единиц.

#4 Ранжируются все семь строк.

#5 Строка с номером `строка` активируется. Через светодиоды этой строки может протекать электрический ток, так как все катоды светодиодов этой строки подключены к корпусу.

#6 Ранжируются все столбцы.

#7 Светятся те светодиоды, для которых в матрице A указана единица.

#8 Светодиоды светятся одну миллисекунду, после чего строка вновь деактивируется.

Вопросы

1. Какие коллекции являются изменяемыми?
2. Какое отличие между циклом `while` и итерацией? Используй термины «коллекция» и «условие».
3. Распредели картинки к подходящим спискам.

```
A = [[T], [F, T, F], [F, T, F]]
```

```
B = [F, [B, S]]
```

```
C = [[F, F], [F, F, F, F], [F, F, F, F], [F, F]]
```

```
D = [[0, 0, 0, 1, 1], [0, 0, 0, 0, 1]]
```

Рис. 12.9. *Списки – это модели*

Задание. Гороскоп

Напиши программу, которая создаст случайный гороскоп. При каждом запуске программы должен появляться другой текст. Пример:

```
Твой гороскоп
Звёзды говорят, что в субботу произойдёт что-то хорошее.
```

Совет: используй три списка с текстом и функцию для выбора случайного значения `choice()` из модуля `random`. Пусть рис. 12.10 поможет тебе.

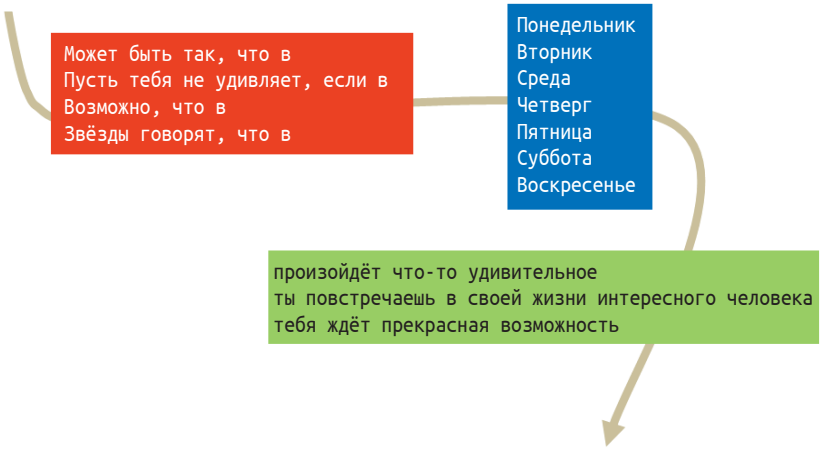


Рис. 12.10. *В каждом прямоугольнике выбери строчку и всё запиши их поочерёдно*

Ответы на вопросы

1. Изменяемые коллекции – это списки, множество и словари.
2. В цикле `while` блок инструкций повторяется до тех пор, пока выполняется условие. В *итерации* выполняется *коллекция* и каждый элемент коллекции обрабатывается каким-либо образом.
3. A. Фасад дома справа внизу. `T` означает дверь, а `F` – окно.
   B. В большом контейнере находится бутылка с моющим средством, а в маленьком стоит щётка с губкой.
   C. Фасад дома слева вверху: наверху два окна, внизу четыре окна, ещё ниже снова четыре окна и в подвале два окна.
   D. Коробка с яйцами в два ряда, в каждом из которых может содержаться 5 яиц. Единица зовётся «Здесь лежит одно яйцо», а ноль – это «Яиц нет».

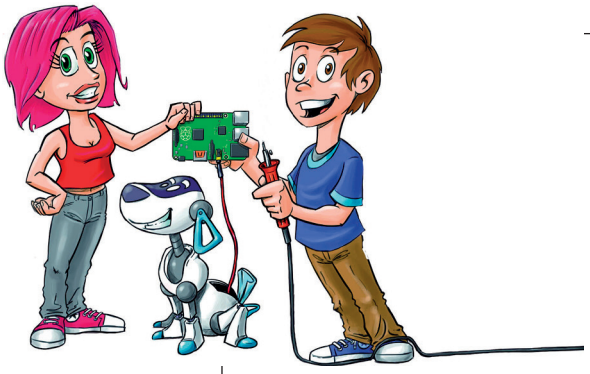
Решение задачи

```
from random import choice

A = ['Может быть так, что в ',
     'Пусть тебя не удивляет, если в ',
     'Возможно, что в ',
     'Звезды говорят, что в ']
B = ['понедельник ', 'вторник', 'среду',
     'четверг '. 'пятницу '. 'субботу ',
     'воскресенье ']
C = ['произойдет что-то удивительное.',
     'ты повстречаешь в своей жизни интересного человека.',
     'тебя ждет прекрасная возможность.']

гороскоп = choice(A) + choice(B) + choice(C)   #1
print('Твой гороскоп')
print(гороскоп)                                #2
```

Как это работает?

#1 Три фрагмента текста, выбранных случайно, объединяются в один гороскоп.

#2 Результатом будет случайный гороскоп.



13 Работа с ЖК-индикатором

Маленький светодиодный дисплей есть у многих устройств: радиоприёмников, калькуляторов или посудомоечных машин. В этой главе ты узнаешь, как можно отображать тексты на двустрочном жидкокристаллическом индикаторе. Мы создадим с тобой цифровой календарь и таймер.

Работа с ЖК-дисплеем

Аббревиатура LCD расшифровывается как *liquid crystal display* – «жидкокристаллический дисплей». Термин «LCD-дисплей» является, по сути, «маслом масляным», но именно такая маркировка прочно укоренилась в разговорной речи. На русском языке LCD значит «жидкокристаллический дисплей». Но что это за жидкие кристаллы? Вообще, жидкие кристаллы – крайне интересная субстанция. Их молекулы действительно ведут себя как молекулы жидкого вещества, находясь в постоянном движении. Но, как и положено кристаллам, их ориентация остается неизменной. Ориентация кристаллов меняется только под воздействием электрического поля.

Если ты внимательнее приглядишься, то обнаружишь на ЖК-дисплее маленькие точки – светлые и тёмные. Каждая

из них и есть жидкий кристалл, за которым находится подсветка. С помощью электрического поля мы изменяем ориентацию этого кристалла, что, в свою очередь, изменяет его оптические свойства. В зависимости от состояния кристалла изменяются свойства проходящего через него света (кристалл можно рассматривать как маленькую линзу, изменяющую поляризацию проходящих через него световых волн). И проходящий через него свет может быть пропущен без изменений (светлая точка), а может быть отфильтрован (тёмная точка).

Мы будем использовать ЖК-дисплей размером 16×2 на базе широко распространённого контроллера HD44780 фирмы «Хитачи» (Hitachi) (рис. 13.1). Подобные детали стоят в магазинах очень недорого. Дисплей имеет две строки, каждая вместимостью 16 знаков. На языке информатики это будет кортеж с двумя списками, имеющий длину 16.

Рис. 13.1. *ЖК-дисплей, модель HD44780 16×2*

Подготовка аппаратной части

Для этого проекта тебе понадобятся:

- ЖК-дисплей (16×2), совместимый с контроллером HD44780;
- 16-контактный разъём;
- резистор номиналом 560 Ом;
- макетная плата;
- провода с разъёмами «мама-папа»;
- паяльник, олово и канифоль (или спиртовой флюс).

Из списка материалов становится ясно, что тебе придётся паять. По времени это займёт не больше пяти минут, и сама работа чрезвычайно простая. Модель контроллера HD44780 имеет 16 портов. Это маленькие отверстия, к которым требуется подпаять контактные штырьки, с помощью которых мы сможем подключить ЖК-индикатор к макетной плате.

Но прежде сделай вот что.

- Возьми 16-контактный разъём (рис. 13.2) и вставь его в макетную плату так, чтобы короткими выводами он был направлен вверх (рис. 13.2 справа).
- Установи на него ЖК-дисплей так, чтобы отверстия в плате жидкокристаллического дисплея совпали с короткими штырьками 16-контактного разъёма, и припаяй контакты разъёма с контактными площадками платы дисплея.

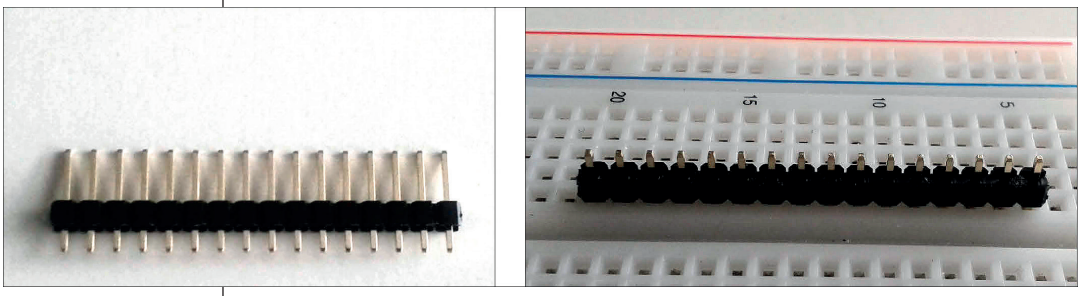


Рис. 13.2. *16-контактный разъём*

Затем конструкцию следует подключить к блоку питания и проверить, работает ли ЖК-дисплей.

- Установи ЖК-дисплей на макетную плату, как показано на рис. 13.3. Слева находится контакт 1, а справа – контакт 16 (рис. 13.3).
- Соедини с помощью перемычек первый контакт платы дисплея с отрицательной линией питания макетной платы (отмеченной синей линией) как на рис. 13.3.
- Соедини контакт 2 платы дисплея с положительной линией питания макетной платы (отмеченной красной линией) (рис. 13.3).
- Соедини таким же образом контакты 3 и 5 с отрицательной линией питания макетной платы (отмеченной синей линией) как на рис. 13.3.
- Соедини контакт 15 платы дисплея через резистор номиналом 560 Ом с положительной линией питания

макетной платы (отмеченной красной линией) как на рис. 13.3.

- Соедини контакт 16 платы дисплея перемычкой с отрицательной линией питания макетной платы (отмеченной синей линией) как на рис. 13.3.
- Используй провод с разъёмами («мама-папа») и подключи макетную плату, чтобы положительная линия питания макетной платы (отмеченная красной линией) была подключена к контакту 2 разъёма GPIO (+5 В) материнской платы компьютера, а отрицательная линия питания макетной платы (отмеченная синей линией) – к контакту 6 (корпус).

Обрати внимание: контакт 3 платы дисплея предназначен для регулирования контрастности изображения. Когда контакт 3 подключён к корпусу, контрастность изображения максимальная. Из соображений безопасности контакт 5 (R/W) мы подключили на корпус. Это сделано из соображений безопасности, чтобы не повредить RPi. Дело в том, что состояние ЖК-дисплея можно запросить с RPi. Но для работы ЖК-дисплея требуется напряжение 5 В, а входы GPIO работают с напряжением 3,3 В. Таким образом, операции чтения могут привести к повреждению RPi. Поэтому мы разрешаем только подавать информацию на дисплей.

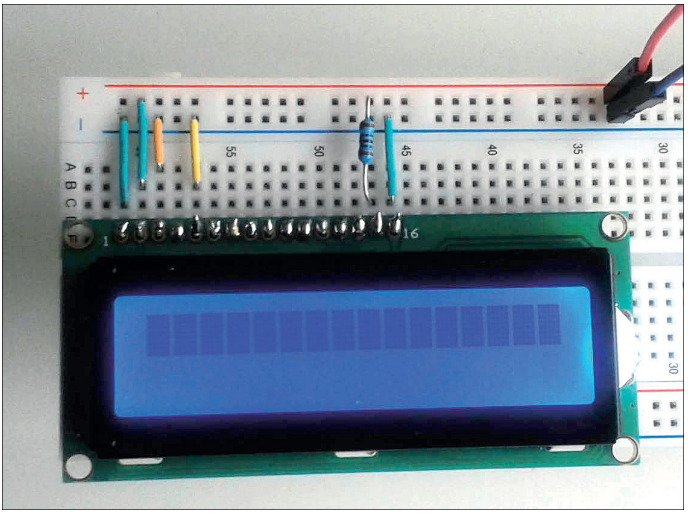


Рис. 13.3. *Подключение платы дисплея к контакту 2 питания (5 В) и контакту 6, подключенному к корпусу*

Если всё сделано правильно и дисплей заработал, в верхней строке дисплея появится 16 блоков. В дальнейшем на

месте этих блоков будут формироваться буквы. Если индикатор слишком тёмный и данные видны нечётко, то можно предположить, что напряжение питания не соответствует тому, на которое рассчитан индикатор. В этом случае возьми мультиметр, выбери режим измерения постоянного напряжения с пределом измерения до 20 В, установи один щуп прибора на контакт 15 платы дисплея, а второй щуп – на контакт 16. Если показания прибора будут меньше 5 В, значит, мощность блока питания недостаточна. Мощность измеряется в ваттах (Вт) и рассчитывается по следующей формуле: $P = I \times U$, где P – это мощность, I – потребляемая сила тока, а U – напряжение источника питания. То есть если, например, на блоке питания написано «Output: 5 В – 1,5 А», значит, блок питания может обеспечить напряжение 5 В с максимальной силой тока 1,5 А. Мощность этого блока питания 7,5 Вт: 5 × 1,5 = 7,5 Вт.

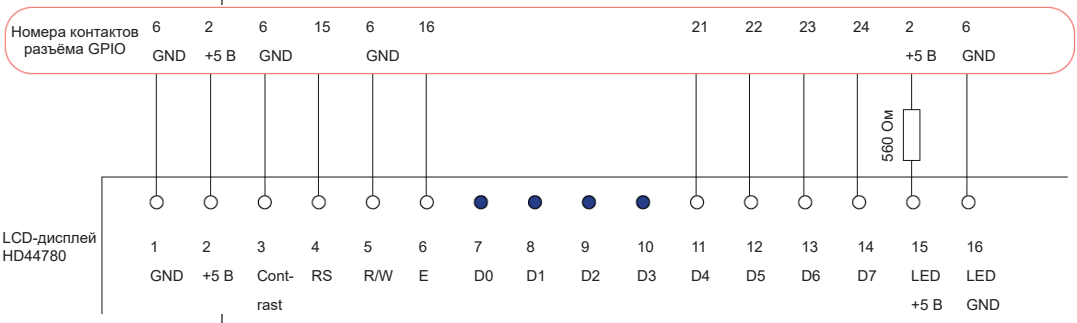


Рис. 13.4. *Подключение индикатора модели HD44780 к GPIO RPi (модуль 4 бита)*

В заключение подключи остальные контакты платы дисплея к контактам разъёма GPIO материнской платы, как показано на рис. 13.4. То есть контакт 4 платы дисплея следует подключить к контакту 15 GPIO, контакт 6 платы дисплея соединяется с контактом 16 GPIO, контакт 11 платы дисплея – с контактом 21 GPIO, контакт 12 платы дисплея – с контактом 22 GPIO, контакт 13 платы дисплея – с контактом 23 GPIO, контакт 14 платы дисплея – с контактом 24 GPIO. Контакты 7, 8, 9 и 10 мы использовать не будем, потому что мы используем модель HD44780 в четырёхбитном режиме. То есть мы будем использовать только четыре канала передачи данных.

Подключения к GPIO подобраны таким образом, чтобы они совпадали с настройками модуля Python, который мы используем. Более подробно об этом в следующем разделе.

Устанавливаем программное обеспечение

Для отображения текста на ЖК-дисплее мы воспользуемся модулем Python RPLCD, который был разработан Данило Баргеном (Danilo Bargen). Этот модуль не входит в стандартную установку, его необходимо сначала загрузить и установить. RPLCD доступен только для Python 2. Для Python 3 он не доступен. Но это нестрашно, потому что на твоём Raspberry Pi установлен и Python 2. Программные тексты, которые мы будем писать, выглядят точно так же, как и для Python 3. Однако есть несколько моментов, которые ты должен учесть. Об этом чуть позже. Вначале тебе необходимо с помощью программы `pip` установить модуль `RPLCD`.

- Введи в командную строку терминала команду

```
$ sudo pip install RPLCD
```

Как отобразить текст на ЖК-индикаторе?

В первом проекте мы выведем на ЖК-дисплей маленький текст, который запишем в двух строках. Обрати внимание: после запуска программы текст на дисплее будет сохранён. Объясняется это тем, что у платы дисплея есть собственная оперативная память, которая не зависит от программ, работающих на этом компьютере.

Программа выполняется с использованием прав суперпользователя. Поэтому оболочку Python следует запустить из командной строки под правами администратора. Обрати внимание: мы запускаем IDLE второй, а не третьей версии!

- Введи в командную строку терминала команду

```
$ sudo idle
```

Пишем программу

```
# ldc_test.py
from RPLCD import CharLCD              #1

lcd=CharLCD(cols=16, rows=2)           #2
lcd.write_string('Raspberry Pi')       #3
lcd.cursor_pos = (1, 0)                #4
lcd.write_string('for Kids')           #5
```

Как это работает?

#1 Импорт типа `CharLCD` из модуля `RPLCD`.

#2 Здесь мы создаём объект типа `CharLCD`. Он должен представлять собой ЖК-дисплей. Оба параметра определяют, что дисплей имеет две строки (`rows`) и 16 столбцов (`cols`).

#3 Для объекта `CharLCD` курсор находится в верхней части, в первой строчке слева. В этом месте (в месте нахождения курсора) записывается строка `'Raspberry Pi'`.

#4 Курсор устанавливается в начало второй строки.

#5 В текущую позицию курсора (т. е. в начало второй строки) вписывается строка `'for Kids'`.

Примечания. Символы в Python 2, отличные от ASCII

При сохранении IDLE, скорее всего, нам понадобится поместить следующую строку в начало программного текста:

```
# -*- coding: utf-8 -*-
```

Так происходит, если твой программный текст сохраняет значение, не принадлежащее коду ASCII.

Таблица 13.1 описывает несколько других атрибутов и методов для объектов `CharLCD`. Ты можешь их испробовать в Python Shell:

```
>>> from RPLCD import *
>>> lcd=CharLCD(cols=16, rows=2)
>>> lcd.cursor_pos = (0, 15)
>>> lcd.text_align_mode = Alignment.right
>>> lcd.write_string('Raspberry Pi')
```

Текст `'Raspberry Pi'` записывается в обратном порядке: справа налево.

Таблица 13.1. Несколько атрибутов и методов объектов CharLCD

Атрибуты/Методы	Значение
`clear()`	Удалить содержимое дисплея и установить курсор в начало первой строки
`cursor_mode`	Для режима курсора (скрытые, горизонтальные линии, мигание) существуют константы `Cursor-Mode.hide`, `CursorMode.line` и `CursorMode.blink`

Таблица 13.1 (окончание)

Атрибуты/Методы	Значение
`cursor_pos`	Позиция курсора определяется через кортеж формы (строка, столбец). Нумерация строк и столбцов начинается с нуля
`shift_display(n)`	Переместить текст на дисплее в место `n`. Отрицательные числа влияют на перемещение влево, а положительные – перемещают текст вправо
`text_align_mode`	Для настройки текста существуют константы `Alignment.left` (слева направо) и `Alignment.right` (справа налево)
`write_string(text)`	Запиши указанный текст на дисплее и начни с текущей позиции курсора

Кусочек целого. Срез

Для следующего проекта нам понадобится техника под названием «слайсинг» (срез). Благодаря ей ты сможешь сформировать фрагмент из секвенции.

```
>>> w = 'Raspberry'
>>> w[4:7]
'ber'
```

Английское слово *slice* означает «ломтик». Представь, что кто-то, используя магический нож, отрезает кусок пирога или, что ещё лучше, *копирует* его. При этом сам пирог никак не изменяется.

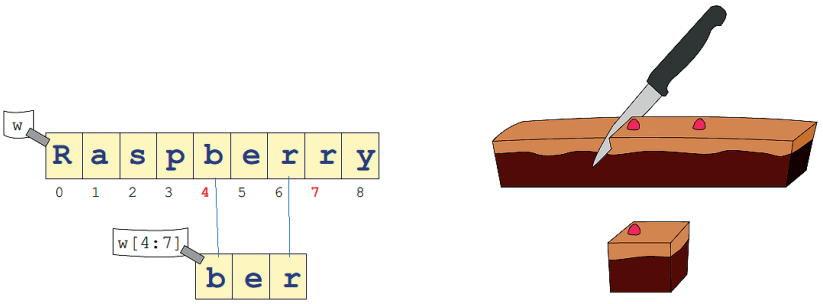


Рис. 13.5. *При нарезке копируется кусок от целого предмета*

Срез состоит из имени секвенции (в данном примере это `w`), за ним следует так называемый *список фрагментов,* заключённый в квадратные скобки. В нашем примере это

фрагменты с индексами [4:7]. Это элементы с индексами 4, 5 и 6. Элемент с индексом 7 в список фрагментов не входит и обозначает, что предыдущий элемент с индексом 6 является в списке заключительным. Обе цифры [4:7] являются маркировкой, как ягоды малины на пироге, показанные на рис. 13.5. Первая цифра – это индекс, с которого должен начинаться фрагмент. Вторая цифра, находящаяся сразу после двоеточия, – индекс, *перед которым* заканчивается срез. Будь здесь особенно внимательным. Элемент с индексом 7 больше к срезу не относится.

Представим, что ты делишь последовательность на две части. Значит, второй индекс первого среза является первым индексом второго среза:

```
>>> w='Raspberry'
>>> w[0:4]
'Rasp'
>>> w[4:]
'berry'
```

Если опустить второй индекс, фрагмент идёт до конца последовательности:

```
>>> w = Maus'
>>> w[1:]
'aus'
```

Проект 35. Цифровые часы с ЖК-индикатором

Мы запрограммируем часы, которые будут отображать дату и время, как показано на рис. 13.6.

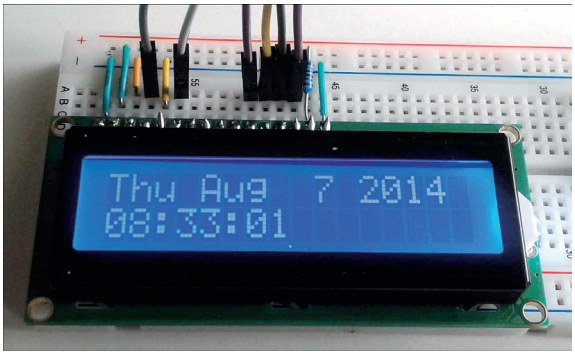


Рис. 13.6. *Цифровые часы: вверху отображается дата, внизу – текущее время*

Пишем программу

```
# lcd_clock.py
import time
from RPLCD import CharLCD

lcd = CharLCD( cols=16, rows=2)

while True:                                    #1
    время = time.asctime()
    lcd.cursor_pos = (0, 0)                    #2
    line_1 = время[:10] + ' ' + время[-4:]#    #3
    line_2 = время[11:19]                      #4
    lcd.write_string(line_1)                   #5
    lcd.cursor_pos = (1, 0)                    #6
    lcd.write_string(line_2)
    time.sleep(0.1)
```

Как это работает?

#1 Это бесконечный цикл. Для завершения работы программы нажми комбинацию клавиш **Ctrl**+**C**.

#2 Курсор помещается в верхнюю левую часть дисплея.

#3 Первая строка собирается из трёх частей: сначала идут первые десять значений строки символов, затем пробел, и потом последние четыре значения строки символов.

#4 Вторая строчка состоит из средней части строки символов. Это те же самые символы, которые отображают время.

#5 Первая строчка (`line_1`) вписывается в текущую позицию курсора, т. е. в начало первой строки на дисплее.

#6 Помещаем курсор в начало второй строки.

Проект 36. Таймер

Таймер управляется одной-единственной клавишей. На рис. 13.7 ты увидишь макетную плату с маленькой кнопкой (в правой части рисунка). Вместо этой кнопки можно использовать любую другую кнопку. Один контакт этой кнопки через провод с контактами типа «папа-мама» подключается к контакту 3 разъёма GPIO. Второй контакт кнопки через проволочную перемычку подключается к отрицательной линии питания макетной платы (отмеченной синей линией), соединённой с корпусом.

Но сначала посмотри на ЖК-индикатор, показанный на рис. 13.7. Если нажать кнопку, а потом отпустить, таймер запустится. В первой строке он показывает количество секунд, прошедших с момента старта. Внизу появится примечание: «Button stop» (кнопка стоп).

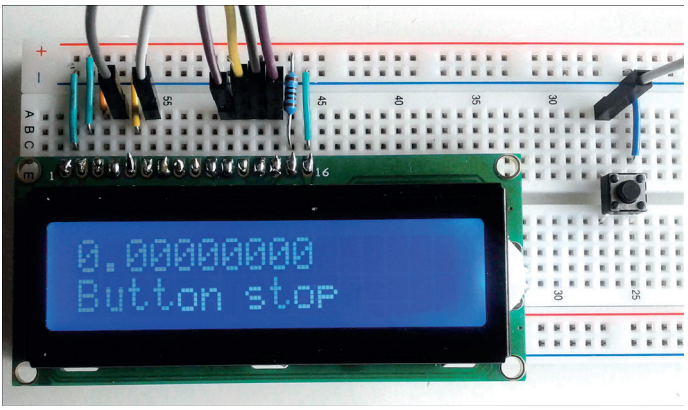


Рис. 13.7. *Так выглядит ЖК-индикатор таймера после запуска программы*

При следующем нажатии этой кнопки таймер остановится, и ты увидишь, сколько прошло секунд между первым нажатием этой кнопки, запустившим таймер, и вторым нажатием кнопки, таймер остановившим. Ты увидишь большое количество цифр, разделённых запятыми. Во второй строке ты увидишь примечание «Button restart» (кнопка перезапуска). Сообщения не должны содержать более 16 символов, включая пробелы.

Диаграмма перехода состояния (рис. 13.8) точно показывает, что происходит. Во время одного цикла таймер проходит шесть состояний. После каждого нажатия и отпускания клавиши состояние таймера меняется.

Эта диаграмма является базой для программирования нашего проекта.

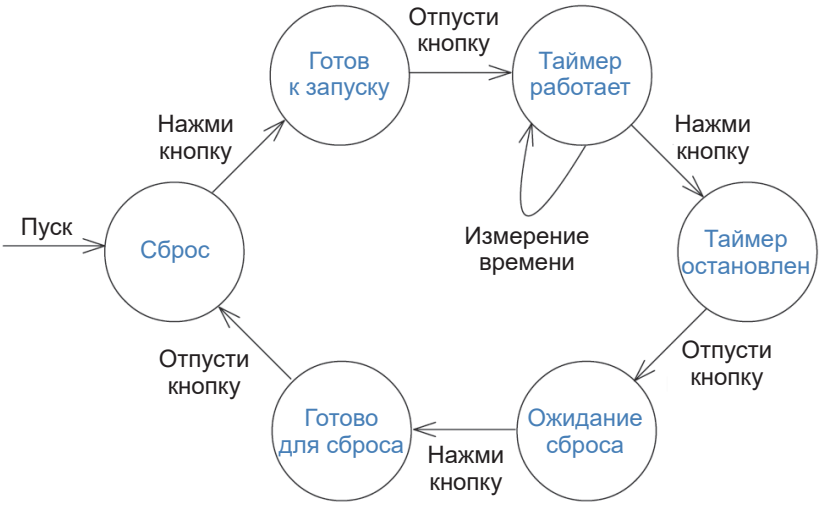


Рис. 13.8. *Шесть состояний таймера*

Аппаратное обеспечение

Дополни схему, чтобы кнопка переключения соединяла контакт 3 GPIO с контактом 6 (корпусом). Из-за внутреннего нагрузочного сопротивления, если кнопка *не* нажата, контакт 3 находится в состоянии `True` (если установлен в качестве входного канала).

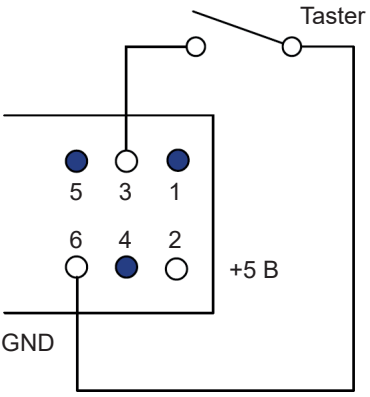


Рис. 13.9. *Подключение клавиши к контакту 3 GPIO*

Пишем программу

```
# lcd_timer.py
import time
from RPi import GPIO
from RPLCD import CharLCD
GPIO.setmode(GPIO.BOARD)
GPIO.setup(3, GPIO.IN)
lcd=CharLCD(cols=16, rows=2)

while True:
    time.sleep(0.1)                          #1
    lcd.clear()                              #2
    lcd.write_string('0.00000000')
    lcd.cursor_pos =(1, 0)
    lcd.write_string('Button start')
    while GPIO.input(3):                     #3
        pass
    while not GPIO.input(3):                 #4
        pass
    start = time.time()                      #5
    lcd.write_string('Button stop')
    time.sleep(0.1)
    lcd.cursor_pos =(1, 0)
    while GPIO.input(3):                     #6
        lcd.cursor_pos =(0, 0)
        time = time.time() - start           #7
        lcd.write_string(str(time)[:10]
    while not GPIO.input(3):                 #8
        pass
    lcd.cursor_pos =(1, 0)
    lcd.write_string('Button stog')
    time.sleep(0.1)
    while GPIO.input(3):                     #9
        pass
    while not GPIO.input(3):                 #10
        pass
```

Как это работает?

#1 Ожидание 0,1 с. Этот период ожидания встроен в несколько мест программы. Он препятствует прохождению ложного сигнала нажатия кнопки (дребезг контактов).

#2 Очистка индикатора от введённых ранее данных. Курсор находится в верхней левой части дисплея.

#3 Время находится в состоянии `отсрочки`. Ожидание нажатия кнопки.

#4 Состояние `Готов к запуску`. Ожидание, когда клавиша будет отпущена.

#5 Запоминается момент нажатия кнопки.

#6 Состояние `Таймер работает`. До тех пор пока кнопка не нажата, работает бесконечный цикл `while`.

#7 Переменная `time` накапливает сведения о времени, прошедшем с момента запуска таймера.

#8 Состояние `Таймер остановлен`. Ожидание, когда кнопка будет отпущена. Отображается общее время работы таймера за прошедший цикл.

#9 Состояние `Ожидание сброса`. Ожидание нажатия кнопки.

#10 Состояние `Готово к сбросу`. Ожидание, когда кнопка будет отпущена. Только после этого будет выполнен сброс данных и таймер вернётся в исходное состояние. После обнуления на дисплее отобразится время `0.00000000`.

Вопросы

1. Что означает аббревиатура LCD?
2. Функция `asctime()` из модуля `time` возвращает строку с датой и временем. Какое значение имеют первые три показателя этой строки?
3. Каков результат работы оператора `time.asctime()[-4:]`?

Задание. Блуждающие звёзды

Подключи ЖК-дисплей, как показано на рис. 13.10, и разработай программу, которая выполнит следующее: на ЖК-дисплее вправо-влево перемещаются три звёздочки.

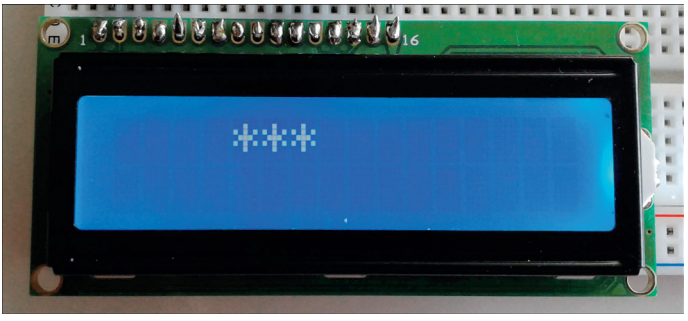


Рис. 13.10. *Создай на дисплее три звезды, перемещающиеся вправо-влево*

Совет: используй следующие инструкции:

```
import time
for I in range(13):
from RPi import GPIO
from RPLCD import CharLCD
lcd = CharLCD(cols=16, rows=2)
lcd.clear()
lcd.shift_display(1)
lcd.shift_display(-1)
lcd.write_string(text)
text = '***'
time.sleep(0.1)
while True:
    ...
```

Ответы на вопросы

1. Аббревиатура LCD расшифровывается как *liquid cristal display*, что означает «жидкокристаллический дисплей».
2. Первые три буквы показывают день недели, правда, на английском языке. Пример: `Thu` – это *Thursday* (четверг).
3. Вызов функции `time.asctime()[-4:]` выдаёт текущий год в качестве строки символов, например `'2019'`. Это последние четыре показателя строки символов, которые демонстрирует функция `time.asctime()`.

Решение задачи

Пишем программу

```
import time
from RPi import GPIO
from RPLCD import CharLCD

lcd=CharLCD(cols=16, rows=2)
text = '***'
lcd.clear()
lcd.write_string(text)
while True:
    lcd.clear()
    lcd.write_string(text)
    for I in range(13):        #1
        lcd.shift_display(1)   #2
        time.sleep(0.1)
```

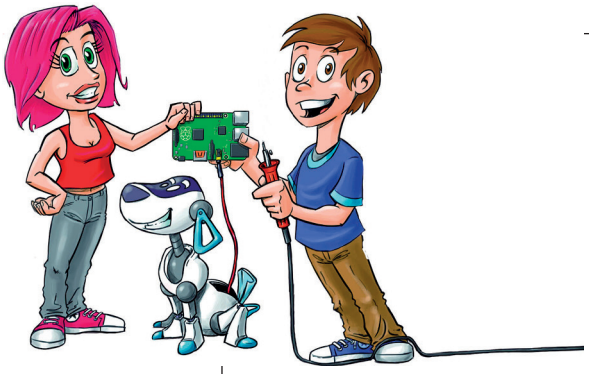
```
    for I in range(13):
        lcd.shift_display(-1)  #3
        time.sleep(0.1)
```

Как это работает?

#1 Вложенный блок повторяется 13 раз. Почему именно 13? Дисплей имеет 16 позиций для ввода символов. Три из них заняты звёздочками. Остаются ещё 13 символов, чтобы им было где «разгуляться».

#2 Содержимое дисплея смещается на одну позицию вправо. То есть три звезды переходят на одну позицию вправо.

#3 Здесь звёздочки передвигаются на одну позицию влево.



14 Проекты с использованием ультразвукового датчика

С помощью ультразвукового датчика Raspberry сможет измерять расстояния и скорость. Значения отображаются на экране или передаются с помощью акустических сигналов (например, для слабовидящих). Если ты оснастишь свой Raspberry батареей, то он станет полностью мобильным. Тогда ты сможешь повсюду брать его и даже в темноте с помощью ультразвукового датчика сумеешь распознать препятствия.

Эту главу мы начнём с техники программирования Python, которая может определить уникальные функции.

Какие бывают функции?

Ты в своих программах, написанных на языке Python, неоднократно использовал предварительно заданные функции. Некоторые из них являются стандартными

функциями, некоторые – модулями, которые нужно было импортировать. То есть *функция* – это программный код, к которому ты можешь обратиться с любого места программы. В основном функция предназначена для работы с данными, которые передаются ей в качестве аргументов. Также функция может формировать некоторое возвращаемое значение. В этом разделе ты узнаешь, как определить функции. И вообще, зачем их определять? Ведь их и так уже много.

Ну, прежде всего функции обладают двумя преимуществами:

- длинный программный текст может быть разделён на маленькие чёткие фрагменты;
- последовательность функций, которые выполняются чаще всего, может быть объединена в одну функцию. Это немного укорачивает программный текст.

Как определить функцию?

Функцию можно определить через определённую инструкцию, имеющую следующий формат:

```
def имя функции(...):
    блок инструкций
```

Первая строка – это *заголовок функции*. Он всегда начинается с кодового слова `def`, что означает «определение». Затем идёт имя функции. В круглых скобках могут указываться параметры. Это имена значений, которые при вызове функции заключены в скобки. Но часто скобки остаются пустыми. Это значит, что функция `stop` никакие параметры не обрабатывает. Такое происходит довольно часто. В конце заголовка всегда ставится двоеточие.

В строке, после двоеточия, находится *тело функции*. Это блок инструкций с отступом на несколько пробелов. Эти инструкции выполняются при вызове данной функции.

Простая функция

Начнём с примера. Мы напишем функцию, которая, в свою очередь, напишет пожелание ко дню рождения.

Как же нам назвать эту функцию? Имя функции должно выражать выполняемое ею действие. Назовём её, например, `write()`.

Имя функции

Имя функции (как и имя переменной) может состоять из букв, цифр или нижнего подчёркивания. Начинаться оно должно с буквы (желательно строчной, а не прописной) или нижнего подчёркивания. Имя должно быть выражением того действия, которое эта функция выполняет или вычисляет. Обрати внимание: имя функции (как и переменной) можно написать и кириллицей. Но такое имя (как переменной, так и функции) может вызвать в дальнейшем большие проблемы. Например, если программа устанавливается на другом компьютере, который кириллицы «не знает». И еще. Желательно, чтобы длина имени функции (как и переменной) не превышала 16 знаков.

Допускаются такие названия, как `minimum()`, `distribute_list()` или `_initials_()`. *Не* допускаются названия следующего типа: `2_allocate()` – начинается с цифры или `a+b()` – содержит недопустимый символ +.

В следующей программе мы определим функцию с двумя параметрами. Первый параметр указывает, как нужно обратиться к тому, кому адресовано поздравление, а второй параметр – имя отправителя. Напиши программный код в оболочке IDLE 3 и сохрани его как программный файл, например под именем *поздравление.py*.

Пишем программу

Сейчас пришло время написать первую программу.

- Запусти программную оболочку IDLE, выбери команду меню **File ⇨ New File** (Файл ⇨ Новый файл), сохрани создаваемую функцию под именем *поздравление.py*.
- Введи код программы:

```
def write(обращение, отправитель):          #1
    print(обращение + '!')                  #2
    print('Поздравляю с днём рождения')     #3
    print('желаю тебе всех благ')
    print(отправитель)                      #4
```

Как это работает?

#1 Это заголовок функции. Слово `обращение` является первым параметром, а слово `отправитель` – вторым. Параметры похожи на переменные. Параметр сохраняет значение, переданное в качестве аргумента при вызове функции.

#2 Здесь начинается блок инструкций для определения функции. После запуска функции в первую очередь выводится содержимое параметра `обращение` с вложенным восклицательным знаком.

#3 Теперь выводится заложенное в функцию приветствие. Для данной функции этот параметр неизменяемый и зависит только от введённого при написании функции текста.

#4 На экран выводится имя отправителя, а именно содержимое параметра `отправитель`.

➢ Запусти программный файл, нажав клавишу **F5**. В окне Shell ты увидишь сообщение о перезапуске.

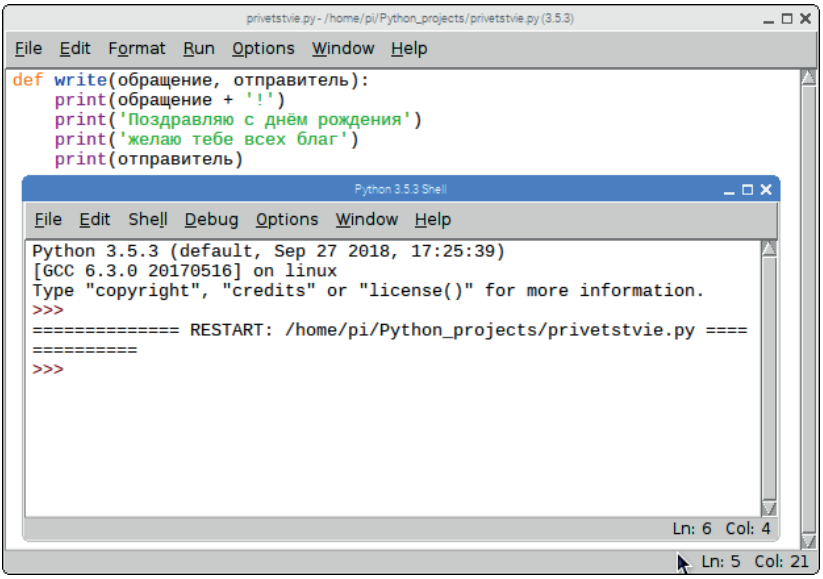


Рис. 14.1. *После запуска программного файла с помощью определения функции откроется окно Shell с сообщением о перезапуске*

После появления сообщения о перезапуске на экране ничего больше происходить не будет. Компьютер будет ожидать вызова функции. Чтобы вызвать функцию, введи в командную строку имя и аргументы функции и нажми клавишу ↵. Если имя функции введено на кириллице, выбери кириллицу, введи имя функции, далее в круглых скобках через запятую – аргументы и нажми клавишу ↵. Обрати внимание: каждый аргумент должен быть заключён в одинарные верхние кавычки, например:

```
>>> write('Кирилл', 'Вася')
```

В ответ компьютер ответит:

```
Кирилл!
Поздравляю с днём рождения
желаю тебе всех благ
Вася
```

Как ты уже понял, *вызов* функции – это несколько другое действие, нежели *определение* функции. Мы уже функцию `write()` вызвали. Ещё раз напомним, при вызове в скобках после имени функции ставятся значения, как, например, обе строки символов `'Кирилл'` и `'Вася'`. То, что находится в скобках, называют аргументом или, если их несколько, аргументами.

Эти аргументы передаются функции и назначаются параметрами определения функции. Здесь в примере параметр `обращение` получает значение `'Кирилл'`, а параметр `отправитель` получает значение `'Вася'`. Вот как раз с этими значениями функция сейчас «работает» и выдаёт текст поздравления.

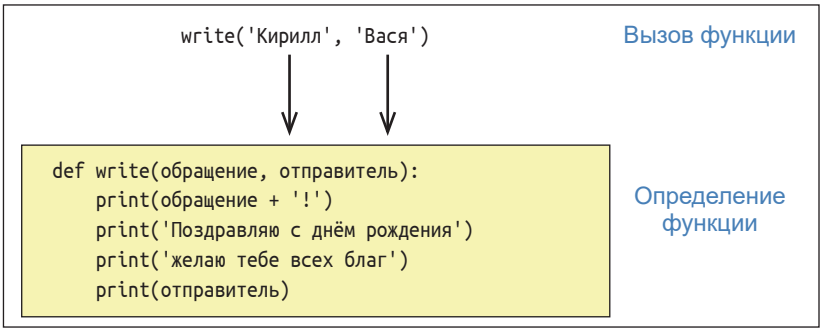


Рис. 14.2. *При вызове функции параметры принимают значения аргументов*

Как вызвать функцию в главной программе?

Функции определяются для того, чтобы можно было вызвать их в основной программе. В этом случае основная программа станет короче, и её будет легче понять. С помощью функции `write()` ты можешь создать интерактивную программу. Сначала пользователю будет предложено ввести обращение и имя отправителя. Затем вызывается функция.

Процесс программы (пример)

```
Автопоздравитель.
Имя адресата: Кирилл
Твое имя: Вася
Дорогой Кирилл!
Поздравляю с днем рождения
желаю тебе всех благ.
Вася
```

Пишем программу

```
def write(обращение, отправитель):              #1
    print(обращение + '!')
    print('Поздравляю с днем рождения!!!')
    print('Желаю тебе всех благ.')
    print(отправитель)

#Главная программа
print('Автопоздравитель')
a = input('Имя адресата: ')                     #2
n = input('Твое имя: ')
write(a, n)                                     #3
```

Как это работает?

`#1` В начале программного текста должно находиться определение функции.

`#2` В этой строке программного кода определяется значение переменной `a`. В нашей программе значение переменной `a` состоит из двух частей: постоянной `Имя адресата:` и изменяемой, в которой находится имя человека, которого мы хотим поздравить. Строка ниже определяет значение переменной `n`, также состоящей из двух частей: постоянной `Твое имя:` и изменяемой, содержащей имя отправителя.

`#3` Эта строка вызывает функцию, которую мы определили выше. То есть в этом программном коде это функция `write()`. Строчки символов, которые сохраняются в переменных `a` и `n`, передаются этой функции и становятся аргументами вызова функции.

А ещё функции возвращают значение

Большинство функций, которые тебе известны, возвращают значение. Что это означает? Если ты вызовешь такую функцию в окне Python, то в следующей строке появится возвратное значение, например:

```
>>> len('Raspberry' )
9
```

Функция `len()` возвратила длину секвенции. Строка символов Raspberry состоит из девяти знаков, поэтому её длина равна 9. В этом примере функция возвращает число.

Когда вы вызываете функцию, такую как `len()`, это похоже на заданный вами собеседнику вопрос. Ответ – это значение, которое функция вычисляет и возвращает.

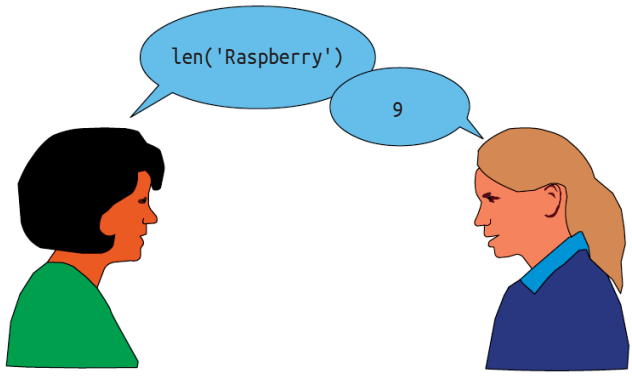


Рис. 14.3. *Вызов функции выглядит как вопрос и ответ*

Кстати, функции, которые ничего не возвращают, называются процедурами. Функция `write()` из последнего раздела как раз является такой процедурой. Она что-то выполняет, но не отвечает на вопрос.

Если функция возвращает значение, то необходимо, чтобы тело функции содержало оператор `return`. Начнём с простейшего примера.

Функция должна вычислить площадь прямоугольного окна. Параметрами являются высота и ширина окна. Как это принято в технике, значение высоты и ширины следует ввести в миллиметрах (мм). Функция вычисляет площадь в квадратных метрах и возвращает значение.

```
def calculate_window(высота, ширина):          #1
area = высота*ширина/10000000                  #2
return area                                    #3
```

Обрати внимание: мы могли бы ввести название функции на кириллице (например: `рассчет_площади_окна`), и такая функция будет работать. Но все программисты во избежание возможных проблем на других компьютерах пред-

почитают вводить имена переменных и функций на латинице. Аргументы функции `высота` и `ширина` тоже было бы корректнее ввести на латинице, взяв английские значения этих слов: `height` (высота) и `width` (ширина).

Как это работает?

`#1` Это заголовок функции с параметрами `высота` и `ширина`.

`#2` Здесь начинается тело функции, т. е. блок инструкций, который вводится с отступом. В первой инструкции вычисляется площадь прямоугольника. Ширина и высота указываются в миллиметрах. Но поскольку площадь должна отображаться в квадратных метрах, а не в квадратных миллиметрах, то необходимо разделить это число на 1 000 000, т. к. в одном квадратном метре содержится один миллион квадратных миллиметров.

`#3` Возвращается результат вычисления. Это и есть вывод функции.

Тестируем функцию

Теперь попробуем выполнить эту функцию.

- Введи программный код функции в окне редактора IDLE и сохрани его в своём каталоге проекта (например, под именем *window.py*).
- Нажми клавишу **F5**. Функция будет запущена, но в командной строке оболочки IDLE ничего не произойдёт. Ты вновь увидишь сообщение о перезапуске: `RESTART`.
- Введи в командную строку окна Python команду вызова функции и нажми клавишу ↵. Будет рассчитана площадь окна высотой 1000 мм и шириной 500 мм.

```
>>> calculate_window (1000, 500)
0.05
```

Как обычно, в строке под вызовом функции появится результат. Это значение, которое возвращает оператор `return`. Возможно, при вызове функции ты что-то заметил. Как только ты впишешь скобки после имени функции, откроется окно с подсказкой, где ты увидишь имена параметров.

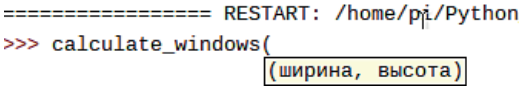


Рис. 14.4. *При вводе имени функции появится окно с параметрами*

Ты можешь вызвать функцию также с помощью аргументов кодового слова. Такой аргумент имеет форму:

```
параметр=значение
```

В этом случае последовательность аргументов не играет никакой роли. Но ты должен знать имена параметров, которые используются в определении функции:

```
>>> calculate_window(ширина=500, высота=1000):
0.05
```

Проект 37. Каков размер окна в доме?

Определим функцию, для того чтобы можно было её вызвать в главной программе. В этом случае понять программу можно будет быстрее и проще. Следующая интерактивная программа вычислит общую площадь всех окон дома.

Запрашиваем высоту и ширину окна, затем вызываем функцию и рассчитываем сумму площади всего окна.

Процесс программы (пример)

```
Высота в мм: 1000
Ширина в мм: 500
Еще одно окно? (д/н) : д
Высота в мм: 1000
Ширина в мм: 1200
Еще одно окно? (д/н) : н
Общая площадь окна составляет: 1.7 квадратных метра
```

Пишем программу

Обрати внимание: чтобы тебе было понятнее, все имена переменных введены на русском языке и кириллицей. Но, как мы уже упоминали ранее, при написании программ во избежание возможных проблем желательно имена переменных и функций писать на латинице.

```
def calculate_window (ширина, высота):          #1
    площадь = ширина * высота / 1000000
    return площадь

ответ = 'д'                                     #2
```

```
сумма = 0                                          #3
while ответ == 'д':                                #4
    в = int(input('Высота в мм: '))                #5
    ш = int(input('Ширина в мм: '))
    сумма = сумма + calculate_window(в, ш)         #6
    ответ = input('Еще одно окно?(д/н): ')         #7
print('Общая площадь окна составляет:',            #8
      сумма, 'Квадратных метров.')
```

Как это работает?

#1 В начале программного кода находится определение функции.

#2 Для первого прохождения цикла `while` переменной `ответ` присваивается значение `'д'`.

#3 В начале цикла, когда ещё не подсчитана площадь окна, переменной `сумма` присваивается значение `0`.

#4 Если переменная `ответ` получает значение `'д'`, выполняется вложенный блок инструкций.

#5 Компьютер ожидает ввода с клавиатуры. Всегда с клавиатуры пользователь вводит строку символов, которые нужно преобразовать в значения, чем и занимается оператор `input`. Далее полученное значение с помощью оператора `int` преобразуется в целое число. Полученное таким образом значение присваивается переменной `в`. Таким же образом мы формируем значение ширины окна, и это значение присваиваем переменной `ш`.

#6 В этой командной строке формируется значение переменой `сумма`. Эта переменная формируется из суммы прежнего значения и рассчитанного значения площади окна. Обрати внимание: для расчёта площади окна используется функция, которую мы определили в верхней части программного кода.

#7 Система спрашивает пользователя, хочет ли он рассчитать значение площади ещё одного окна и приплюсовать это значение к ранее вычисленным значениям окон. Если пользователь нажмёт клавишу **д**, цикл повторится, и после подсчёта площади ещё одного окна это значение приплюсуется к ранее вычисленным значениям площади окон. В конце цикла программа снова спросит у пользователя, нужно ли рассчитать площадь ещё одного окна. Если будет нажата клавиша **н**, выполнение цикла завершится, и программа перейдет к следующей командной строке `#8`.

#8 Выводится на экран результат подсчётов.

Строка документации

Первая строка тела функции может содержать так называемую *строку документации*. Это длинная строка, которая вводится тремя верхними запятыми (или тремя восклицательными знаками). Например:

```
def calculate_window (высота, ширина):         #1
    ''' Рассчитать площадь одного окна

    Параметры - это высота и ширина
    окна в мм. Площадь возвращается
    в квадратных метрах '''
```

Первая строчка строки документации содержит краткое описание действия функции. Затем идёт пробел и вводится остальной текст. В конце опять вводятся три верхние запятые. Общее содержимое строки документации выводится при вызове функции `help()`.

```
>>> help(calculate_window)
Help on function calculate_window в модуле _main_:

calculate_window (высота, ширина)
    Рассчитать площадь одного окна

    Параметры – это высота и ширина окна в мм. Площадь
    выводится в квадратных метрах
```

Первая строчка строки документации имеет особое значение. Она появляется в информационном окне, когда ты вводишь имя функции в редакторе IDLE или окне Shell.

```
>>> calculate_windows(
                      (ширина, высота)
                      Расчёт площади окна
```

Рис. 14.5. *Окно функции со строкой документации*

Предустановленные параметры значений

Функции могут обладать произвольными параметрами, которые при вызове функции можно опустить. Тебе это известно из стандартных опций, таких как `print()` или `round()`.

```
>>> round(1.234)
1
>>> round(1.234, 2) # два разряда после запятой
1.23
```

Значение по умолчанию присваивается произвольному параметру, который находится внутри функции в заголовке функции.

Большинство окон имеет высоту и ширину в 1000 мм. Введём это значение в качестве настройки по умолчанию.

```
def calculate_window (высота=1000, ширина=1000):
...
```

Остаток определения не изменяется. Попробуй эту функцию в окне Python Shell. Вызови её с вводом аргумента и без него:

```
>>> calculate_window ()
1.0
>>> calculate_window (2000)
2.0
```

При первом вызове были приняты оба заданных значения для параметров. При втором вызове был принят аргумент `ширина` заданного значения.

Проекты с использованием ультразвукового датчика

Звук состоит из колебаний воздуха определённых частот, которые воздействуют на наш слух. *Частота звука* – это количество вибраций в секунду. Она, как и все вибрации, измеряется в герцах. Один герц – одно колебание в секунду. Человеческому слуху доступен диапазон частот от 20 до 20 000 Гц (20 килогерц, 20 кГц). Но это у людей с очень хорошим слухом. Обычный человек слышит примерно от 30 до 15 000 Гц. Звук, частота которого выше 20 000 Гц, – это ультразвук. Люди ультразвук уже не слышат. Но это не значит, что звук частотой выше 20 000 Гц не слышат животные. Например, летучие мыши распознают ультразвук до 200 кГц и используют его для ориентации в пространстве. Как они это делают? Летучие мыши издают короткий ультразвуковой крик и ждут его отголосок в виде эха. Чем дольше ультразвуковое эхо возвращается к летучей мыши, тем сама мышь дальше от того препятствия, от которого ультразвук был отражён. Таким образом зверьки ориентируются в полнейшей темноте.

По такому же принципу работают ультразвуковые сенсоры, которые ты можешь приобрести за небольшие деньги. Рисунок 14.6 показывает ультразвуковой датчик US-020. С ним ты сможешь измерить расстояние от 3 до 7 см.

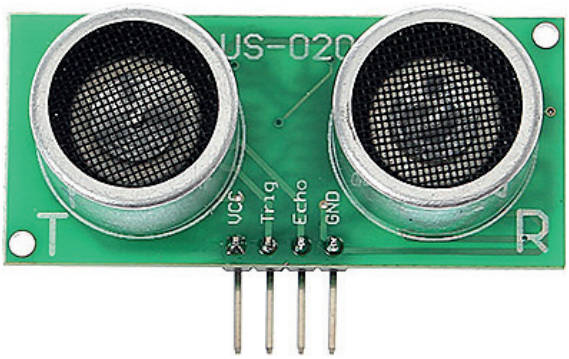


Рис. 14.6. *Ультразвуковой датчик расстояния US-020*

Маркировка US – это сокращение от английского слова *ultrasound* (ультразвук). Плата имеет четыре контакта, с помощью которых её можно подключить к компьютеру. Это следующие контакты (слева направо):

- **VCC**: питание 5 В;
- **Trig** (триггер): это цифровой вход датчика. В состоянии покоя напряжение на этом контакте равно 0 В. Для запуска цикла измерения на этот контакт следует подать положительный импульс длительностью не менее 10 микросекунд. Номинальное напряжение подаваемого импульса запуска 5 В. Цикл измерения состоит из излучения ультразвукового импульса, ожидания отражённого эхо-сигнала и получения отражённого сигнала. Но если на контакт **Trig** подать импульс напряжением 3,3 В, датчик сработает, и измерительный цикл будет запущен. Почему нам это так важно? Дело в том, что выходное напряжение программируемых выводов GPIO равно 3,3 В;
- **Ecxo** (эхо): это цифровой выход датчика. В состоянии покоя напряжение на этом контакте 0 В. Во время всего цикла измерения (ультразвуковой сигнал отправлен – ожидание – эхо-сигнал получен) напряжение на этом контакте будет равно 5 В;
- **GND**: корпус (земля).

На рис. 14.7 наглядно показано, как производится измерение расстояния с помощью ультразвукового импульса.

Когда датчик получает сигнал от Raspberry, излучатель (T, *трансмиттер*, *преобразователь*) отправляет восемь коротких ультразвуковых импульсов (40 кГц). По достижении препятствия этот пакет импульсов от встреченного препятствия отражается, возвращается к датчику и принимается микрофоном (R, *receiver*).

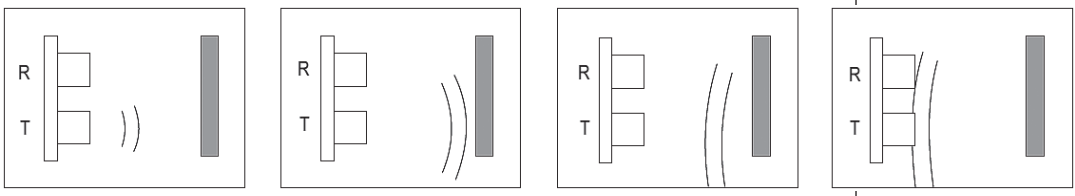


Рис. 14.7. *Измерение расстояния при помощи датчика US-020*

После отправки пакета импульсов в период ожидания отражённого сигнала на контакте **Ecxo** (Эхо) присутствует положительное напряжение 5 В. Когда эхо-сигнал получен, напряжение на контакте **Ecxo** (Эхо) понижается до 0 В. Так формируется П-образный импульс. Расстояние от датчика до препятствия измеряется длиной этого П-образного импульса.

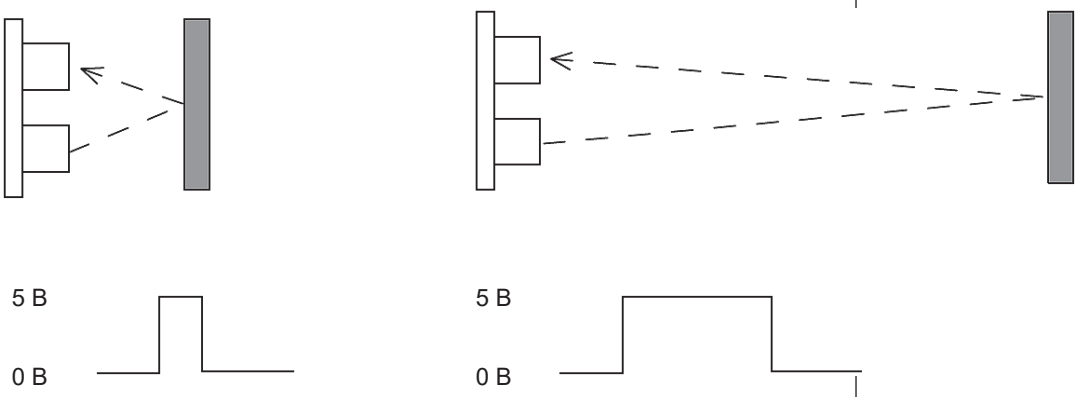


Рис. 14.8. *Расстояние от датчика до препятствия измеряется длиной этого П-образного импульса*

Строение аппаратной части

Цифровой выход эхо-сигнала ультразвукового датчика подключается к контакту разъёма GPIO. Мы для подключения выбрали контакт 10. И здесь возникает проблема. Дело в том, что вход GPIO рассчитан на входное напряжение 3,3 В. Выходное напряжение на контакте **Ecxo** (Эхо) ультразвукового датчика при формировании П-образного сигнала

равно 5 В. Такое напряжение может повредить, как минимум, микросхему Raspberry, к которой подключён разъём GPIO. Поэтому нам нужно понизить напряжение, формирующее исходящий от ультразвукового датчика П-образный импульс. Для этого мы соединим контакт **Ecxo** (Эхо) ультразвукового датчика с контактом 10 разъёма GPIO не напрямую, а через делитель напряжения, состоящий из двух резисторов. Схема подключения ультразвукового датчика к контактам разъёма GPIO показана на рис. 14.9.

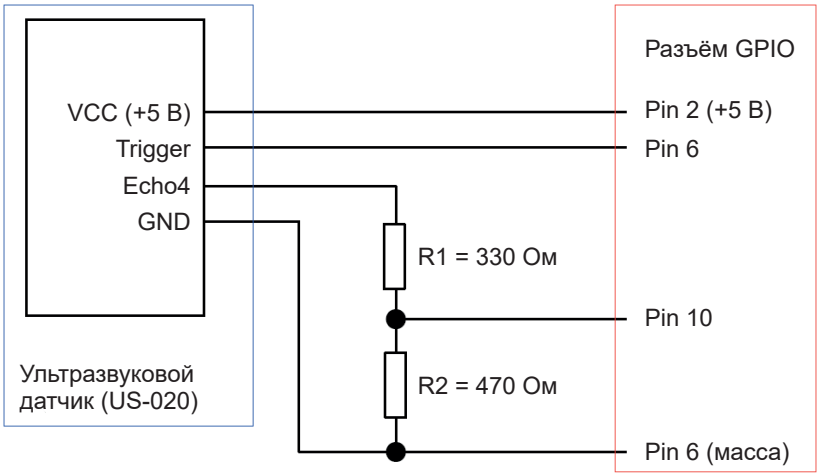


Рис. 14.9. *Схема подключения ультразвукового датчика к контактам разъёма GPIO*

Ты можешь использовать и другие номиналы резисторов. Главное – сохранить соотношение `R1 / R2 = 330 : 470 = 0,7`. То есть если ты возьмёшь для `R1` резистор номиналом 500 Ом, то значение резистора `R2` должно быть равно 710 Ом.

Как работает делитель напряжения?

Если этот вопрос тебя не интересует, то данный раздел ты можешь смело пропустить.

Делитель напряжения состоит из двух резисторов `R1` и `R2` и способен разделить общее напряжение `Uges` на два меньших напряжения `U1` и `U2`. Таким образом, ты подберёшь напряжение, необходимое для нормальной работы контакта 10 разъёма GPIO.

Согласно закону Ома напряжение, приложенное между контактом 10 GPIO и землёй (корпусом), высчитывается по следующей формуле:

```
U = R2 × I.
```

При этом I – это сила тока, текущего через резистор R2. Ток такой же силы течёт и через резистор R1, так как эти два резистора включены последовательно. То есть, согласно закону Ома, через делитель будет течь ток, рассчитанный по формуле

```
I = Uges / (R1 + R2).
```

При этом Uges – это напряжение 5 В, снимаемое с контакта **Ecxo** (Эхо) платы ультразвукового датчика. Это напряжение нужно поделить на делителе, состоящем из двух резисторов.

Если мы в первую формулу U = R2 × I подставим вместо значения силы тока I вторую формулу I = Uges/(R1 + R2), то получим следующее выражение:

```
U = Uges × R2 / (R1 + R2).
```

Это же соотношение ты можешь использовать для всех делителей напряжения. Если мы подставим в полученную формулу значения напряжения Uges, R1 и R2, то получим следующее выражение:

```
U = 5 В × 470 Ом / (330 Ом + 420 Ом);
U = 5 В × 0,5875 = 2,94 В.
```

Проект 38. Измерение расстояния

Устанавливаем Raspberry Pi с ультразвуковым датчиком на столе. Ты подводишь руку к сенсору, а потом её убираешь. Каждую секунду программа записывает текущее расстояние руки от сенсора до экрана.

Процесс программы

```
Расстояние: 102 см
Расстояние: 30 см
Расстояние: 15 см
Расстояние: 28 см
...
```

Пишем программу

```
# distance.py
import time
from RPi import GPIO
TRIGGER = 8                                          #1
ECHO = 10

def init_gpio():
    ''' Инициализация интерфейса GPIO'''             #
    GPIO.setwarnings(False)                          #2
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TRIGGER, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.output(TRIGGER, False)
    time.sleep(1)                                    #3

def send_pulse():
    '''Послать импульс длиной 10 микросекунд'''
    GPIO.output(TRIGGER, True)                       #4
    time.sleep(0.00001)                              #5
    GPIO.output(TRIGGER, False)                      #6

def take_echo():
    '''Возвращает длину эхо-сигнала'''
    while GPIO.input(ECHO) == False                  #7
        pass
    start = time.time()                              #8
    while GPIO.input(ECHO) == True:                  #9
        pass
    stop = time.time()
    return stop - start                              #10

def calculate_distance():
    '''Отправить расстояние в см'''
    send_pulse()
    time = take_echo()
    distance = (time * 34000)/2                      #11
    return round(distance)
# Главная программа
init_gpio()
while True:                                          #12
    p = calculate_distance()
    print('Расстояние: ', p, 'см')                   #13
    time.sleep(1)
```

Как это работает?

#1 Здесь мы определяем две константы: TRIGGER и ECHO. Константы – это переменные, чьё значение в программе не изменяется. Обычно названия констант пишут

большими буквами. Значения этих двух констант (TRIGGER и ECHO) обозначают номера контактов разъёма GPIO, к которым подключаются контакты Trig (Trigger) и Echo платы ультразвукового датчика. Если же ты подключишь контакты Trig (Trigger) и Echo к другим контактам разъёма GPIO, значит, тебе нужно в программу ввести реальные номера контактов, к которым подключены выводы Trigger и Echo.

#2 Эти командные строки предназначаются для блокировки оповещений, выводимых через контакты, не предназначенные для этого. Предназначена она для «красивого» вывода данных и в принципе может быть пропущена.

#3 Ожидание готовности к работе US-датчика. Время ожидания – 1 с.

#4 Отправляем короткий импульс напряжением 3,3 В на контакт Trig (Trigger) платы ультразвукового датчика. Этот импульс запускает цикл измерения, на контакте Echo ультразвукового датчика устанавливается напряжение 5 В, и начинается формирование П-образного импульса.

#5 Время ожидания 10 микросекунд. Это длительность импульса, подаваемого на контакт Trig (Trigger) платы ультразвукового датчика.

#6 На контакте Trig (Trigger) напряжение становится равным 0 В. Импульс, запустивший измерительный цикл, завершён.

#7 Ожидание отражённого эхо-сигнала. Назначение команды pass – избежать внутреннего цикла (петли).

#8 Эхо-сигнал получен. Значение времени запуска измерительного цикла (отправки ультразвукового импульса) записывается в переменной start. Это значение мы получаем с помощью функции time() из модуля time. Этот модуль (time.time()) обеспечивает точное время с 1 января 1970 года, с 1 часа ночи (начала эпохи) в секундах. Так как мы импортировали общий модуль (import time), то мы должны вставить имя модуля time() в функцию.

#9 Цикл длится до тех пор, пока не будет получен эхо-сигнал. Всё это время на контакте Echo платы ультразвукового датчика будет присутствовать высокий уровень напряжения (True) и формирование П-образного импульса продолжается. Завершение цикла произойдёт, как только будет получен отражённый от препятствия ультразвуковой импульс (эхо-сигнал).

`#10` В переменной `stop` записывается время получения эхо-сигнала. Это возвращаемое значение.

`#11` Эта функция (`calculate_distance()`) вычисляет расстояние от ультразвукового датчика до препятствия. Сначала вычисляется разница во времени между отправкой ультразвукового сигнала и получением отражённого эхо-сигнала. Это значение вычисляется с помощью определённой ранее функции `take_echo()` и присваивается переменной `time` (`time = take_echo()`). Далее вычисляется расстояние от ультразвукового датчика до препятствия. Вычисленное значение присваивается переменой `distance`. Как ты помнишь, чтобы вычислить расстояние, нужно время, за которое это расстояние было пройдено, умножить на скорость движения. Так как у нас в переменной `time` записано время, затраченное ультразвуковым импульсом на преодоление расстояния от ультразвукового датчика до препятствия и обратно, полученное значение времени нужно разделить на два. А скорость – это скорость распространения звука в атмосфере, она равна 34 000 см/с. Исходя из вышесказанного, расстояние от ультразвукового датчика вычисляется по следующей формуле: `distance = (time × 34000) / 2` (рис. 14.8).

`#12` Запускается бесконечный цикл, в котором полученное значение расстояния передаётся переменной `p` (`p = calculate_distance()`). Далее эта переменная участвует в выводе результатов.

`#13` Этот оператор (`print('Расстояние: ', p, 'см')`) выводит на экран полученный результат. Вывод состоит из трёх частей: слова `Расстояние`, значения переменной `p` и букв `см`.

Программа работает в режиме бесконечного цикла. Для завершения работы программы нажми комбинацию клавиш **Ctrl**+**C**.

Караул! Программа зависла! Что делать?

Если программа долго работает, рано или поздно она даст сбой. Сбой может произойти после сотни-другой циклов измерений. В этом случае на экране ты не увидишь, что данные не изменяются. То есть программа «зависла», и мы отправляемся на поиски ошибок.

Если после того как программа зависла, ты завершишь её с помощью нажатия комбинации клавиш **Ctrl**+**C**, то получишь примерно такое сообщение:

```
^Ctraceback (most recent call last):
  File "distance.py", line 42, in <module>
    p = calculate_distance()
  File "distance.py", line 34, in calculate_distance()
    time = take_echo()
  File "distance.py", line 23, distance.py
    while GPIO.input(ECHO)== 0:
KeyboardInterrupt
```

Первые два знака – это отголосок комбинации клавиш **Ctrl**+**C**, которые ты нажал. Затем идёт фактическое сообщение. Оно указывает, что последняя инструкция, которая проводилась перед сбоем программы, была такой:

```
while GPIO.input(ECHO)== 0:
```

В этом цикле в функции `take_echo()` застрял интерпретатор (#6 в вышеозначенном списке). Из этого цикла он не появляется. Почему?

Это связано с некорректной работой ультразвукового датчика. Импульс триггера *не всегда* запускает ультразвуковое измерение. После отправки импульса на запуск триггера (начала следующего цикла измерений) программа ждёт появления на контакте Echo платы ультразвукового датчика положительного напряжения, формирующего начала П-образного импульса. Но если ультразвуковой датчик дал сбой, а импульс на запуск триггера был отправлен на контакт Trig ультразвукового датчика, программа ожидает появления эхо-сигнала и будет его ждать до бесконечности. То есть программа «зависла», и поэтому процесс приостановлен. На такую ошибку система не настроена. Но можно внести изменения с помощью паузы.

Представь, что ты стоишь на остановке и ждёшь автобуса, а его всё нет. Он и не придёт, потому что дорога заблокирована упавшим деревом (но ты об этом не знаешь). Как ты поступишь в этой ситуации? Предположим, что ты ждёшь ещё какое-то время, и если автобус не появляется, твоему терпению приходит конец. Тогда скрепя сердце ты отправляешься пешком либо звонишь родителям, чтобы они тебя забрали. Максимальное время, которое ты потратишь на ожидание, прежде чем предпримешь какие-то действия, называется *тайм-аутом*. Сейчас мы изменим программу и сделаем её невосприимчивой к неполадкам, т. е. установим в неё такой тайм-аут.

Пишем программу

Тебе необходимо модифицировать программу в некоторых местах. Места эти выделены жирным шрифтом. Если ты хочешь исправить старую версию программы, то сохрани её под новым именем.

```
# distance_timeout.py
import time
from RPi import GPIO
TRIGGER=8
ECHO=10
TIMEOUT = 0.1                                    #1
...

def take_echo():
    '''Возвращает длину эхо-сигнала.
       В случае если измерение не удалось,
       возвращается значение 0 '''
    zero_time = time.time()                      #2
    while GPIO.input(ECHO)==0:
        if time.time() > zero_time + TIMEOUT:    #3
            return 0                             #4
    start = time.time()                          #5
    while GPIO.input(ECHO)==1:
        pass
    stop = time.time()
    return stop - start
...
# Главная программа
init_gpio()
while True:
    a = calculate_distance()
    if p > 0:                                    #6
        print(I, 'Расстояние:' a, 'cm')
    time.sleep(1)
```

Как это работает?

#1 Вводится одна дополнительная константа: время тайм-аута должно составлять 0,1 с.

#2 Здесь мы запоминаем время начала ожидания эхо-импульса. Этот период мы назовём *нулевым временем*, переменную назовём `zero_time` (нулевое время).

#3 Если тайм-аут – максимальное время ожидания и оно превышено…

#4 … функция возвращает 0, и выполнение функции завершается. Последующие инструкции больше не вы-

полняются, так как, вероятно, произошла ошибка. Но выполнение самой программы продолжается, и программа не блокируется.

#5 Теперь ты можешь быть уверен, что ошибки нет, иначе был бы ещё один тайм-аут. Программа продолжает работу в нормальном режиме.

#6 В случае ошибки расстояние до препятствия считается равным 0, но это ошибочное значение не отображается. Инструкция `print()` выполняется, когда выводимое значение будет больше нуля.

Проект 39. Ориентация в пространстве с помощью ультразвука

Может ли измеритель расстояния быть полезным для слабовидящих людей? Сможет ли он помочь им распознать препятствия, чтобы их обойти? Можно ли, подобно летучим мышам, с помощью ультразвука человеку ориентироваться в помещении?

Всё это мы протестируем в нашем проекте. Чтобы систему можно было использовать в полной темноте, необходимо соблюдать четыре правила.

- Прибор должен быть мобильным, т. е. небольшим и переносным.
- Нам необходим акустический выход.
- При включении Raspberry Pi программа должна запускаться автоматически.
- Прибор должен уметь самостоятельно завершать выполняющуюся программу и выключаться после нажатия кнопки (Shutdown). Просто отключать электропитание нельзя, так как в этом случае SD-карта может быть повреждена (*см.* главу 1).

Подготовка аппаратной части

Мобильный ультразвуковой дальномер состоит из следующих деталей (рис. 14.10):

- Raspberry Pi;
- наушники, которые подключаются к аналоговому аудиовыходу;

- макетная плата с ультразвуковым модулем и прилагаемая к ней электросхема (делитель напряжения), показанные на рис. 14.9;
- аккумулятор с USB-выходом (блок питания), который при напряжении в 5 В даёт минимум 1000 мА;
- дополнительный выключатель для Raspberry Pi (рис. 14.11). Он подключается к контактам 3 и 6 GPIO (контакт 6 – это корпус). Контакт 3 уже имеет встроенный нагрузочный резистор. Поэтому при разомкнутом выключателе на нём будет положительное напряжение 3,3 В.

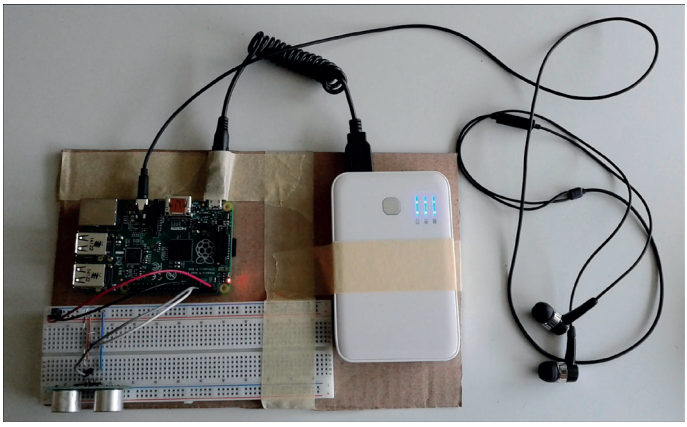


Рис. 14.10. *Мобильный Raspberry Pi: все детали скреплены с куском картона*

Все детали ты можешь закрепить на куске картона с помощью клейкой ленты.

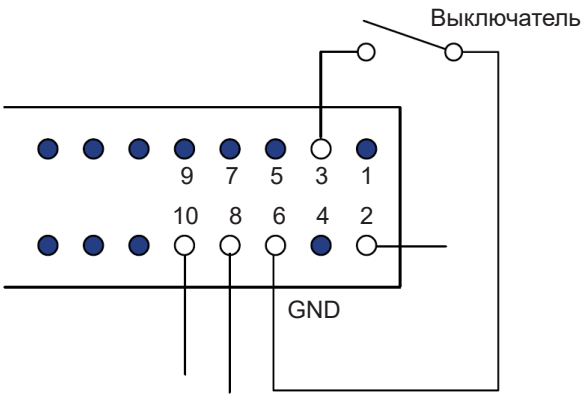


Рис. 14.11. *Выключатель для Raspberry Pi*

Информация о расстоянии до препятствия будет передаваться через наушники. Формат вывода сигнала может быть следующий:

- голосовой вывод, когда значение расстояния озвучивается голосом;
- тональность сигнала (чем ближе к препятствию, тем выше тональность);
- повторение сигнала с различной скоростью (чем ближе к препятствию, тем чаще поступает звуковой импульс).

В следующей программе мы воспользуемся третьим вариантом обозначения расстояния до препятствия. У нас будет звуковой файл, содержащий короткий сигнал, например «Бип!». Этот звук будет повторяться снова и снова. Чем короче время ожидания между двумя сигналами, тем ближе объект.

«Бип! Бип! Бип!» означает: «Осторожно! Впереди препятствие!».

«Бип! – – – – – – – Бип! – – – – – – – Бип!» означает: «Опасности нет, следующий объект находится на безопасном расстоянии».

В папке проекта к этой главе имеется подпапка с несколькими подходящими звуковыми файлами в формате wave. А также в музыкальном архиве Scratch ты найдёшь несколько звуков: */usr/share/scratch/Media/Sounds*. Просто скопируй в свой проект любой wave-файл с коротким звуковым эффектом, который тебе понравится.

Проигрываем звуковые файлы

На твоём Raspberry Pi установлен *Advanced Linux Sound Architecture* (ALSA). С помощью Aplay ты можешь проиграть эти звуковые файлы. Попробуй:

- подключи наушники;
- открой LXTerminal;
- с помощью команды `cd` перейди в каталог проекта звуковых файлов (`cd usr/share/scratch/Media/Sounds/Animal`).
- следи за тем, чтобы звук передавался через аналоговый вывод наушников (не через HDMI). В строке Linux эта коммутация выполняется с помощью команды:

```
amixer cset numid=3 1
```

Примечание: последняя цифра определяет, какой аудиоканал используется. 0 = автоматический, 1 = аналоговый, 2 = HDMI.

- Чтобы проиграть файл, например *Kitten.wav*, введи команду `aplay Kitten.wav` и нажми клавишу ↵.

Пишем программу

В этой программе используются все функции, определённые в предыдущей программе *distance_timeout.py*. Для этого проекта нам потребуется программу *distance_timeout.py* в нескольких местах доработать. Ниже приведён листинг программы, изменения в котором выделены жирным шрифтом.

Обрати внимание: в командной строке `os.system('aplay sounds/boing.wav')` предлагается воспроизвести файл *boing.wav*. Но, как уже ранее было сказано, звуковой файл нужно скопировать в папку, в которой находится сама программа. И ты можешь скопировать туда любой короткий звуковой файл, например *Kitten.wav* из папки *usr/share/scratch/Media/Sounds/Animal*. В программе нужно ввести имя звукового файла, который ты сохранил в папке с программой. Если ты, например, скопировал файл *Kitten.wav*, командная строка `os.system('aplay sounds/boing.wav')` должна выглядеть так: `os.system('aplay sounds/Kitten.wav')`.

```
# distance_ton.py
import os
import time
TRIGGER = 8
ECHO = 10
SWITCH = 3
TIMEOUT = 0.1

def init_gpio():
    '''Инициировать интерфейс с GPIO'''
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TRIGGER, GPIO.OUT)
    GPIO.setup(ECHO, GPIO.IN)
    GPIO.setup(SWITCH, GPIO.IN)
    GPIO.output(TRIGGER, False)
    time.sleep(1)

...
# Главная программа
init_gpio()
while GPIO.input(SWITCH):                    #1
    pause = calculate_distance ()/100        #2
    os.system('aplay sounds/boing.wav')      #3
    time.sleep(pause)                        #4
os.system('halt')                            #5
```

Как это работает?

#1 Этот командный блок обеспечивает корректное выключение компьютера. Если контакты выключателя разомкнуты, а это его нормальное состояние, то на контакте 3 разъёма GPIO благодаря встроенному нагрузочному резистору будет присутствовать положительное напряжение 3,3 В (состояние `True`). До тех пор пока контакты переключателя разомкнуты, выполняется цикл `while`. Как только контакты замыкаются, состояние контакта 3 разъёма GPIO изменяется на `False`, и цикл завершается.

#2 Переменная `pause` сохраняет паузу при проигрывании звукового файла. По паузе между двумя звуками мы можем примерно определить расстояние до препятствия. Если расстояние до препятствия 2 м (200 см), время паузы будет равно 2 с. Когда расстояние уменьшится до метра, длительность паузы составит 1 с. То есть при приближении к препятствию сигнал будет звучать чаще.

#3 С помощью функции `system()` из модуля `os` вызывается нужная команда Linux. В этой командной строке мы вызываем команду `aplay`, которая воспроизведёт звуковой файл, имя которого находится в данной командной строке `os.system('aplay sounds/boing.wav')`, а сам аудиофайл находится в папке с программой.

#4 Переменная `pause` определяет время паузы между двумя звуковыми сигналами.

#5 С помощью этой команды компьютер выключается.

Как автоматически запустить программу Python?

Для этого проекта ты используешь Raspberry Pi без клавиатуры, мышки и монитора. При выключении он должен автоматически запускать программу *distance_ton.py*. Допустим, что программа находится в каталоге */home/pi/Python-projects* (у тебя папка, в которой ты хранишь все экспериментальные программы, может называться по-другому).

Теперь делаешь следующее.

Открой окно LXTerminal. Проследи, чтобы программный файл был рабочим. Сделать это можно с помощью команды Linux `chmod` (англ. *change mode* – «изменить модуль»):

```
$ chmod +x /home/pi/Python-проекты/distance_ton.py
```

Аргумент +x способствует тому, чтобы все пользователи компьютеров получили права, x – для запуска программы.

Затем ты вносишь изменения в файл автоматического запуска программ */etc/rc.local*. В этом файле ты можешь писать команды, которые будут выполняться автоматически после запуска компьютера. Эти команды выполняются даже тогда, когда не запущен графический пользовательский интерфейс. Обозначение rc относится к английскому слову *run commands* (выполнение команд).

В файле */etc/rc.local* добавь одну строчку, которая запустит твою программу. Легче всего это сделать через редактор Nano. Важно, чтобы Nano ты вызвал под правами администратора:

```
$ sudo nano /etc/rc.local
```

Используй клавиши со стрелками ← и → для перемещения курсора в конец файла. Последняя строчка гласит:

```
exit 0
```

Перед этой строкой введи такую:

```
python3 /home/pi/Python-project/distance_ton.py
```

Текст выглядит как на рис. 14.12. Новая вставленная строка обведена.

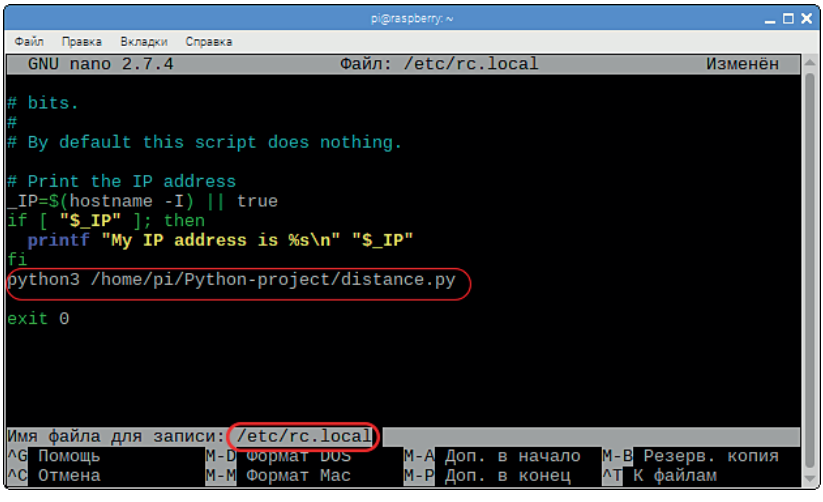


Рис. 14.12. *В новом текстовом файле /etc/rc.local добавляется новая строка*

Используя комбинацию клавиш **Ctrl**+**O**, сохрани изменённый файл. После нажатия этой комбинации клавиш Nano отобразит имя сохраняемого файла и будет ожидать подтверждения выполняемой операции (на рис. 14.12 имя файла обведено). Для подтверждения сохранения нажми клавишу ↵. С помощью комбинации клавиш **Ctrl**+**C** ты завершишь работу редактора Nano.

Теперь ты можешь завершить работу компьютера. Если ты его вновь подсоединишь к источнику питания, твоя Python-программа запустится автоматически.

Создаём собственный модуль

Функции измерения расстояния с помощью ультразвука можно использовать для нескольких проектов. Если тебе ультразвуковой модуль для измерения расстояния нужен, программу следует немного доработать. Для этого возьми ранее созданную программу *distance_timeout.py* и удали находящийся в конце программы цикл:

```
while True:
    a = calculate_distance()
      if p > 0:
print(I, 'Расстояние:' a, 'см')
```

После этого в начале программы добавь `Docstring` (строка документации), которая объяснит функции этого модуля. Далее помести вновь созданный модуль в папку со своими проектами и назови его именем, объясняющим, что этот модуль является Python-интерфейсом для ультразвукового модуля, например *us_python.py*.

Еще раз обращаем твоё внимание: строка документации на работу модуля не влияет. Она предназначена только для пояснения функций этого программного кода и содержит дополнительную информацию, например к каким контактам GPIO подключены контакты ультразвукового датчика (Pin 8 GPIO → Trigger; Pin 10 GPIO → Echo).

Ниже показан фрагмент программы со строкой документации, взятой в тройные верхние кавычки:

```
'''Имя файла: us_python.py
Python-модуль для измерения расстояний
с помощью ультразвука.
Pin 8 GPIO --> Trigger
Pin 10 GPIO --> Echo
Автор: Имя Фамилия
```

```
Последнее изменение: 29 июля 2014
'''
import time
...
# Главная программа
init_gpio()                    #1
```

Этот модуль или имена из этого модуля Python могут импортировать другие скрипты. Имя модуля – это имя файла без окончания *.py*, например `us_python`. Все инструкции в этом файле выполняются во время импорта модуля, в особенности #1, в котором вызывается функция `inti_gpio()`. В этом случае тебе не нужно набирать программный код всего импортируемого модуля. Сама программа, которая с помощью ультразвукового датчика измеряет расстояние до препятствия, уменьшается до нескольких строчек:

```
#1 distance_ton.py
from us_python import *                          #1
while True:
    pause = calculate_distance()/100
    os. system('aplay sounds/boing.wav')
    time. sleep(pause)
```

Звёздочка в конце строки #1 позволяет импортировать *все* функции модуля `us_python`.

Вопросы

1. Что такое необязательный (произвольный) аргумент?
2. Что нужно учитывать при передаче функции списка в качестве аргумента?
3. Какие преимущества имеет строка документации перед комментарием?
4. Сколько параметров с заданными по умолчанию значениями имеет функция `print()`? Введи в командную строку команду `help(print)`.
5. Для чего нужен делитель напряжения при подключении к Raspberry Pi ультразвукового датчика?
6. Из каких деталей состоит делитель напряжения?
7. Какова минимальная частота ультразвука?

Задания

Задание 1. Приветствие

Функция `welcome()` должна вернуть текст приветствия, который соответствует времени суток, например:

```
>>> welcome('Вася', 12)
'Добрый день, Вася!'
>>> welcome('Оксана', 8)
'Доброе утро, Оксана!'
```

Заголовок строки определения функции здесь:

```
def welcome(имя, время):
```

Первый параметр – это имя человека, и оно должно быть строкой. Второй параметр отображает время суток и при вызове функции вводится цифрами.

Задание 2. Расход калорий

Функция `calories()` должна высчитывать и возвращать количество энергии, потраченной во время катания на велосипеде. У неё два произвольных параметра. Первый – это *время*, и он обозначает длительность поездки в часах (предустановленное время 1 ч). А второй параметр *вес* обозначает вес тела в килограммах (по умолчанию 75 кг).

Количество израсходованных калорий (при нормальной поездке по ровной местности) рассчитываются по следующей формуле:

```
энергия = время × вес × 7,5.
```

Задание 3. «Бесконечный» режим сигнала SOS

В восьмой главе (проект 18) давалось описание программы, которая азбукой Морзе отправляла сигнал тревоги SOS. Сигнал передавался с помощью мигающих светодиодов. Измени эту программу. Она должна отправлять сигнал SOS в режиме бесконечного повторения. Для этого следует в программном тексте использовать функции.

Совет

Светодиод соединён с анодом (длинный вывод) через резистор (номиналом 130 Ом) к контакту 1 GPIO (3,3 В). Катод (короткий вывод) подключён к контакту 10. Используй вот эту программную структуру:

```
from RPi import GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.OUT)

def punkt():
    GPIO.output(10, False)
    time.sleep(0.2)
    GPIO.output(10, True)
    time.sleep(0.2)

def dash():
    ...

def pause():
    ...

def s():
    point()
    point()
    point()
    point()

def o():
...

# Главная программа
while True:
    ...
```

Ответы на вопросы

1. Произвольный (необязательный) аргумент во время вызова функции может быть пропущен.
2. Список – это изменяемый объект. Он меняется для каждой функции.
3. Строка документации в функции `f` отображается, если используется справочная система

```
>>> help(f)
```

4. Стандартная функция `print()` обладает четырьмя аргументами с предустановленными значениями:
   - `sep=' '`: в качестве Separatorstring, разделяющей два выходных значения и являющейся пробелом по умолчанию;
   - `end='\n'`: по умолчанию после инструкции `print()` на выходном потоке вводится разрыв строки;
   - `file=sys.stdout`: по умолчанию вывод выполняется в стандартный выход. Это Python Shell. Используя `print()`, можно написать ещё один файл. Затем при вызове через кодовое слово аргумента передаётся имя файлового объекта в формате `file=`*`fileobjekt`*;
   - `flush=False`: по умолчанию означает, что при написании в файл этот файл *не* запоминает данные в буфере после каждой операции записи на SD-карту. Однако если кодовое слово аргумента передаётся на `flush=True`, он сохраняется после каждой инструкции `print()`.
5. Выходное напряжение на контакте ультразвукового датчика равно 5 В. Это напряжение для разъёмов GPIO слишком большое и может детали материнской платы, подключённые к контактам GPIO, повредить. Делитель напряжения уменьшает его до 3,3 В.
6. Делитель напряжения состоит из двух резисторов.
7. Звуки с частотой, превышающей 20 кГц, не воспринимаются слухом. Это и есть ультразвук.

Ответы на задания

Решение 1. Приветствие

Пишем программу

```
def welcome(name, time):
    if time <11:       #1
        return 'Доброе утро, ' + name +'!'     #2
    if time <18:
        return 'Добрый день, ' + name +'!'
    else:
        return 'Добрый вечер, ' + name +'!'
```

Как это работает?

#1 Если второй аргумент является цифрой, которая меньше 11…

#2 … то возвращается текст, состоящий из трёх частей.

В первой части приветствия находится 'Доброе утро, ', затем следует содержимое первого аргумента, т. е. имя, которое ты введёшь при вызове функции. В конце строки приветствия ставится восклицательный знак. После инструкции return выполнение функции прекращается.

Запусти редактор IDLE 3, создай новый файл, введи программный текст функции и нажми клавишу **F5**. Введи команду вызова функции welcome('Вася', 10) и нажми клавишу ↵.

Решение 2. Расход калорий

Пишем программу

```
def calories(время=1, вес=75):
    энергия = время * вес * 7.5
    return энергия
```

В окне редактора IDLE 3 создай новый файл и введи приведённый выше программный текст функции. После нажми клавишу **F5**, введи команду вызова функции calories(время=1, вес=75) и нажми клавишу ↵. Обрати внимание: при вызове функции ты можешь ввести как один из аргументов, так и оба.

```
>>> калории()                  #1 час. 75 кг
562.5
>>> калории(2)                 #2 часа. 75 кг
1125.0
>>> калории(0.5. 65)           #1/2 часа. 65 кг
243.75
```

Решение 3. «Бесконечный» режим сигнала SOS

Пишем программу

```
from RPi import GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10. GPIO.OUT)

def point():                        #1
    GPIO.output(10, False)
    time.sleep(0.2)
    GPIO.output(10, True)
    time.sleep (0.2)
```

```
def dash():                          #2
    GPIO.output(10, False)
    time.sleep(0.6)
    GPIO.output(10, True)
    time.sleep(0.2)

def pause():                         #3
    time.sleep(0.4)

def s():                             #4
    point()
    point()
    point()
    pause()

def o():                             #5
    dash()
    dash()
    dash()
    pause()

#Главная программа
while True:                          #6
    s()
    o()
    s()
    pause()
```

Как это работает?

#1 Функция `point()` обеспечивает свечение светодиода на протяжении 0,2 с.

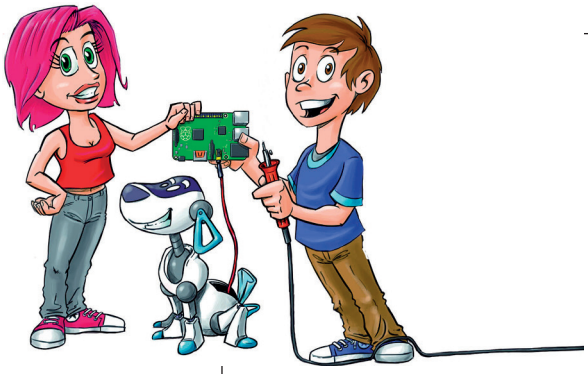
#2 Функция `dash()` обеспечивает свечение светодиода на протяжении 0,6 с.

#3 Функция `pause()` обеспечивает паузу в 0,4 с.

#4 Функция `s()` обеспечивает ввод буквы s. Она последовательно вызывает три функции `point()` и одну функцию `pause()`.

#5 Функция `o()` обеспечивает передачу буквы o.

#6 Главная программа очень краткая и доступна для понимания. Смысл её понятен сразу. В бесконечном цикле повторений буквы вводятся последовательно.



15 Измерение температуры и система «Умный дом»

В этой главе мы разработаем программы, которые с помощью цифрового температурного датчика (DS1820) отображают данные о температуре и анализируют их. Если у тебя есть аккумулятор для питания RPi, ты сможешь собрать температурные данные в своей местности и сохранить их или определить самые тёплые и самые прохладные места в помещении.

Кроме того, из этой главы ты узнаешь, как с помощью передатчика управлять электрическими беспроводными розетками. Используя эту технику, ты сможешь создать программы, которые будут автоматически включать и выключать домашние устройства.

Измерение температуры

Измерить температуру проще всего с помощью цифрового датчика прямого назначения, например моделью DS18B20 или DS18S20 фирмы «Даллас» (Dallas). Один экземпляр тебе обойдётся примерно в 150 руб. Температурный датчик встроен в микросхему этого устройства. Он измеряет тем-

пературу от –55 до +125 °C и отправляет цифровое сообщение по двунаправленной шине связи для низкоскоростных соединений (1-Wire-Bus). Данные передаются по проводу питания цифрового датчика. Цифровое сообщение состоит из двух частей:

- сведений об изменяемой температуре;
- уникальной строки символов (признака), которая идентифицирует датчик.

Благодаря уникальному признаку (идентификатору) к одной шине можно подключить несколько датчиков.

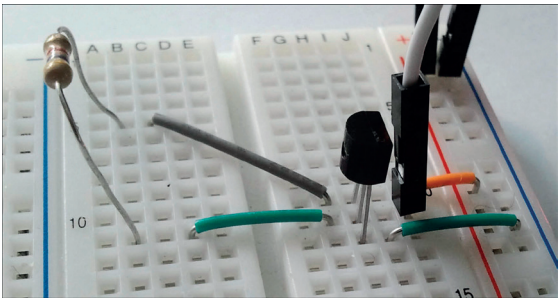


Рис. 15.1. *Датчик температуры DS18B20, расположенный на плате (столбец H)*

Для проекта тебе понадобятся следующие компоненты:

- макетная плата;
- резистор номиналом 4,7 кОм;
- короткие джампер-кабели (перемычки);
- три длинных провода с разъёмами «папа-мама»;
- термодатчик DS18B20 или DS18S20 (по цене от 150 руб. за штуку).

Принципиальная схема подключения термодатчика к разъёму GPIO материнской платы компьютера показана на рис. 15.2. В соответствии с этой схемой ты сможешь подключить любое количество датчиков. Но сейчас мы обойдёмся и одним устройством. На схеме датчик показан сверху. Его выводы вставляются в монтажную плату.

Таблица 15.1. Подключение к GPIO датчика температуры (модель DS18B20 или DS18S20)

DS1820	GPIO
Контакт 1	Контакт 6 (GND, корпус)
Контакт 2	Контакт 7 (1-Wire-Bus)
Контакт 3	Контакт 1 (3,3 В)

Между контактами 2 и 3 модели DS18B20 или DS18S20 установлен резистор в 4,7 кОм. Если ты допустишь ошибку и неправильно соединишь датчик с резистором и контактами GPIO, датчик температуры сначала нагреется, а затем через короткое время сгорит. Так что после проверки монтажа для безопасности при первом включении лучше контролировать нагрев датчика пальцем, и если корпус детали начнёт греться, его нужно быстро выключить (выдернуть из монтажной платы).

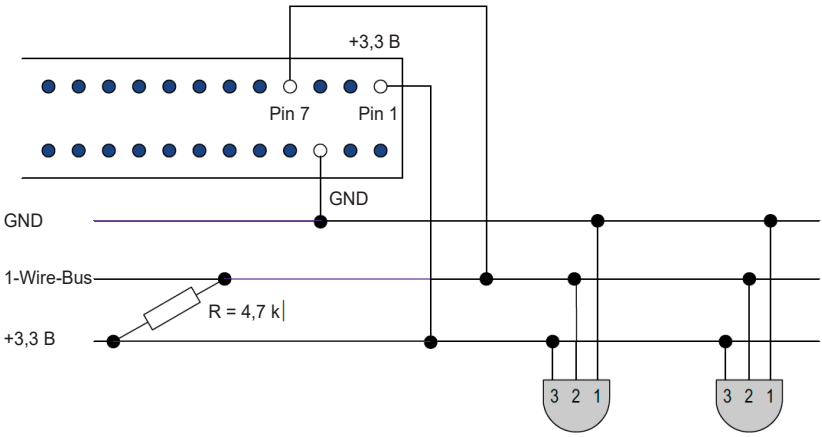


Рис. 15.2. *Принципиальная схема подключения термодатчика DS18B20 или DS18S20 к контактам разъёма GPIO. Возможно параллельное подключение нескольких датчиков*

Как читать температурные данные?

Итак, ты датчики подключил? Отлично. Далее нам нужно будет изменить конфигурацию файлов. В этом тебе поможет любой текстовый редактор. Главное, чтобы ты делал это с правами администратора. Мы будем использовать редактор Nano. Чтобы запустить его, введи в командную строку LXTerminal следующую команду:

```
$ sudo nano /boot/config.txt
```

Используя клавиши со стрелками [↑], [↓], [←] и [→], переместись в конец файла конфигурации и введи под последней строчкой следующее значение:

```
dtoverlay=wl-gpio
```

Чтобы сохранить, нажми комбинацию клавиш **Ctrl**+**O** и подтверди сохранение файла, нажав клавишу [↵]. Для за-

вершения работы редактора Nano нажми комбинацию клавиш **Ctrl**+**X**.

Перезагрузи RPi, чтобы изменение конфигурации вступило в силу:

```
$ reboot
```

Теперь введи в командной строке окна LXTerminal следующие команды, активирующие температурные датчики:

```
$ sudo modprobe w1-gpio
$ sudo modprobe w1-therm
```

Команда Linux `modprobe` добавляет операционной системе дополнительный модуль.

С этого момента данные этого маленького по габаритам сенсора будут сохраняться в виде текстового документа в специальном каталоге Raspberry Pi.

Воспользуйся файловым менеджером и открой папку, расположенную здесь: */sys/bus/w1/devices*.

Здесь для каждого подключённого цифрового датчика имеется свой собственный каталог. Обозначение датчика – это имя каталога. У модели DS18S20 оно начинается с цифры 10, а у моделей DS18B20 – с цифры 28.

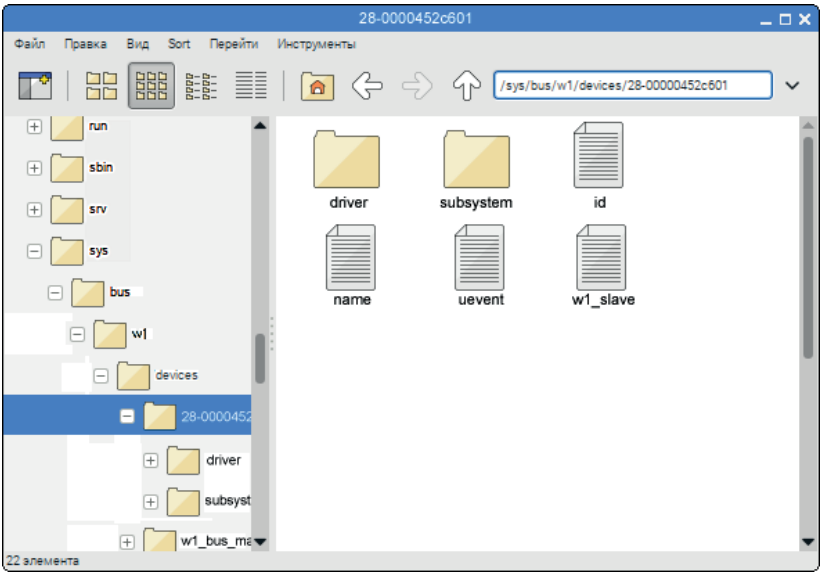


Рис. 15.3. *Это папка датчика с обозначением (именем) 28-00000452601. В ней содержится файл w1_slave с данными температуры*

Важно! Запиши себе имя каталога! Для твоего проекта тебе понадобится точное имя. В каталоге сенсора имеется текстовый документ с названием *w1_slave* (рис. 15.3). Открой его в редакторе Leafpad. В этом файле содержится двухстрочный текст, который может выглядеть примерно так:

```
6f 01 4b 46 7f ff 01 10 67 : crc=67 YES
6f 01 4b 46 7f ff 01 10 67 : t=22937
```

Первая строка начинается с передаваемых необработанных данных. Они включают в себя контрольную сумму (здесь это число 67), которая позволяет убедиться, что данные не были искажены. Запомни: к 1-Wire-Bus можно подключить несколько датчиков. Для передачи данных они используют провод питания датчика, поэтому данные могут быть повреждены. В конце первой строки стоит информация о том, верно ли прошла передача температурных данных. Завершает же строку слово YES, оно уведомляет о том, что всё в порядке.

Вторая строка также начинается с необработанных данных. В конце (после знака равенства) значение температуры указывается с тремя знаками после запятой (в тысячных долях градуса Цельсия). В этом примере мы имеем температуру 22,937 °C. К сожалению, точность показания датчика (допуск) составляет не 0,001 °C, а около 0,5 °C.

Проект 40. Делаем замеры температурных изменений

Программа каждую секунду должна считывать данные термодатчика и выводить их на экран. Чтобы проверить работу датчика, прикоснись к нему пальцем и некоторое время подержи палец. Изменение температуры датчика ты увидишь на экране.

Вывод (пример)

```
0 22.937 градуса Цельсия
1 23.312 градуса Цельсия
2 23.312 градуса Цельсия
...
```

Запусти IDLE3 в окне LXTerminal на правах администратора (режим root):

```
$ sudo idle3
```

Потом эту программу ты можешь запустить и протестировать прямо из окна редактора, нажав клавишу **F5**.

Функция os.system()

Используя функцию `os.system()` из модуля **os**, ты можешь команды Linux запускать из программы Python. Например: программная строка в программе Python

```
os.system('modprobe w1-gpio')
```

выполнит то же действие, что и команда

```
$ modprobe w1-gpio
```

вводимая из командной строки в консоли (LXTerminal).

Аргумент `os.system()` всегда является строкой символов. То есть если ты используешь команду Linux в своей программе, её нужно вводить как строку символов функции `os.system()`. Всегда помни о том, что команда Linux должна заключаться в кавычки.

Этот метод мы используем в следующей программе.

Пишем программу

```
# temp_log.py
import os, time
os.system('modprobe w1-gpio')                #1
os.system('modprobe w1-therm')

PATH = '/sys/bus/w1/devices/xxx/w1_slave'    #2

def readTemp():                              #3
    ok = False                               #4
    while not ok:                            #5
        f = open(PATH, 'r')                  #6
        line_1 = f.readline()                #7
        line_2 = f.readline()                #8
        f.close()                            #9
        if 'YES' in line_1:                  #10
            ok = True
    part_1, part_2 = line_2.split('=')       #11
    return int(часть_2)/1000                 #12

# Главная программа
```

```
i = 0
while True:
    print(i,readTemp(),'Градусов по Цельсию')
    i + l
    time.sleep(1)
```

Как это работает?

#1 Выполняются команды для инициализации 1-Wire-Bus.

#2 Эта константа содержит путь к файлу с данными температуры. Вместо xxx ты вписываешь обозначение датчика, которое записал во время тестирования (например, 28-00000452c601).

#3 Здесь определяется новая функция readTemp(). У неё нет параметров.

#4 Переменная ok имеет истинное значение (True или False). Здесь сначала устанавливается значение False. Позже, если передача данных термодатчика пройдёт успешно, переменная получит значение True.

#5 Пока переменная ok имеет значение False, необходимо выполнить вложенный блок инструкций (внутренняя часть цикла). Можно было бы написать while == False, но строка while not ok более читабельна. Или ты не согласен?

#6 В этой командной строке открывается для чтения файл с текущими данными о температуре, и эти данные присваиваются переменной f. Буква f часто используется в качестве имени переменной для файлов, потому что на английском языке слово «файл» – file. Переменная f – это объект, который позволяет считывать содержимое файла.

#7 Эта команда считывает первую строку текста. Вызов метода f.readline() преобразует знаки прочитанной строки в строку символов и сохраняет её в переменной line_1.

#8 Далее читается вторая строка текстового файла и сохраняется в переменной line_2.

#9 Здесь файл вновь закрывается.

#10 Если в первой строке появляется YES, значит, передача файла прошла успешно и переменная ok установлена в значение True. Тогда инструкция while завершается. Если же нет, то необходимо повторить процесс чтения. Это значит, что во время передачи данных от температурного датчика произошёл сбой. Нам придётся читать файл до тех пор, пока в конце первой строки не появится слово YES.

#11 Вторая строка делится на две части, а знак равенства «=» используется как символ разделения. Переменная `part_1` получает текст перед знаком равенства (он нам вообще не нужен), а переменная `part_2` получает текст после знака равенства, т. е. температуру в тысячных градуса после запятой (рис. 15.4).

#12 Из строки `part_2` сначала получается *целое число*. Далее это число делится на 1000. Это температура в градусах Цельсия. И это число возвращается.

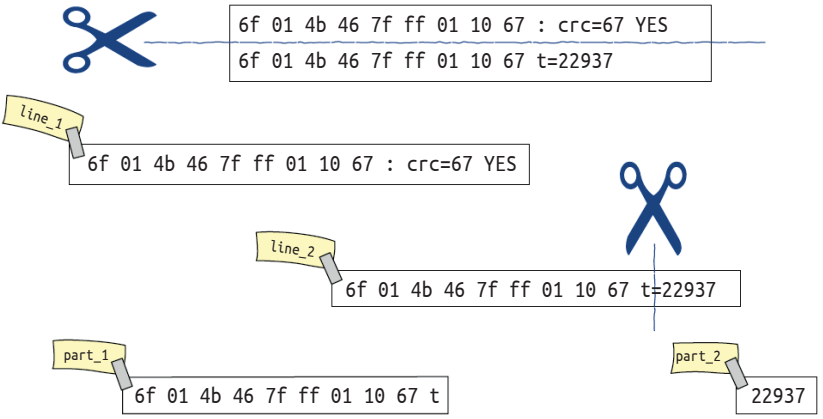


Рис. 15.4. *Разделяя строки, вы получаете число для измеренной температуры в 1000 °C*

Проект 41. Сохранение данных в формате CSV

Известно ли тебе, мой друг, что такое табличные вычисления? Это в высшей степени удобное программное обеспечение, с помощью которого можно проводить расчёты и отображать изменения данных в виде графиков или диаграмм. Наиболее популярные табличные редакторы – это Excel (Microsoft), OpenOffice Calc LibreOfice Calc. Результаты своих измерений ты можешь сохранять в определённом формате CSV. Такой файл можно импортировать из табличного редактора. Аббревиатура CSV означает *comma separated values* (значения, разделённые запятой). Следующий пример наглядно продемонстрирует строение файла CSV. Каждая *текстовая строка* содержит значения строки *таблицы* и разделяется точкой с запятой, а не просто запятой. Обозначение CSV сюда не совсем подходит. Точка с запятой (вместо запятой) используется в качестве разделительно-

го символа, так как сами цифры могут содержать запятые. О чём говорят таблицы? В момент времени 0 с температура составляет 20 °C, через 1 с она по-прежнему 20 °C и т. д.

```
Секунды ; температура в °C
0 ; 20
1 ; 20
2 ; 21
3 ; 21
4 ; 22
5 ; 21
6 ; 20
7 ; 20
```

Пишем программу

Программу *temp_log.py* из раздела «Проект 40. Измеряем температурные данные» следует в нескольких местах изменить. Значения на экран не выводятся, а данные записываются в CSV-файл, который выглядит как в данном примере:

```
# Главная программа
i = 0
f = open('data.csv', 'w')                                  #1
print('Секунды; температура в °C', file=f, flush=True)     #2
while True:
    temp = round(readTemp())
    print(i, ';', temp, file=f, flush=True)                #3
    i +=1
    time.sleep(1)
```

Как это работает?

#1 В этой командной строке с помощью функции `open()` открывается файл с именем *data.csv* (первый параметр) для записи (второй параметр `'w'`). Если этого файла в папке проектов нет, его следует создать. Открытый файл присваивается переменной `f`.

#2 Текст `'Секунды: температура в °C'` записывается только в файл `f`. Аргумент кодового слова `flush=True` позволяет сохранить текущее содержимое файла на периферийном носителе (SD-карте).

#3 В файле CSV записывается новая строка, которая состоит из трёх частей: количества секунд `i`, точки с запятой «;» и значения измеряемой температуры `temp`. Это значение записывается в виде целого числа.

Файл CSV сохраняется в папке проектов. Ты можешь его импортировать и обработать с помощью LibreOffice Calc.

Импортировать из этой программы можно так.

LibreOffice – это коллекция стандартных программ, которые можно использовать для работы в офисе или в школе. Здесь и обработка текста, и презентации, и работа с изображениями, и т. д. Мы будем использовать программу табличных вычислений Calc.

- Нажми кнопку основного меню в верхнем левом углу экрана. На экране появится основное меню.
- Щёлкни мышью на строке Офис (Office), а затем в появившемся подменю щёлкни мышью на строке LibreOffice Calc.
- Выбери команду меню **Файл** ⇨ **Открыть** (File ⇨ Open) и открой папку с данными о температуре.

Файл находится в той папке проектов, где сохранена сама программа. Затем появится окно с функциями импортирования (рис. 15.5).

Помни о том, что в качестве разделительного символа используется лишь точка с запятой (а не только пробел).

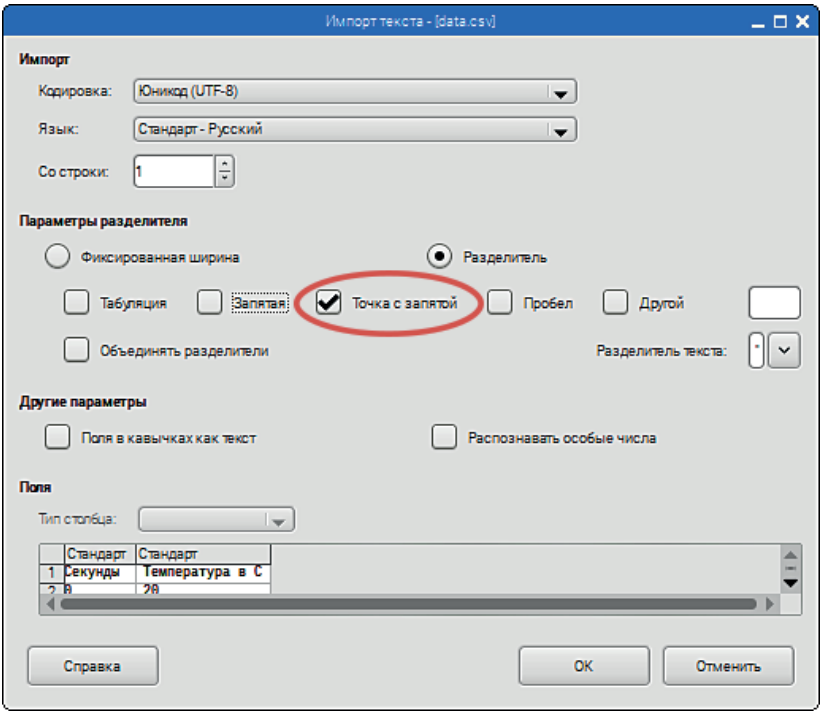


Рис. 15.5. *Импортируем файл CSV, используя LibreOffice на твоём RPi. Единственным разделительным символом нам служит точка с запятой*

Если всё получилось так, как надо, ты получишь таблицу (рис. 15.6 вверху слева).

- Выдели всю таблицу, выбери команду меню **Вставка** ⇨ **Диаграмма** (Insert ⇨ Diagramm). На экране появится диалог **Мастер диаграмм** (Chart wizard) (рис. 15.6 середина).
- В списке диаграмм **Выберите тип диаграммы** (Select the type of chart) выбери тип диаграмм **Линии** (Lines) и в списке диаграмм справа щёлкни мышью на типе диаграммы **Линии и точки** (Lines and Points).
- Нажми кнопку **Готово** (Done). Диаграмма построена (рис. 15.6 внизу).

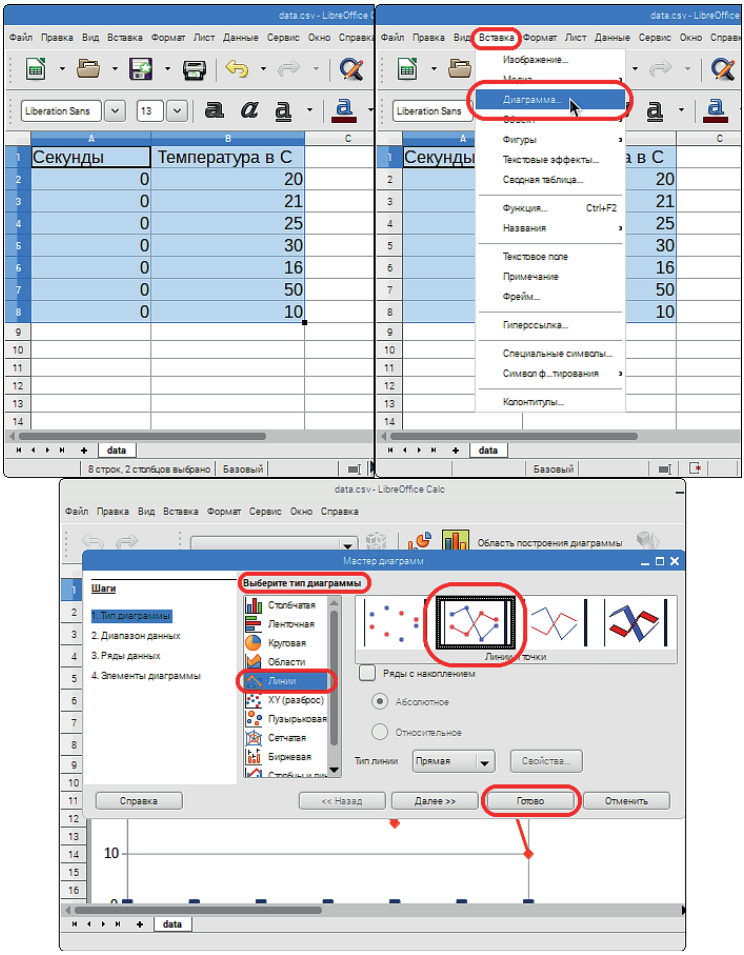


***Рис. 15.6** (начало). Последовательность построения диаграммы температур в программе OpenOffice/LibreOfice*

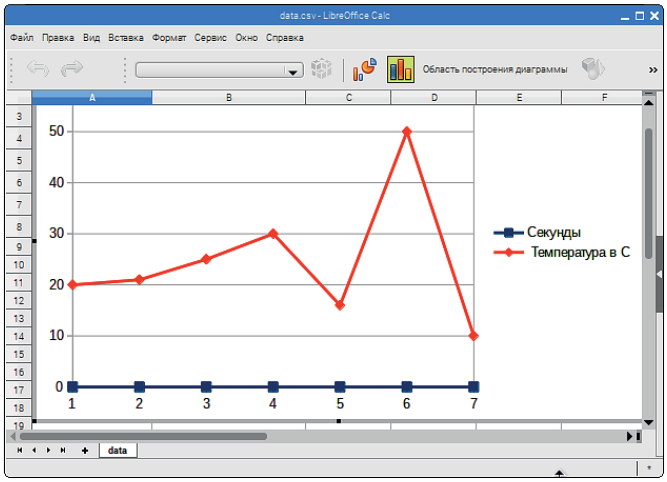


Рис. 15.6 *(окончание)*

Предложения для экспериментов

Охлаждение при испарении

Помести небольшой колпачок из фольги на термодатчик, как показано на рис. 15.7. Для этого подойдёт маленький кусочек абсорбирующей бумаги. Капни туда капельку воды и измерь температурный режим (диаграмма температуры и времени). Наблюдение: температура падает, т. к. вода при испарении окружающей среды отводит тепло (термин «энтальпия испарения»). Повтори эту попытку с алкоголем и средством для снятия лака.

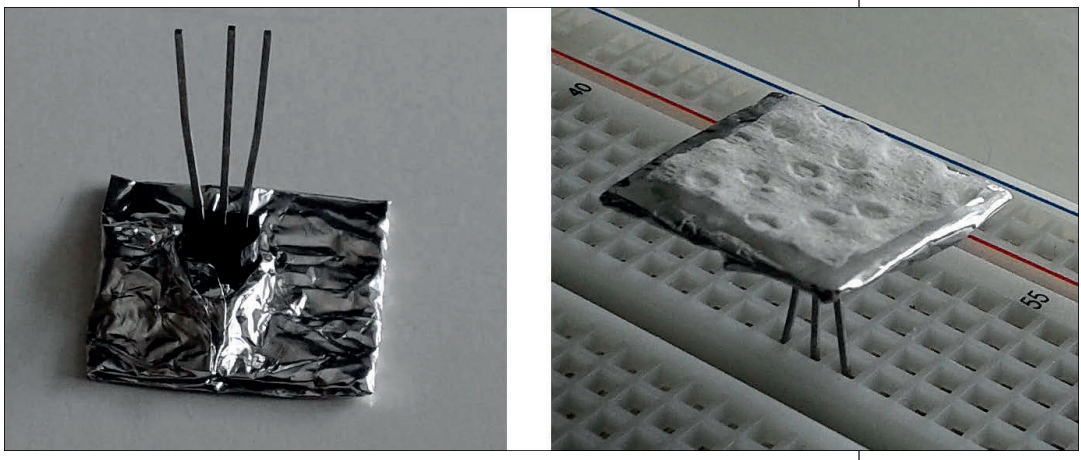


Рис. 15.7. *Термодатчик модели DS18S20 со шляпкой из фольги. Справа на шляпке расположен кусочек абсорбирующей бумаги, который можно смочить небольшим количеством жидкости*

Запись профиля температуры

Возьми мобильное аппаратное устройство с аккумулятором, которое мы сделали в главе 14 (проект 39), и позаботься о том, чтобы программа из раздела «Проект 41. Сохранение данных в формате CSV» (сохранение температурных данных в виде CSV-файла) запускалась автоматически при включении Raspberry Pi. Добавь туда также кнопку выключения, которая при нажатии будет завершать работу Raspberry Pi.

Медленно и равномерно передвигайся с этим мобильным устройством по местности, включающей в себя самое разное покрытие (каменная поверхность, трава и т. д.). С помощью табличного вычисления определи данные и составь температурный профиль.

Измерение температуры в воздушных потоках

В разных местах помещения установи датчики температуры. Все они могут быть подключены к одному и тому же кабелю. Измерь распределение температуры при различных формах воздушного потока.

Как управлять беспроводной розеткой?

Менее чем за две тысячи рублей ты получишь комплект беспроводных розеток с пультом дистанционного управления. Они уже протестированы и абсолютно безопасны. Ты к этим розеткам можешь подключить мощные бытовые приборы, такие как лампа или вентилятор, и вполне безопасно управлять ими на расстоянии. При покупке обрати внимание на то, чтобы они управлялись на радиочастоте 433,92 МГц. Для этого подойдут приборы фирмы Elro или Mumbi.

Подготовка беспроводной розетки

Беспроводная розетка (к примеру, фирмы Elro или Mumbi) обычно имеет на обратной стороне узкую крышку, закрытую при помощи винта. Открой её. За ней спрятано 10 ДИП-переключателей (маленькие «мышиные» клавиши). С ними ты можешь настраивать системные номера и Unit-номера. Оба номера тебе понадобятся позже, чтобы включить и выключить розетку с помощью Raspberry Pi.

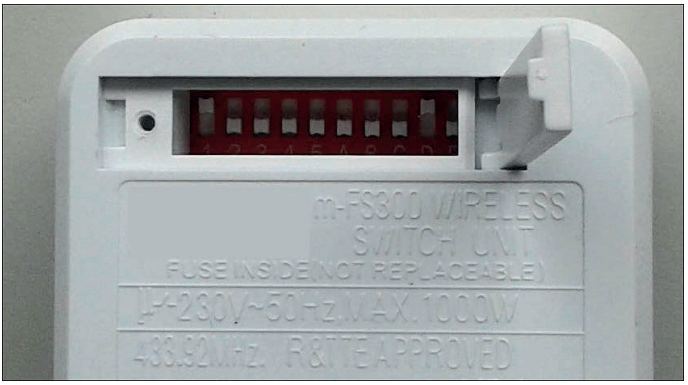


Рис. 15.8. *Так выглядит ДИП-переключатель в беспроводной розетке*

Первые 5 переключателей (левая половина) представляют собой системный номер. Эта цифра высчитывается следующим образом:

```
N = 0
Если рычажок первого переключателя поднят вверх, добавь 1.
Если рычажок второго переключателя поднят вверх, добавь 2.
Если рычажок третьего переключателя поднят вверх, добавь 3.
Если рычажок четвёртого переключателя поднят вверх, добавь 8.
Если рычажок пятого переключателя поднят вверх, добавь 16.
```

Например, если рычажки первого и второго ДИП-переключателей подняты вверх, значит, системный номер будет 3.

Следующие 5 ДИП-переключателей определяют номер устройства. Номер устройства определяется по такому же принципу. Этот номер используется для идентификации розетки. Если ты хочешь управлять несколькими розетками отдельно друг от друга, каждой розетке потребуется отдельный номер, по которому эта розетка будет распознаваться компьютером. Этот номер называется Unit-номером.

Какие номера присвоены розетке, показанной на рис. 15.8? Системный номер 1, Unit-номер 8. В этом разделе автор использовал беспроводные розетки Elro и Mumbi. Приборы других производителей работают точно так же.

Подключение передатчика

Мы используем передатчик (трансмиттер) модели WRL-10534 RF-Link. Он передаёт управляющий сигнал (радиоволну) на частоте 433,92 МГц. Один мегагерц (МГц) – это 1 000 000 колебаний в секунду. Через радиоволны команды управления передаются на беспроводную розетку. Такая

команда содержит адрес розетки, которой нужно управлять, и команду **On** (Включить) или **Of** (Выключить). Адрес состоит из системного номера и Unit-номера. Но об этом позже. Каждая беспроводная розетка принимает отправленные команды и проверяет адрес. Если команда верна, то розетка реагирует на неё и в зависимости от полученной команды включает или выключает подключённую к ней нагрузку.

Маленькая плата с передатчиком имеет четыре порта. Ты можешь подключить их к плате, как показано на рис. 15.9.

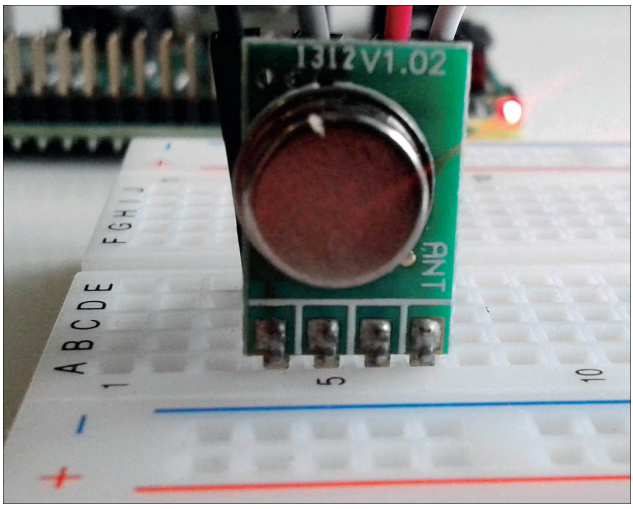


Рис. 15.9. *Передатчик (трансмиттер) мощностью 434 МГц (модель WRL-10534), установленный на плате*

Ориентируйся на маркировку ANT с правой стороны платы. Таблица 15.2 наглядно демонстрирует, как можно подключить передатчик с GPIO (слева направо).

Таблица 15.2. Подключение передатчика WRL-10534 (слева направо)

Передатчик	GPIO
GND	Контакт 6 (GND)
DATA	Контакт 11
VCC	Контакт 2 (5 В)
ANT	Антенна, кабель около 17 см длиной

Установка Pilight

Pilight – это система с открытым кодом для системы «Умный дом». Используя команду `pilight-send`, ты сможешь со

своего Raspberry Pi управлять беспроводной розеткой через передатчик, работающий на частоте 434 МГц. Однако для этого тебе понадобится установить *Pilight*.

До этого момента для установки мы всегда использовали команду `sudo`, которая позволяла выполнять команду на правах системного администратора.

Но для установки *Pilight* было бы удобнее получать права администратора с помощью команды `su`:

```
pi@raspberry ~ $ sudo su
root@raspberry: /home/pi#
```

Теперь приглашение командной строки Promt изменено с `pi@raspberry ~ $` на `root@raspberry: /home/pi#`. Здесь слово «root» показывает, что ты сейчас находишься в системе с правами администратора. Теперь у тебя больше прав, чем у обычного пользователя `pi`. И ты можешь выполнять следующие команды.

Обнови список пакетов `apt-get`:

```
# apt-get update
```

Открой с помощью редактора Nano (в лице администратора) файл */etc/apt/sources.list*:

```
# nano /etc/apt/sources.list
```

В этом файле указано, из каких источников пакетное управление получает новые программы и обновления.

Добавь в самом низу следующую строку:

```
deb http://apt.pilight.org/ stable main
```

Обрати внимание на пробел. Сохрани расширенный файл. Для этого нажми комбинацию клавиш **Ctrl**+**O** и клавишу ↵. Заверши работу редактора Nano, нажав комбинацию клавиш **Ctrl**+**X**.

Пакеты *Pilight* защищены подписью. Прежде чем устанавливать *Pilight*, необходимо загрузить ключ.

Для загрузки ключа введи в строке следующую команду:

```
# wget -O - http://apt.pilight.org/pilight.key | apt-key add -
```

Следи за тем, чтобы после первого знака минус стояла большая буква О (это не ноль!).

Обнови пакеты `apt-get`:

```
# apt-get update
```

Установи *Pilight*:

```
# apt-get install pilight
```

Pilight – это служба, которая постоянно работает в фоновом режиме. При следующем старте системы *Pilight* запустится автоматически. Ты же можешь это сделать вручную:

```
# service pilight start
```

Важно, чтобы перед использованием `send-pilight` был запущен *Pilight*-сервер. Все команды, адресованные беспроводной розетке, идут через *Pilight*-сервер.

Теперь в роли администратора ты можешь снова выйти из программы:

```
# exit
```

И ты снова увидишь привычный Promt.

Управление беспроводной розеткой с помощью Pilight: pilight-send

Если трансмиттер подключён, ты можешь управлять розеткой с помощью команды `pilight-send`. Если команда отправлена, то цифровая команда через радиоволны частотой 433,92 МГц направляется сразу на несколько розеток. Чтобы эта команда была понятной, её нужно построить согласно определённым правилам. Эти правила называются *протоколом*. И это тебе тоже известно из обычной жизни. Например, для писем есть такой протокол: вверху находятся имя отправителя и имя адресата. Затем идёт обращение, основной текст и, наконец, форма прощания и подпись. Очень важно придерживаться такого протокола. Если, скажем, адрес стоит в другом месте, возможно, письмо вообще не будет отправлено.

Для различных беспроводных розеток тоже существуют различные протоколы связи. *Pilight* поддерживает прото-

колы любых производителей. Для тебя это сейчас немного детективная работа. Тебе нужно разузнать, какой протокол используется для твоей розетки и как можно команду `pilight-send` использовать с этим протоколом. Действуй так.

Прежде всего узнай, какие вообще протоколы *Pilight* поддерживаются. Вызови справку `pilight-send`:

```
$ sudo pilight-send -H
```

Ты получишь текст с информацией, которая содержит длинный список обозначений протоколов.

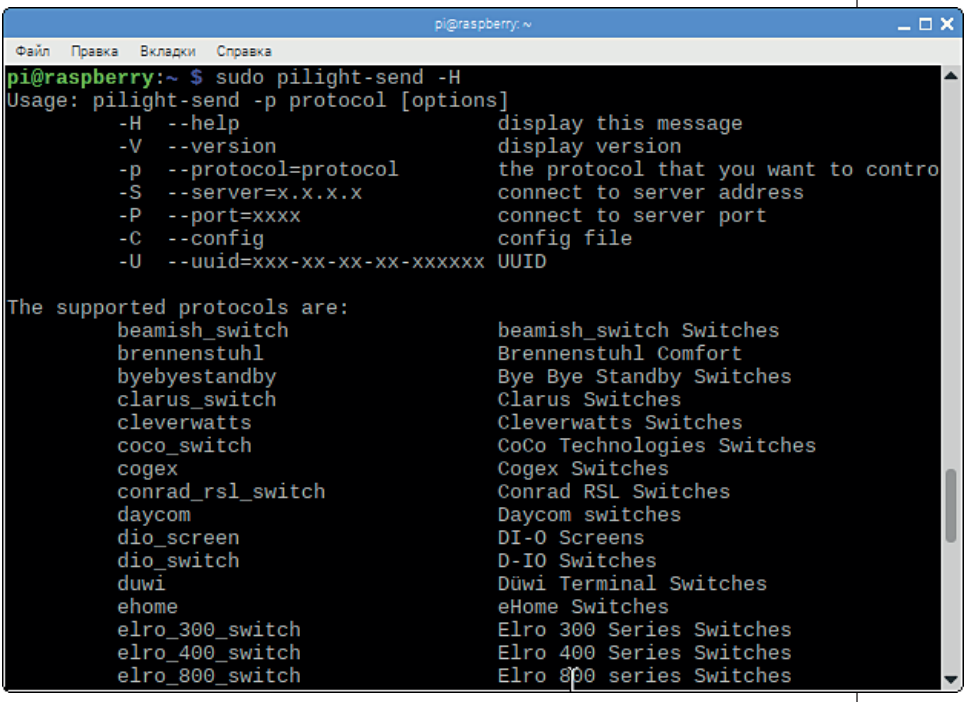


Рис. 15.10. *Вызов справочной информации для `pilight-send`*

В этом списке ты найдёшь (я надеюсь) название своей беспроводной розетки. Если же её там нет, то поищи в интернете. Существуют типы розеток, которые продаются под разными марками. Розетки фирм Mumbi по строению сходны с розетками Elro. В крайнем случае попробуй поочерёдно разные протоколы.

Предположим, что ты нашёл протокол, который подходит твоей розетке. Допустим, это `mumbi`. Теперь ты должен выяснить, какие аргументы командной строки указывались до этого протокола:

```
$ sudo pilight-send -p mumbi -H
```

Ты получишь подробную инструкцию (на английском), как использовать команду `pilight-send`.

Аргументы командной строки Linux

Аргументы командной строки (опции) существуют в краткой и длинной форме. Краткая форма начинается с простого знака минус и состоит из одной-единственной буквы, после которой часто может быть записано значение, например:

```
$ sudo pilight-send -p mumbi -H
```

Длинная форма аргумента начинается с двойного знака минус и одного слова, которое описывает, о чём идёт речь. Если данные нужно перенести, аргумент имеет форму `--`*код*`=`*значение*, например:

```
$ sudo pilight-send --protocol=mumbi -help
```

Следующие примеры относятся к Mumbi-протоколу.

У нас есть системный номер 1 и Unit-номер 8. Если ты используешь другие номера, то должен подобрать соответствующие аргументы.

Включение розетки

```
$ sudo pilight-send -p mumbi -s 1 -u 8 -t
```

После буквы `-s` идёт системный номер, а после буквы `-u` – Unit-номер. Аргумент `-t` означает *включить*. Если всё сделано правильно, то ты услышишь в розетке тихий щелчок реле, а контрольный светодиод начнёт мигать.

Выключение розетки

```
$ sudo pilight-send -p mumbi -s 1 -u 8 -f
```

И снова мы услышим щелчок реле, а светодиод выключится. Это происходит только в случае, если всё работает. Но, возможно, розетка не реагирует на команды, и ты получишь какое-нибудь оповещение об ошибке:

```
ERROR: no pilight ssdp connection found
```

В этом случае, скорее всего, поможет остановка и перезагрузка сервиса *Pilight*:

```
$ sudo service pilight stop
```

Остановка программы займёт некоторое время. Затем можно перезагрузить её:

```
$ sudo service pilight start
```

Проект 42. Отправляем секретные команды ночью

На окне стоит светильник, он подключён к беспроводной розетке. За окном ночь. В определённое время светильник начинает мигать. Он отправляет сообщение с помощью азбуки Морзе. Команда повторяется трижды, затем снова становится темно.

Программа, управляющая лампой, является интерактивной. Сначала пользователь вводит данные для команды: текст, количество повторений и время запуска. Затем берёт на себя управление розеткой, которая в определённый момент будет включать и выключать лампу. Кроме того, программа содержит инструкции `print()` и выдаёт в Shell отправляемые знаки азбуки Морзе в виде точки и тире.

Процесс программы (пример)

```
Что мне нужно отправить?
Текст: утром в девять
Сколько раз нужно отправить?
Количество: 2
Через сколько минут отправить?
Минуты: 0.1
-- --- .-. --. . -. ..- -- -. . ..- -.
-- --- .-. --. . -. ..- -- -. . ..- -.
Сообщение отправлено 2 раза.
```

Пишем программу

```
# morse.py
from time import time, sleep
from os import system
DIT = 1                                                    #1
TO='sudo pilight-send -p mumbi -s 1 -u 8 -t'               #2
FROM = 'sudo pilight-send -p mumbi -s 1 -u 8 -f'
                                                           #3
MORSECODE = {'a':'.-', 'b':'-...', 'c':'-.-.',
```

```
    'd':'-..', 'e':'.', 'f':'..-.', 'g':'--.',
    'h':'....', 'i':'..', 'j':'.---', 'k':'-.-',
    'l':'.-..', 'm':'--', 'n':'-.', 'o':'---',
    'p':'.--.', 'q':'--.-', 'r':'.-.', 's':'...',
    't':'-', 'u':'..-', 'v':'...-', 'w':'.--',
    'x':'-..-', 'y':'-.--', 'z':'--..'}

def morse(letter):
    for a in MORSECODE[buchstabe]:
#4
        system(TO)                                          #5
        if a == '.':                                        #6
            sleep(DIT)
        else:
            sleep(3*DIT)                                    #7
        system(FROM)                                        #8
        sleep(DIT)
        print(a, end='')                                    #9
    sleep(3*DIT)                                            #10
    print(' ', end='')                                      #11

def sende(text):
    for letter in text:
         if letter in MORSECODE.keys():                     #12
             morse(letter)
         else:
             sleep(7*DIT)                                   #13
             print(' ', end='')
    print()                                                 #14

# Input
print(,Что мне нужно отправить')
text = input('Text: ').lower()                              #15
print(,Сколько раз нужно отправить?')
number = int(input(,Количество: '))                         #16
print(,Через сколько минут отправить?')
minute = float(input('Минуты: '))                           #17

# Processing
sleep(60*minuten)                                           #18
for i in range(number):
    sende(text)                                             #19

# Issue
print(,Сообщение отправлено %i раз' %(number))              #20
```

Как это работает?

#1 Константа DIT содержит время краткого сигнала Морзе. Радисты называют такой краткий сигнал Dit, а

длинный – Dah. Поскольку беспроводная розетка реагирует довольно вяло, мы должны эти сигналы отправлять не спеша. Короткий сигнал Dit длится всего одну секунду.

#2 Константы `TO` и `FROM` содержат команды для включения и выключения розетки. Если ты используешь другие системные и Unit-номера, тебе нужно ввести соответствующие аргументы `-s` и `-u`.

#3 `MORSEKODE` (азбука Морзе) – это словарь. Он содержит подходящий знак для каждой буквы в виде последовательности точек (Dit) и тире (Dah).

#4 Мы проходим через последовательность точек и тире, относящихся к буквам азбуки Морзе. То есть управляющая переменная `a` является либо точкой, либо тире.

#5 Розетка включается.

#6 Время передачи точки (Dit) короткое.

#7 Время передачи тире (Dah) в три раза дольше, чем время передачи точки. Это соответствует правилам радиосвязи.

#8 Лампа выключается.

#9 Для контроля отображается точка или тире. Кодовое слово аргумента `end=''` не разбивает строку. Следующий знак пишется за ним.

#10 После того как все точки и тире (Dit и Dah) отправлены, возникает пауза.

#11 Здесь отображается пробел, чтобы отделить переданную с помощью азбуки Морзе букву от буквы, которая будет передаваться следующей.

#12 Метод `keys()` возвращает коллекцию со всеми ключами словаря. Если буква присутствует в этой коллекции ключей, он её распознаёт и может отправить…

#13 … в противном случае символ будет рассматриваться как пробел, и возникнет пауза.

#14 Если весь текст отображается как последовательность знаков азбуки Морзе, выполняется перенос строки.

#15 Введённый с помощью `lower()` текст преобразовывается в последовательность из маленьких букв, поскольку в словаре ключей есть только маленькие буквы.

#16 То, что вводится с помощью `input()`, всегда является строкой символов. Но проводить вычисления со строкой символов нельзя, поэтому она с помощью функции `int()` должна преобразоваться в целое число.

#17 Текст с количеством минут, вводимый с помощью `float()`, преобразовывается в число с плавающей точкой.

#18 Функция `sleep()` в виде аргумента содержит количество секунд. Теперь программа ожидает рассчитанное количество секунд, после чего текст отправляется заданное количество раз.

#19 Текст передаётся с помощью знаков азбуки Морзе. Для этого розетка включается и выключается.

#20 При вводе с помощью инструкции `print()` в последней строке формат строки символов меняется. Замещающий символ `%i` отображает число, сохранённое в переменной `number`.

Другие проекты

Можно использовать Raspberry Pi для опроса датчиков и управления устройствами. Используя методы программирования, представленные в этой книге, ты можешь реализовать много проектов. Возможно, у тебя уже есть идеи, которые ты бы хотел воплотить. Вот ещё несколько предложений.

- Кондиционер для террариума. Используя вентилятор, электрическую лампу накаливания как источник тепла и термодатчик (см. раздел «Измерение температуры»), ты можешь разработать программу для регулирования температуры в террариуме. В случае, если температура в террариуме будет понижаться, автоматически включится лампа накаливания и подогреет воздух. Если температура поднялась до критического значения – включается вентилятор.
- Сценическое освещение. Для театральной или музыкальной постановки ты можешь запрограммировать последовательность включения прожекторов, которые создают световые эффекты.
- Управление через интерактивный сайт. Raspberry Pi может быть использован как HTML-сервер. На сайте есть кнопки и флажки, управляя которыми (например, с мобильного устройства), ты можешь включать и выключать устройства. Более подробно об этом ты прочитаешь в главе 16.

Вопросы

1. Какую задачу выполняет контрольная сумма в данных, которые отправляет DS18S20 на Raspberry Pi?
2. Какой смысл имеет уникальное обозначение (ID) у термодатчика?
3. Какие из последующих волн не являются электромагнитными? Свет, рентгеновское излучение, радиоволны, звуковые волны?
4. Сколько колебаний в секунду в одном килогерце и в одном мегагерце?
5. Как беспроводная розетка определяет, что отправленная по радио команда предназначена именно для неё?
6. На беспроводной розетке рычажки пяти ДИП-переключателей (слева направо) установлены в следующие положения: вверх, вверх, вниз, вверх, вниз. Какой системный номер получится?
7. Как называется свод правил между передатчиком и приёмником?

Задания

Задание 1. Игра «Горячо – холодно»

Разработай систему управления для робота, который занимается поиском источников тепла в квартире. Источником тепла может быть включённая электроплита или обогреватель. Мобильное строение корпусной части состоит из следующих частей:

- Raspberry Pi с мобильной системой управления;
- макетная плата;
- датчик температуры;
- светодиод;
- мини-переключатель;
- два резистора (номиналом 130 Ом и 4,7 кОм).

Программа запускается автоматически при включении Raspberry Pi. Как только программа заработает, светодиод трижды мигнёт. Потом, если измеряемая температура повышается, он начинает постоянно светиться. При понише-

нии температуры светодиод гаснет. При нажатии на кнопку выключения Raspberry Pi завершает свою работу и может быть отключен от источника питания.

Задание 2. Незаконное проникновение!

Этот проект является своего рода интеллектуальным датчиком движения для предотвращения нежелательного проникновения. Кто-то входит в комнату и приближается к Raspberry Pi. Это движение запускает целую цепь событий, например сразу же гаснет свет, а через шесть секунд он вновь включается. Одновременно с этим включается вентилятор, а через десять секунд выключается.

Советы по подготовке к проекту

Для работы тебе понадобится передатчик (трансмиттер), работающий с беспроводными розетками на частоте в 434 МГц, и ультразвуковой модуль (см. главу 14). Подключи оба устройства так, как показано в разделе 15.2 и описано в разделе 14.2. Скопируй модуль *us_python.py* из главы 14 в свою папку проектов. В модуле находится функция `calculate_distance()`.

Советы по программированию

Цепь событий – это текстовый файл *events.txt*. Он должен быть сохранён в той же папке, что и программа. Каждая строчка содержит три задачи, разделённые пробелом:

- Unit-номер беспроводной розетки, которую необходимо включить;
- время ожидания в секундах с момента последнего события;
- команды `-t` (включить) и `-f` (выключить).

Пример. Предположим, настольная лампа подключена к беспроводной розетке с Unit-номером 8. Вентилятор относится к беспроводной розетке с Unit-номером 4.

Тогда следующий текстовый файл отправляет последовательность событий, описанную в задаче:

```
8 0 -t
8 6 -f
4 0 -t
4 10 -f
```

Используй следующие инструкции Python:

```
f = open('events.txt', 'r')
    line = f.readline()
    while line:
       nr, seconds, command = line.split()
```

Здесь текстовый файл с последовательностью событий открывается и читается построчно. Каждая строка разделяется на три части.

Ответы на вопросы

1. При переносе данных через 1-Wire-Bus могут возникнуть повреждения. Контрольная сумма устанавливает, успешно ли прошла передача данных. Если да, то в конце первой строки температурного файла выставляется YES.
2. Уникальное обозначение ID позволяет передавать данные для конкретного датчика по линии связи, которую используют несколько датчиков. Каждый датчик распознаётся благодаря его ID, и его данные записываются в единый файл.
3. Звуковые волны – это механические колебания или изменения давления в воздухе. То есть они не являются электромагнитными волнами.
4. Один килогерц содержит тысячу колебаний в секунду, а один мегагерц – один миллион колебаний в секунду.
5. В команде содержится адрес (системный номер и Unit-номер). Если адрес совпадает с собственным адресом розетки, то розетка «понимает», что команда предназначена для неё.
6. Для положений рычажков ДИП-переключателя: вверх, вверх, вниз, вверх, вниз – системный номер будет 1 + 2 + 8 = 11.
7. Правила общения между передатчиком и приёмником называются *протоколом*.

Решение задач

Решение 1. Игра «Горячо – холодно»

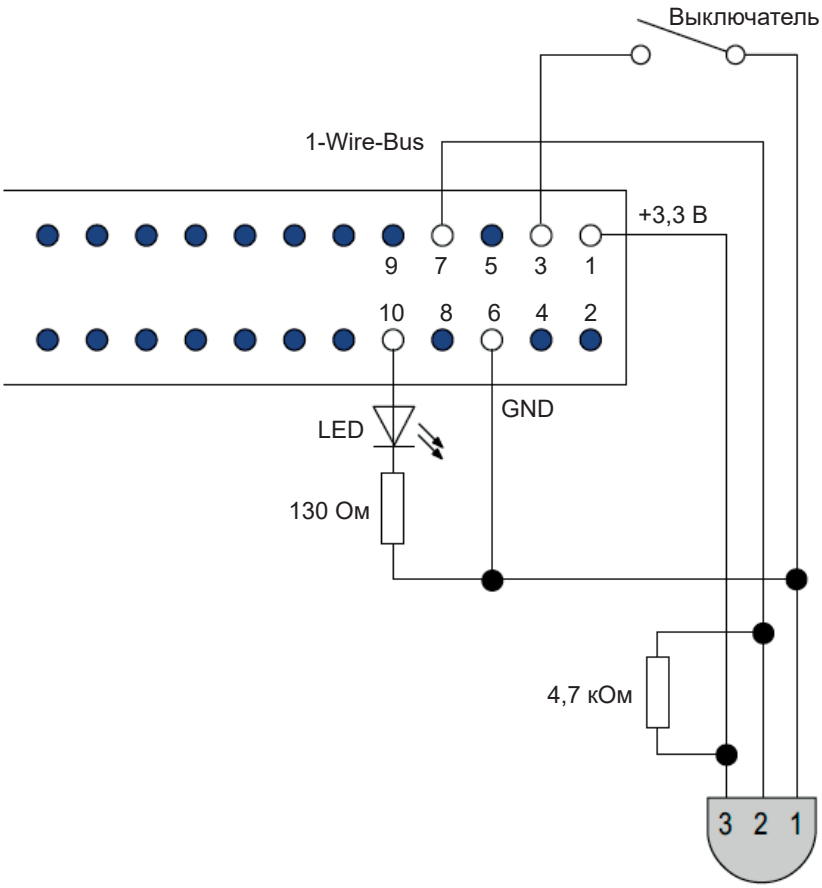


Рис. 15.11. *Схема подключения датчиков к контактам GPIO для поиска горячих мест*

Пишем программу

```
import os, time
from RPi import GPIO
SWITCH = 3                                          #1
LED = 10
FILE_NAME = '/sys/bus/w1/devices/xxx/w1_slave'      #2

def init():
    '''Инициализировать датчик и GPIO'''

    os.system('modprobe w1-gpio')
```

```
    os.system('modprobe w1-therm')
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(LED, GPIO.OUT)
    GPIO.setup(SWITCH, GPIO.IN)

def flash ():
    '''Светодиод мигает три раза'''
    for i in range(3):
        GPIO.output(LED, True)
        time.sleep(0.3)
        GPIO.output(LED, False)
        time.sleep(0.3)

def readTemp():
    '''Измеряем температуру в градусах Цельсия'''
    ok = False
    while not ok:
        f = open(FILE_NAME, 'r')
        Line_1, line_2 = f.readlines()
        f.close()
        if line_1.find('YES') != -1:
            ok = True
    path_1, path_2 = line_2.split('=')
    return int(path_2)/1000

# Главная программа
init()
Flash()
alt = readTemp()                                    #3
while GPIO.input(SWITCH):
    new = readTemp()                                #4
    if new > alt:                                   #5
        GPIO.output(LED, True)
    else:
        if new < alt:
            GPIO.output(LED, False)
    alt = new                                       #6
    time.sleep(0.5)

os.system('halt')                                   #7
```

Как это работает?

#1 Выключатель подключается к контакту 3 GPIO.

#2 В этой команде указывается путь к файлу с данными температуры. Вместо xxx введи обозначение датчиков. Эти обозначения ты с помощью файлового менеджера легко найдёшь в каталоге */sys/bus/wl/devices/*.

#3 Переменная alt содержит значение температуры из последнего цикла измерения. В этой командной строке данная переменная имеет начальное значение.

#4 Переменная new содержит последнее значение измеряемой температуры.

#5 Новое значение измерения сравнивается со старым: есть ли какие-то изменения?

#6 С этого момента новое значение измерения устаревает, и поэтому оно присваивается переменной alt. В следующем цикле оно может сравниться с другим значением измерения.

#7 RPi завершает работу.

Решение 2. «Незаконное проникновение!»

Для контроля программа выводит все данные на экран. Документируется всё, что сейчас происходит. То же самое делают стандартные программы, такие как apt-get. Программа содержит бесконечный цикл, а её работа завершается нажатием комбинации клавиш **Ctrl**+**C**.

Задача (пример)

```
Я наблюдаю за панелью управления.
Кто-то проник. Sun Oct 5 15: 37: 23 2014
Через 0 секунд блок 8 command -t.
Через 6 секунд блок 8 command -t.
Через 0 секунд блок 4 command -t.
Через 10 секунд блок 4 command -t.
Готово
```

Пишем программу

```
# action.py
from us_python import calculation_distance
import time, os
COMMAND = 'pilight-send -p mumbi -s 1 -u %s %s'
ISSUE = 'After %s Second Unit %s Commands %s.'

def taxes ():
    f = open('event.txt', 'r')
    line = f.readline()                          #1
    while line:                                  #2
       nr, second, command = line.split()        #3
       print(ISSUE % (second, nr, command))      #4
       time.sleep(int(second))                   #5
       os.system(COMMANDS %(nr, switch))         #6
```

```
        line = f.readline()                          #7
    f.close()

# Главная программа
print ('Я наблюдаю за панелью управления.')
while True:                                          #8
    if calculion_distance() < 150:                   #9
        print('Кто-то проник. ',time.asctime())
        taxes()
        print('Готово.')
    else:
        time.sleep(1)
```

Как это работает?

#1 Следующая строчка считывается из текстового файла.

#2 Пока есть ещё одна строка…

#3 Раздели строчки на три части. Символом разделения является пробел.

#4 Вывод для контроля. Константа `ISSUE` содержит три замещающих символа для времени ожидания, Unit-номер и команду (`-t` или `-f`).

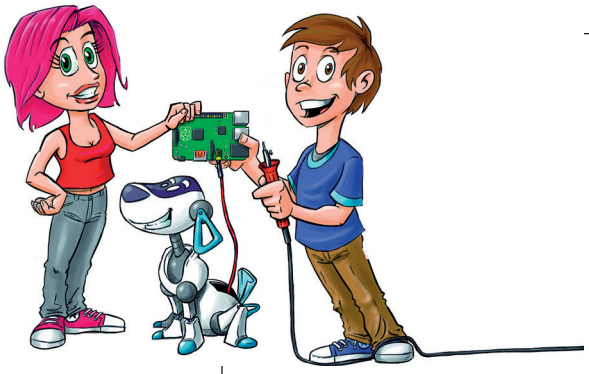
#5 Ожидание.

#6 Розетка включается.

#7 Считывается следующая строка.

#8 В главной программе проходит бесконечный цикл.

#9 Если объект находится от ультразвукового сенсора ближе, чем на 150 см, запускается последовательность событий.



16 Графический пользовательский интерфейс

В этой главе с помощью Python мы с тобой разработаем графический пользовательский интерфейс (*Graphical User Interface*, GUI). В пользовательском окне находятся поля для ввода, кнопки, текстовые элементы и изображения. Общение с компьютером больше напоминает не диалог, а работу с машиной. Нажимаешь на кнопку и получаешь ответ не только в виде текста, но и в виде изображения.

В своих проектах ты разработаешь графические программы с экранными кнопками, флажками, полями для ввода, обычными кнопками и фонами. Ты поэкспериментируешь с цветом, видами шрифтов и картинками.

Как создать пользовательский интерфейс?

Модуль `tkinter` содержит всё, что тебе нужно для создания с помощью Python графического интерфейса (GUI). При разработке тебе необходимо позаботиться о трёх вещах:

- виджеты: ты определишь «блоки» пользовательского интерфейса – так называемые виджеты. К ним относится окно (рамка) с маленькими кнопками в правом верхнем углу, с помощью которых это окно можно закрыть, свернуть или развернуть; а также кнопки, текстовые элементы, изображения, экранные кнопки, флажки и прочие составляющие;
- разметка: тебе придётся каким-то образом распределить все виджеты в пользовательском окне, например расположить их рядом друг с другом или же разместить один под другим. Это называется разметкой;
- интерактивность: наконец, тебе необходимо связать этот пользовательский интерфейс с какой-либо деятельностью. Например, закрепить за одной из кнопок определённую функцию, которая будет выполняться всякий раз при нажатии на эту кнопку.

Проект 43. «Сегодня ты выглядишь великолепно!»

Мы создадим приложение, которое будет мотивировать пользователя и дарить ему хорошее настроение. Графический интерфейс состоит из окна, метки и одной кнопки. Каждый раз при нажатии на кнопку графический интерфейс будет отображать новый, выбранный случайным образом текст.

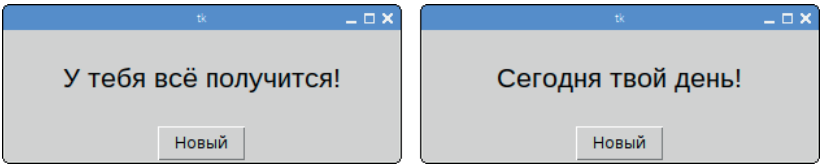


Рис. 16.1. *Пользовательский интерфейс с ободряющей цитатой*

Пишем программу

```
#! /usr/bin/python3                                          #1
# motivator.py
from tkinter import *
from random import choice
PROVERBS = ['Ты сегодня хорошо выглядишь!',
            'У тебя всё получится!', 'Сегодня твой день!',
            'Тебя никто не остановит!']
def select():                                                #2
```

```
    text = choice(PROVERBS)                                    #3
    label.config(text=text)                                    #4
# Widgets
window = Tk()                                                  #5
button = Button(master=window, text='Новый',
                command=select)                                #6
label = Label(master=window, width=25, height=3,
              font=('Arial', 16), text= PROVERBS[0])           #7
# Layout
label.pack()                                                   #8
button.pack()

window.mainloop()                                              #9
```

Как это работает?

#1 Эта строка предназначена для запуска программы по щелчку мыши. Но прежде ты должен активировать эту функцию (команда Linux `chmod 755`).

#2 Данная функция должна вызываться нажатием кнопки, расположенной на графическом интерфейсе программы. Следи за тем, чтобы она не содержала инструкцию `return`.

#3 Выбери новый случайный текст…

#4 … и вставь его в метку. Все объекты виджетов содержат метод `config()`, используя который, можно изменить характеристики.

#5 Здесь создаётся объект «окно».

#6 В этой командной строке создаётся кнопка и с помощью опций настраивается её внешний вид. `master` – это объект по имени `window`. На кнопке стоит текст `Новый`. При нажатии на неё выполняется функция `select()`.

#7 Здесь создаётся метка. Определим фон: вид шрифта `Arial`, размер букв 16 кеглей. Этот текст в метке является первым элементом списка `PROVERBS`.

#8 Метку и кнопки располагаем друг над другом. Сверху идёт метка, ниже – кнопки.

#9 Активируем пользовательское окно.

Устанавливаем виджеты

Внешний вид виджетов ты можешь настроить по своему усмотрению: цвет фона, тип шрифта, ширина, высота и другие критерии. Используй для этого соответствующие опции.

Попробуй сам. Что изменится, если в инструкции #7 опустить параметры `width` и `height`?

```
label = Label(master=window, font=("Arial", 20) text=PROVERBS[0])
```

Тогда метка (окно программы) быстренько подстроит свой размер под размер текста.

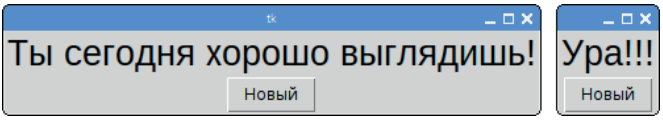


Рис. 16.2. *Метка без заданных параметров ширины и высоты*

Ты можешь также изменить цвет фона (`bg`) и цвет текста (`fg`) (рис. 16.3 слева):

```
label = Label(master=window,
              font=('Script', 30,'bold'),
              bg='blue', fg='white',
              text= PROVERBS[0])
```

Все виджеты имеют рамки. Рамка имеет определённую форму (опция `relief`) и ширину (опция `bd`). В виджетах метки используется форма границ `FLAT`, т. е. когда рамок не видно вовсе. Но ты можешь это изменить (рис. 16.3 справа):

```
label = Label(master=window, font=('Courer', 16),
                        relief=RIDGE, bd=4,
                        text= PROVERBS[0])
```

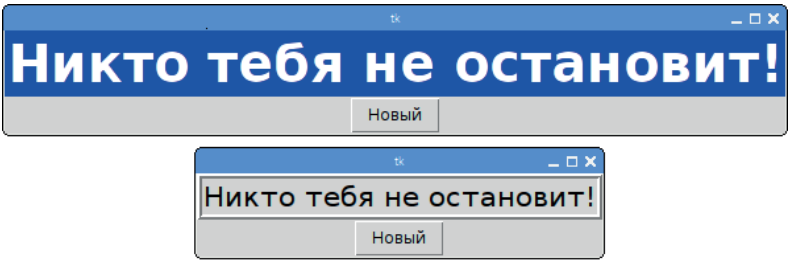


Рис. 16.3. *Цвета фона, шрифта, а также рамки виджетов изменены*

Базовые опции настройки одинаковы для общего вида всех виджетов. В табл. 16.1 представлен их обзор.

Таблица 16.1. Общие опции виджетов

Опция	Пояснение
`bd.borderwidth`	Ширина рамки, например 10
`bg.background`	Цвет фона
`fg.foreground`	Основной цвет (цвет текста)
`font`	Описание типов шрифта
`height`	Высота виджета (вертикально)
`image`	Название картинки (объект изображения), отображаемое на виджете
`justify`	Направление строк текста у виджетов: `CENTER` (по центру) `LEFT`, `RIGHT` (выровнять по правому или по левому краю)
`padx`	Пустое пространство справа или слева от виджета или от текста, например 10
`pady`	Пустое пространство над или под виджетом или текстом, например 10
`relief`	Форма рамки: `SUNKEN`, `RAISED`, `GROOVE`, `RIDGE`, `FLAT`
`text`	Маркировка виджета (например, кнопка или ярлык)
`width`	Ширина виджета (горизонтально)

Разметка

После того как новый виджет был создан, ему необходимо присвоить действие. В наших проектах это всегда окно приложения (Tk-объект). В разметке отмечено, где именно появится виджет на этом мастере. Используем два стиля разметки:

- упаковщик (менеджер расположения): виджет устанавливается в свободном месте мастера. Он касается кромки окна или уже размещённого виджета. Ты помещаешь (упаковываешь) виджет в окно точно так же, как упаковываешь свою одежду и другие вещи в чемодан;
- разметка-растр: виджет устанавливается в определённую ячейку сетки из строк и столбцов. Это приблизительно то же самое, как если бы ты поставил фигуру на шахматную доску.

В программе-образце использовался упаковщик. Для упаковки виджета применяется метод `pack()`. Если ты вызываешь метод без этих параметров, то новый виджет просто помещается под предыдущий объект. Что произойдёт, если поменять местами последовательность обоих методов `pack()`?

```
button.pack()
label.pack()
```

Теперь кнопка находится в верхней части метки.

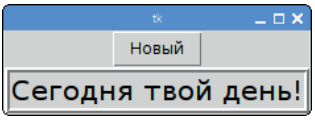


Рис. 16.4. *Кнопка и метка обратной последовательности*

Для метода `pack()` есть несколько опций. Если ты хочешь разместить виджеты рядом друг с другом, а не один над другим, то используй опцию `side=LEFT`:

```
button.pack(side=LEFT)
label.pack(side=LEFT)
```

Рисунок 16.5 (слева) демонстрирует результат такого действия. Возможно, тебе мешает тот факт, что над кнопками и под ними есть ещё немного места. Если ты хочешь подогнать размер кнопок в вертикальном положении (ось Y) под размер окна приложения, используй опцию `fill=Y`:

```
button.pack(side=LEFT, fill=Y)
```

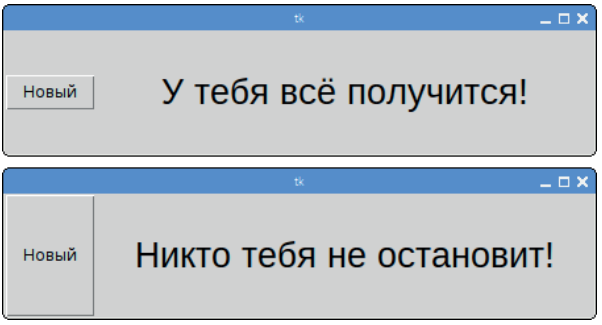


Рис. 16.5. *Установить виджеты рядом друг с другом можно с помощью опции `side=LET`. А используя параметр `fill=Y`, мы изменим высоту кнопки до высоты окна*

Таблица 16.2. Важные параметры упаковщика

Опция	Пояснение
anchor	Возможные значения: CENTER, E, N, W, S, NE, NW, SE, SW. Эта опция определяет расположение по «сторонам света». По умолчанию это CENTER – в середине, NE – правый верхний угол (северо-восток), N – установленный по центру вверху страницы (север), E – установленный по центру в правой части страницы и т. д.
fill	fill=X: виджет заполняется в горизонтальном направлении (ось X) до тех пор, пока имеется свободное пространство, чтобы его размер соответствовал размеру мастера
side	LEFT, RIGHT, TOP, BOTTOM: виджет устанавливается в левой, правой, верхней или нижней части мастера

Картинки в виджетах

С помощью категории tkinter PhotoImage ты можешь вставлять картинки в приложение. Ты создашь объект PhotoImage, используя инструкцию следующего типа:

```
bild = PhotoImage(file=path)
```

При этом path – это адрес файла картинки в формате GIF, PPM или PGM. Конечно, это большое ограничение. К сожалению, форматы JPEG или PNG не поддерживаются. Для таких файлов тебе понадобится специальный модуль, о котором мы поговорим позже.

Объект PhotoImage ты можешь поместить на кнопку или метку, используя параметр image. Возможно, тебе хочется, чтобы кнопка или мотивирующая цитата вмещала в себя какую-нибудь картинку, а не слово «New». Что-то, что символизировало было вдохновение или движение к цели. Как насчёт ракеты?

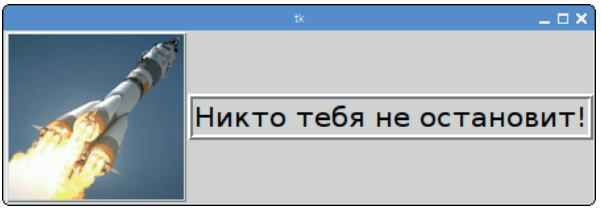


Рис. 16.6. *Кнопка с картинкой*

Картинку можно скачать из интернета или нарисовать самому в графическом редакторе. Сохрани файл картинки

(в формате GIF) в своей папке проектов. Обрати внимание: чтобы всё сработало правильно, измени имя скачанной фотографии (рисунка) на `'missile.gif'`. Замени в программе строку `#6` из раздела «Проект 43. Сегодня ты выглядишь великолепно!» на две строки программного текста:

```
bild = PhotoImage(file='missile.gif')
button = Button(master=window, image=bild, command=select)
```

Цвета

Цвета в графических пользовательских интерфейсах играют огромную роль. Ты их будешь использовать, например, для фона (опция `bg`) или для шрифта (опция `fg`). Отобразить основные цвета ты можешь с помощью текста, например `'black'` для чёрного, `'white'` для белого, `'blue'` для синего и т. д. Также можно описать цвета, используя цифры. Один такой номер – это строка символов, которая начинается со знака `#`. Затем идут три шестнадцатеричных числа, представляющих компоненты красного, зелёного и синего цветов. Что такое шестнадцатеричные числа? Из повседневной жизни тебе известны *десятичные* числа, которые имеют десять разных цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. *Шестнадцатеричные* числа имеют 16 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Здесь вместо цифр используются буквы от A до F. Цифра A имеет числовое значение 10, цифра B – значение 11, ну и т. д. В табл. 16.3 приведено несколько примеров. Цвет можно представить тремя одноразрядными шестнадцатеричными числами (в общей сложности три цифры). Это позволит тебе определить 4096 различных цветов (16 × 16 × 16). Но ты также можешь использовать двухразрядные шестнадцатеричные числа (т. е. всего 16 цифр). Тогда станет возможным определить почти 17 000 000 цветов.

Таблица 16.3. Примеры номеров для определения цвета

Номер цвета	Десятичные числа (красный, зелёный, синий)	Цвет
`'#FFF'`	15, 15, 15	Белый
`'#000'`	0, 0, 0	Чёрный
`'#F00'`	15, 0, 0	Красный
`'#FF0'`	15, 15, 0	Жёлтый
`'#0F0'`	0, 10, 0	Тёмно-зелёный
`'#0000FF'`	0, 0, 255	Синий

Проект 44. «Смешивание цветов»

Графический пользовательский интерфейс состоит из цветной метки, под которой находятся маленькое поле для ввода и одна кнопка. В поле для ввода тебе нужно ввести номер цвета (без кавычек), например #F00. Далее нажми на кнопку, и цвет отобразится.

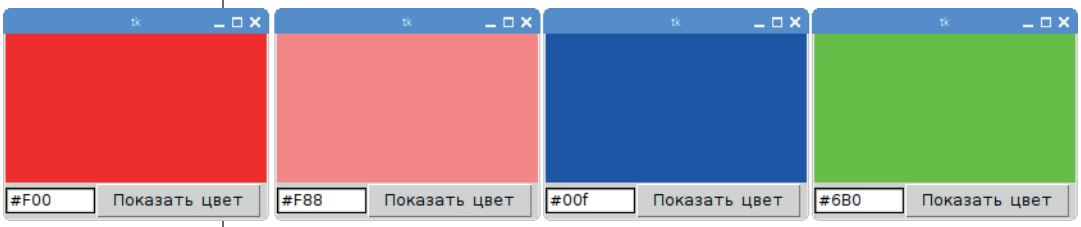


Рис. 16.7. *Так выглядит красный (#F00), светло-красный (F88), тёмно-голубой (#00F) и оттенок зелёного цвета (#6B0) с использованием параметра смешивания цветов*

Виджет entry

Для поля ввода, состоящего из одной строки, ты будешь использовать виджет **entry** – однострочное текстовое поле. Помимо обычных параметров и методов, которыми обладают все виджеты в целом, **entry** обладает дополнительными характеристиками. Важнейшим из них является метод `get()`. С его помощью ты получишь текст, введённый в поле ввода.

Таблица 16.4. Список особых параметров и методов виджета **entry**

Метод/опция	Пояснение
`delete(`*`first`*`[, `*`last`*`])`	Знаки из поля **entry** можно удалить. Параметр *`first`* – это индекс первого удалённого знака. Если второй аргумент отсутствует, тогда удаляется только этот символ. В противном случае удаляются все символы, идущие до индекса *`last`*
`get()`	Содержимое поля **entry** возвращается в виде строки символов
`show`	Параметр `show='`*`x`*`'` способствует тому, чтобы вместо введённого символа в поле entry отображался знак *x*. Это важно для ввода пароля

Программирование. Пишем программу

```
from tkinter import *

def color_change():
    label.config(bg=input.get())                    #1

# Widgets
window = Tk()
label = Label(master=window, width=25, height=8)
input = Entry(master=window, width=8)               #2
button = Button(master=window, text='Color show',
                command= color_change)
# Layout
label.pack()
input.pack(side=LEFT)
button.pack(side=LEFT)
window.mainloop()
```

Как это работает?

#1 С помощью функции `get()` формируется поле ввода, в которое пользователь вводит в виджете **entry** номера нужных цветов, например `'#F00'`. Эта строка (которая представляет цвет) назначается опцией `bg` в метке.

#2 Здесь виджет **entry** настраивается для всех символов.

Проект 45. Сигнальная азбука

В мореплавании все команды между кораблями передаются с помощью флагов. Самый известный пример – это пиратский флаг с черепом и костями. Красный флаг предупреждает, что судно перевозит опасный груз. Уже примерно 200 лет существует так называемая сигнальная азбука, где каждой букве присвоен свой цветной флаг. Несколько флагов, выстроенных в определённой последовательности, отображают целый текст. Мы с тобой напишем программу, которая будет преобразовывать слово в набор флагов (рис. 16.8).

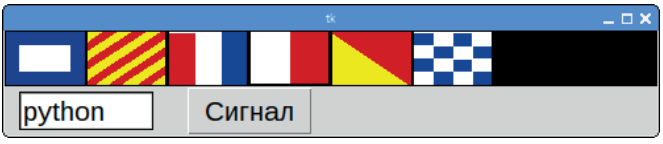


Рис. 16.8. *Слово из букв, показанных с помощью флагов*

В левом нижнем углу окна нашей программы мы поместим поле ввода, в которое вводится слово. После нажатия на кнопку **Сигнал** в верхней строке появятся флаги, отображающие это слово.

Подготовка

Перед тем как написать программу, тебе следует в папке проектов создать папку, в которой будут храниться одинаковые большие картинки флагов из сигнальной азбуки. Следует заметить сразу, для нашего проекта требуются изображения флажков в формате *.gif*, и размер каждого изображения должен быть 60×40. В интернете найти такие изображения сигнальных флажков сложно. Необходимо скачать большие изображения в формате *.jpeg*, преобразовать в формат *.gif* и уменьшить до нужного размера. Найти изображения сигнальных флагов можно, например, по адресу: http://seaman-sea.ru/mcc.html. Но здесь ты скачаешь изображения флагов в формате *.jpeg* и размером 658×639 пикселов (23,21×22,54 см). Нам же нужно это изображение обрезать от лишнего фона, преобразовать в формат *.gif* размером 60×40 пикселов. Иначе флаги не будут отображаться.

Конечно, флаги можно скачать из интернета и преобразовать их. Но со стандартным программным обеспечением Raspberry Pi преобразовать картинки до нужного размера затруднительно. Поэтому если ты, читая седьмую главу, скачал папку *9873958457676_Projecte.zip*, открой эту папку, открой вложенную папку *kap_16*, далее – папку *Flaggen* и воспользуйся сохранёнными в этой папке изображениями.

- Создай в папке *Python_project* папку *Flags*.
- Скопируй в эту папку изображения сигнальных флажков и фона (файл *leer.gif*) из папки *9873958457676_Projecte.zip/kap_16/Flaggen*.

Файл *leer.gif* – это чёрный фон, он нам также необходим. В переводе с немецкого *leer* – «пустой». Не будем изменять имя файла и в программе воспользуемся этим именем.

Теперь о названии файлов изображений. Названия этих файлов будут соответствовать вводимым в поле ввода буквам. Букве A (латиница) соответствует файл с изображением флага *A.gif*, букве B соответствует файл *B.gif*, букве C – файл *C.gif* и т. д. Фоновым цветом должен быть чёрный.

Кстати, вместо флагов можно использовать и другие картинки, например:

- предметы, чьё название начинается с определённых букв (т. е. P – *пудель*);

- жесты из азбуки языка жестов, которые используют глухонемые (например указательный палец вперёд и средний вниз будут обозначать букву P).

Рис. 16.9. *Перевод слова Python на язык жестов (американская версия)*

Растр

Для разметки виджета мы будем использовать метод `grid()`. С его помощью виджет будет помещён в растр. Координаты каждого виджета определяются номером строки (`row`) и номером столбца (`column`). Широкий виджет (например, поле ввода) можно разделить на несколько столбцов. Используя опцию `columnspan`, ты укажешь количество столбцов, которые должны охватывать виджет.

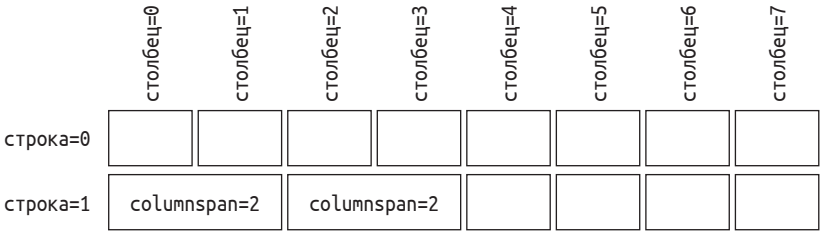


Рис. 16.10. *Разметка растра*

Программирование

Важнейшей структурой файла программы является словарь `flag`. В этом словаре каждой букве алфавита присваивается картинка флага, например вот так объект `PhotoImage` отображает флаг буквы A:

```
flag['A']
```

Затем эта картинка помещается в метку.

Пишем программу

```
from tkinter import *
def give_signal():
    '''Представляем буквы флагами'''
    word = input.get()                          #1
    for i in range(8):                          #2
```

```
        if i < len(word):                          #3
            b = word[i].upper()                    #4
            label[i].config(image=flag[b])         #5
        else:
            label[i].config(image=leer)            #6

# Widgets
window = Tk()
leer = PhotoImage(file='leer.gif')                 #7
label = []                                         #8
for i in range(8):
    new = Label(master=window, bg='black',
                width=60, height=40,
                image=leer)                        #9
    label.append(new)                              #10
input = Entry(master=window, width=8,
              font=('Arial', 16))                  #11
button = Button(master=window, text='Сигнал',
                font=('Arial', 16),
                command=give_signal)
# Layout
for i in range(8):
    label[i].grid(column=i, row=0)                 #12
input.grid(row=1, column=0, columnspan=2)          #13
button.grid(row=1, column=2, columnspan=2)         #14

flag = {}                                          #15
for letter in 'ABCDEFGHIJKLMNOPQRSTUVWXYZ':
    bild = PhotoImage(file=letter+'.gif')          #16
    flag[letter] = bild                            #17
window.mainloop()
```

Как это работает?

#1 Переменной `word` присваивается прочитанный из поля ввода текст.

#2 Здесь формируется поле для отображения флагов. Если значение `range` равно 8, значит, будет отображаться максимум восемь флагов. Переменная `i` имеет в итерации числа от 0 до 7. Если значение `range` увеличить до 10, будет отображаться 10 флагов, а переменная будет иметь в итерации числа от 0 до 9.

#3 Если окончание слова ещё не определено…

#4 … создай из буквы i заглавную букву…

#5 … и установи в поле отображения флагов на соответствующее очерёдности буквы место `i` флаг, который соотносится с введенной буквой. Метка `i` обозначает текущее месторасположение очередного флага. То есть

если в поле ввода это третья буква, значит, флаг будет поставлен в столбец 2 строки 0. Обрати внимание: нумерация столбцов начинается с цифры 0.

#6 В случае если окончание слова уже определено и других букв нет, на метку `i` помещается пустая чёрная картинка.

#7 Примечание: лишь после создания окна с помощью функции `Tk()` можно создать экземпляры объектов `PhotoImage`. В противном случае система выдаст сообщение об ошибке. Здесь объект `PhotoImage` создаётся с использованием чёрного прямоугольника.

#8 Создаём пустой список, который заполняется восемью виджетами меток.

#9 Новая метка-виджет создаётся с чёрным фоном и чёрной картинкой. Чернее не бывает. Обрати внимание на значение `width=60`, `height=40`. Это значение размера изображений наших флажков. Если у тебя в папке сохранены файлы с изображениями флагов других размеров (например, 70×45), значит, вместо значений `width=60`, `height=40` введи значения `width=70`, `height=45`.

#10 Новая метка прикрепляется к списку.

#11 Создаётся поле ввода для восьми символов (букв). Количество вводимых символов определяется значением `width=8`, а шрифт и его размер, которым будут отображаться в поле ввода вводимые буквы, определяются строкой `font=('Arial', 16)`.

#12 Теперь идёт разметка. Здесь создаётся поле, на котором будут отображаться флаги. Восемь меток виджетов в списке помещаются в первую строку растра (`row=0`).

#13 Виджет **entry** (поле ввода для букв) появляется в нижнем левом углу программы (растра). Этот виджет занимает в ширину два столбца – две объединённые ячейки (столбец 0 и столбец 1, строка 1) (рис. 16.10). Обрати внимание: здесь обозначается первый столбец, в котором начинается виджет, и определяется, сколько столбцов он занимает. То есть строка (`row=1, column=0, columnspan=2`) обозначает, что поле ввода находится в строке 1 (`row=1`), начинается в столбце 0 (`column=0`) и объединяет две ячейки (`columnspan=2`).

#14 Здесь определяется местоположение кнопки, для которой зарезервированы две объединённые ячейки (`row=1, column=2, columnspan=2`).

#15 Создается переменная `flag`, которая формируется следующим образом.

#16 Для введенной буквы выбирается соответствующая картинка в виде объекта `PhotoImage`...

#17 ... и вносится в словарь.

Проект 46. Делаем гимнастику с ультразвуком

Теперь поговорим о спорте. На стене висит LCD-монитор с Raspberry Pi. Под потолком закреплён ультразвуковой датчик, соединённый с GPIO с помощью длинных кабелей и небольшой электрической схемы. В точности как в разделе «Проекты с ультразвуковым датчиком». Ты стоишь прямо под этим датчиком и смотришь на экран. «Тренер» показывает, что нужно делать. Стойка прямо, руки поднимаем вверх, затем садимся на корточки. Всё выполняется в произвольном порядке. Ультразвуковой датчик контролирует твои движения. Как только ты примешь позицию «тренера», получаешь очко, а «тренер» переходит к следующей позиции. Часы идут в обратном направлении и останавливаются на нуле (левая метка). Цель состоит в том, чтобы в течение 120 с заработать как можно больше очков. При этом ты можешь изрядно вспотеть.

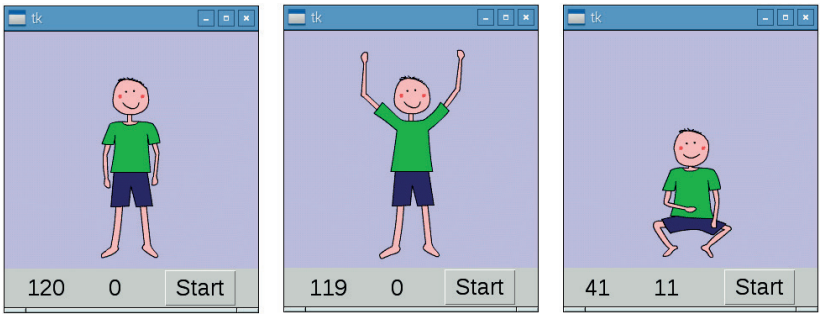


Рис. 16.11. *Повторяя движения за фигурой (человечком), ты зарабатываешь очки*

Подготовка

Для проекта нужно создать отдельную папку, например *Gimnastika*. С помощью графических инструментов подготовь три одинаковые большие картинки, на которых человек изображён в трёх позах. Сохрани эти картинки в папке проекта в формате GIF. Можешь воспользоваться картинками из папки *9873958457676_Projecte.zip/kap_16/Gymnastik*.

Скопируй в эту папку модуль для измерения расстояния *us_python.py* из проекта 38 главы 14. В папке *9873958457676_Projecte.zip/kap_16/Gymnastik* это файл *us_python.py*.

```
# Main_program
init_gpio()
print('GPIO инициализорован')
```

Программируем

Особенностью данного проекта является то, что функция `run()` выполняется в собственном Thread (потоке управления).

Что такое Thread?

В графическом пользовательском интерфейсе кнопка остаётся нажатой до тех пор, пока соответствующая функция не будет завершена. Если выполнение функции занимает слишком много времени, это выглядит неестественно.

Если функция выполняется в новом потоке, то ждать никого не надо. Эта функция выполняется параллельно остальным функциям, и всё остальное работает в привычном режиме. То есть кнопки из нажатого состояния сразу же возвращаются обратно. Вот это уже выглядит лучше. Слово «Thread» в переводе на русский означает «поток (управления)». Его ты можешь себе представить в виде сюжетной линии в какой-нибудь истории, линии, которая разворачивается параллельно другим действиям.

Используя функцию `start_new_tread()` из модуля `_thread`, можно выполнить функцию в отдельном потоке.

Пишем программу

Обрати внимание: здесь используется ранее определенная функция `calculate_distance`. Но, так как в скопированном нами из папки *9873958457676_Projecte.zip/kap_16/Gymnastik* файле *us_python.py* эта функция называется `berechne_abstand`, в программе оставим именно это название функции. Если же ты будешь использовать эту функцию из своей папки и она у тебя называется `calculate_distance`, измени во всей программе название этой функции `berechne_abstand` на `calculate_distance`:

```
#! /usr/bin/python3
from us_python import berechne_abstand
from tkinter import *
```

```
import random, time, _thread                                    #1

def new_position ():
    ''' Установить новое изображение на этикетке.'''
    global position                                             #2
    new = position
    while new == position:                                      #3
        positions = list(bilder.keys())
        new = random.choice(positions)
    position = new                                              #4
    bild_label.config(image=bilder[position])                   #5
def start():
    ''' Запуск нового приложения'''
    global time, point
    time = 120
    time_label.config(text=str(point))
    Point = 0
    _tread.start_new_thread(run, ())_                           #6

def ferting():
    ''' Проверяет, сделал ли игрок упражнение'''
    distance = berechne_abstand()
    print(distance)                                             #7
    if position == 'high':
        return distance < 60                                    #8
        if position == 'stand':
        return 60 < distance < 140
    else:
        return distance > 140

def run():                                                      #9
    global time, point
    while time > 0:                                             #10
        new_position()
        time.sleep(1)
        while not fertig() and (time > 0):
            time.sleep(1)
            time -= 1
            time_label.config(text=str(time))
        point += 1
        point_label.config(text=str(point))

# Main_program
position='stand'
window = Tk()                                                   #11
bilder = 'stan':PhotoImage(file='stehen.gif'),
         'high':PhotoImage(file='hoch.gif'),
         'crouch':PhotoImage(file='hocke.gif')}
bild_label = Label(master=window, image=bilder['stan'])
bild_label.grid(row=0, column=0, columnspan=3)
```

```
time_label = Label(master=window time_label,
                   text='120',font=('Arial', 20))
time_label.grid(row=1, column=0)
point_label = Label(master=window, text='0',font=('Arial', 20))
point_label.grid(row=1, column=1)
button = Button(master=window, command=start, text='Start',
                font=('Arial', 20))
button.grid(row=1, column=2)
window.mainloop()
```

Как это работает?

#1 Нам потребуется модуль `_thread`. В нём определяется функция `start_new_thread()`.

#2 Переменная `position` содержит строку символов `'stand'` или `'high'`, объясняющую от имени «тренера», какое положение подопечный должен принять. Поскольку значение этой переменной в этой функции меняется, она должна быть объявлена как глобальная.

#3 В этом цикле происходит поиск новой позиции, отличающейся от текущего положения тела.

#4 Цикл завершается, если `position` отличается от `new`. Теперь переменная `position` принимает значение `new`.

#5 В словаре `image` выбирается картинка, соответствующая описанию позиции для переменной `position`. Эта картинка помещается в метку. Фигура (человечек) на картинке принимает другое положение тела.

#6 Функция `run()` выполняется в новом потоке. Функции `start_new_thread()` требуется два аргумента. Первый аргумент – это имя функции, а второй – кортеж с аргументами для этой функции. Функция `run()` вообще не имеет аргументов, поэтому там стоит пустой кортеж `()`.

#7 Эта инструкция служит лишь для проверки работы измерения расстояния. Её ты можешь завершить.

#8 Идёт проверка, выполняет ли игрок рекомендации «тренера» и какие позы он принимает. Например, если переменная `position` имеет значение `'high'` и изображённый на экране человечек поднял руки вверх, игрок тоже должен поднять и выпрямить руки над головой. Если он это сделал, ультразвуковой датчик, закреплённый на потолке, фиксирует изменение расстояния от датчика до игрока, например 60 см. Тебе, возможно, придётся изменить эту цифру в соответствии с высотой потолка в твоей квартире и твоим ростом. Например, если у тебя невероятно высокий потолок, то вмес-

то значения 60 введи цифру 70. Если расстояние будет менее 60 см, то возвращается истинное значение `True`, в остальных случаях будет `False`.

`#9` Функция `run()` выполняется в отдельном потоке (см. строку `#6`).

`#10` До тех пор пока время игры не вышло, на экране постоянно отображается новая позиция «тренера» (например присесть), а затем система ждёт, пока игрок примет то же положение тела.

`#11` В следующей строке определяется словарь. Он закрепляет объект `PhotoImage` за каждым описанием позиции (например, `'stan'` показывает стоящего человечка).

Делаем выбор с помощью экранных переключателей и чекбоксов

Для выбора той или иной опции существуют чекбоксы и экранные кнопки.

- *Экранный переключатель* всегда относится к одной группе. В этой группе выбрать можно только *один* переключатель (простой выбор). Если щёлкнуть мышью на этом переключателе, то ранее выбранный переключатель перейдёт в состояние отключённого. Экранные переключатели чаще всего представлены в виде круглых элементов с точкой внутри (точка появляется в случае выбора).
- *Чекбокс* – это своего рода переключатель. Он может быть в двух состояниях: включённом и выключенном. То есть «выбран» или «не выбран». Этот элемент управления позволяет выбрать одновременно несколько опций (расширенный выбор). Чаще всего чекбокс представлен в виде клетки с флажком (флажок устанавливается в случае выбора).

Проект 47. Выбери цвет

Цветное освещение влияет на наше эмоциональное состояние. Тебе известно, например, что в современных самолётах подсветка салона постоянно меняет свой цвет? Освещение при посадке отличается от освещения во время еды.

В этом проекте ты будешь использовать экран своего Raspberry Pi как светильник. Ты создашь дизайн освещения для различных ситуаций. Используя экранные кнопки, ты сможешь выбирать подходящий цвет.

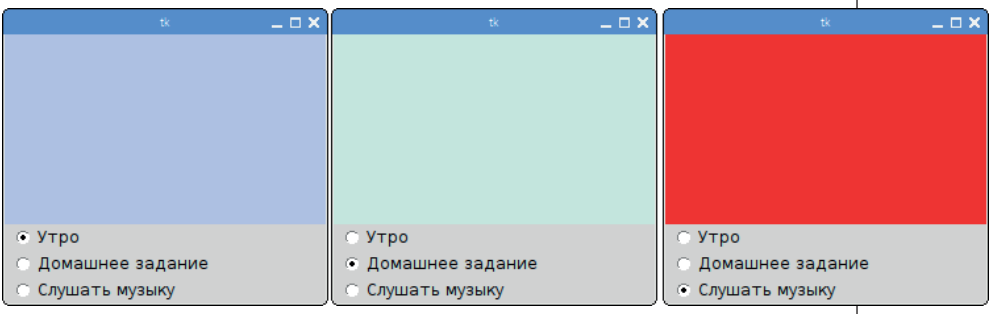


Рис. 16.12. *Экранные кнопки для выбора цвета*

Создать экранную кнопку ты можешь, используя следующую форму:

```
rb = Radiobutton(master=window,
                 variable=control_variable,
                 value=value,
                 command=function,
                 text=текст_объявления)
```

С помощью опции `valiabre` (переменная) ты определишь контрольную переменную. Экранные переключатели соответствующей группы имеют общую контрольную переменную. Каждой из них присвоено значение (опция `value`). У контрольной переменной всегда значение только что выбранной кнопки. Опция `command` определяет функцию, выбранную в случае нажатия кнопки. Опция `text` содержит текст, который будет виден справа от переключателя рядом с круглым полем.

Контрольные переменные – это не простые переменные, а невидимые виджеты. Существует несколько типов контрольных переменных: `StringVar` для цепи символов и `IntVar` для целых чисел. Используя метод `set()`, можно поставить значение контрольной переменной. А с помощью `get()` это значение можно запросить.

Пишем программу

```
from tkinter import *

def color_change():
```

```
    label.config(bg=color.get())                                    #1

# Widget
window = Tk()
color = StringVar()                                                 #2
label = Label(master=window, width=30, height=10)                   #3
rb1 = Radiobutton (master=window, variable=color,
                   value='#BCF', command= color_change,
                   text='Утро')                                     #4
rb2 = Radiobutton (master=window, variable=color,
                   value='#CFE', command= color_change,
                   text='Домашнее задание')
rb3 = Radiobutton (master=window, variable=color,
                   value='#E22', command= color_change,
                   text='Слушать музыку')
rb1.select()                                                        #5
color_change()                                                      #6

# Layout
label.pack()
rb1.pack(anchor=W)
rb2.pack(anchor=W)
rb3.pack(anchor=W)
window.mainloop()
```

Как это работает?

#1 Функция вызывается при нажатии экранного переключателя.

#2 Здесь настраивается контрольная переменная для цепочки символов. У этой переменной будет значение выбранной в данный момент кнопки.

#3 Метка без текста. Эта командная строка определяет размер экрана, на котором будет отображаться образец цвета. Сейчас для нас важен только фоновый цвет.

#4 Создаётся первая экранная кнопка. Как и две другие, кнопка связана с функцией `color_change()` (изменение цвета) и с контрольной переменной. При нажатии кнопки контрольная переменная получает значение `'#BCF'`. Это очень светлый синий цвет. Кроме того, вызывается функция `color_change()`.

#5 Выбирается первая экранная кнопка. То есть после старта автоматически включается этот переключатель.

#6 Вызывается функция `color_change()`. Она обеспечивает заполнение метки выбранным цветом (в нашем случае для переключателя «Утро» это светло-синий цвет).

Проект 48. Меню-консультант

Что у тебя сегодня на ужин? Не знаешь, чего хотел бы съесть? Меню-консультант тебе в помощь. Просто выбери вариант с помощью чекбокса, и консультант тебе что-нибудь предложит.

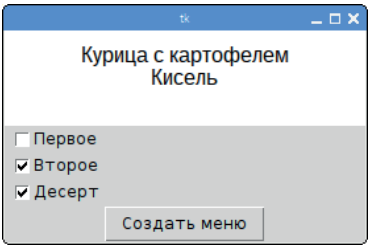


Рис. 16.13. *Меню-консультант с чекбоксом. Выбрано только второе блюдо и десерт*

Кнопка **Checkbutton** может находиться в двух состояниях: **On** (флажок установлен) и **Off** (флажок не установлен). Каждому из двух состояний присваивается значение (параметры `onvalue` и `offvalue`). Текущее значение хранится в переменной управления (опция `variable`).

Пишем программу

```
from tkinter import *
from random import choice

ПЕРВОЕ = ['Зеленый Салат', 'Борщ', 'Суп']
ВТОРОЕ = ['Курица с картофелем', 'Плов', 'Лапша с котлетой']
ДЕСЕРТ = ['Мороженое', 'Кисель', 'Компот', 'Пирожное']      #1

def propose():
    '''Создание меню'''
    text = ''                                               #2
    if appetizer.get() == 1:
        text += choice(ПЕРВОЕ)+ '\n'
    if main_course.get() == 1:
        text += choice(ВТОРОЕ)+ '\n'
    if dessert.get() == 1:
        text += choice(ДЕСЕРТ)
    label.config(text=text)

window = Tk()
# Переменные управления
appetizer = IntVar()
main_course = IntVar()                                      #3
```

```
dessert = IntVar()

# Виджет
label = Label(master=window, width=30, height=4,
              font=('Arial',12), bg='white')
cb1 = Checkbutton (master=window, variable= appetizer,       #4
                   offvalue=0, onvalue=1, text='Первое')
Cb2 = Checkbutton (master=window, variable= main_course,
                   offvalue=0, onvalue=1, text='Второе')
Cb3 = Checkbutton (master=window, variable= dessert,
                   offvalue=0, onvalue=1, text='Десерт')
button=Button(master=window, command= propose,              #5
              text='Создать меню')
# Layout
label.pack()
cb1.pack(anchor=W)
cb2.pack(anchor=W)
cb3.pack(anchor=W)
button.pack()

window.mainloop()
```

Как это работает?

#1 Для каждого варианта (`ПЕРВОЕ`, `ВТОРОЕ` и `ДЕСЕРТ`) существует список возможных блюд. Из него в произвольном порядке выбирается блюдо для создаваемого меню.

#2 Здесь создаётся следующий пустой текст, который постепенно будет расширен до меню.

#3 Для каждого варианта меню существует контрольная переменная для целых чисел (`appetizer`, `main_course` или `dessert`). Каждая переменная относится к конкретному чекбоксу.

#4 Здесь создаётся первая кнопка чекбокса, связанная с контрольной переменной `appetizer` (закуска). Кнопка имеет значение 1 в случае, если она была выбрана, и значение 0, если кнопка не выбрана.

#5 Определяется кнопка, которая будет связана с функцией `propose()` (предложить). При нажатии на кнопку создаётся меню и в соответствии с опцией выдаётся в метку.

Автоматическое создание текстов

Программа, конечно же, является лишь основным образцом для автоматического создания текстов с опциями. Возможно, ты захочешь разработать что-то и для другой темы:

- приветствие из отпуска (опции: погода, жильё, экскурсии);
- поздравления с днём рождения (опции: здоровье, успех, счастье);
- приглашение на вечеринку по случаю дня рождения (опции: принести напитки, позвать друзей, ужин);
- лесть (опции: хорошо выглядишь, имеешь чувство юмора, темперамент, интеллигентный).

Вопросы

1. Переменные управления также являются виджетами, потому что они связаны с окном приложения (window). Чем отличаются контрольные переменные из других виджетов, таких как этикетки или кнопки?
2. PhotoImage, Label, Button, Canvas, Radiobutton и Checkbutton являются важными элементами графического интерфейса пользователя. Какие из этих виджетов служат для вывода, а какие – для ввода данных?
3. Какая из следующих ситуаций напоминает выполнение функции в новом потоке? Ситуация A: во время просмотра футбола по телевизору Вася делал свои домашние задания по математике. Ситуация B: сперва Вася сделает домашние задания, а потом посмотрит футбольный матч.
4. Что общего имеют номера оттенков серого?

Задания. Таймер

Создай таймер с графическим интерфейсом, как показано на рис. 16.14. При нажатии на кнопку **Пуск** часы начнут отсчёт с 0 с. Они будут непрерывно отображать время с точностью 0,001 с. При нажатии на **Стоп** часы останавливаются.

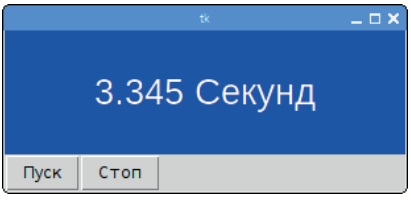


Рис. 16.14. *Таймер*

Ответы на вопросы

1. Контрольные переменные остаются невидимыми.
2. Метка и Canvas предназначены больше для вывода данных, так как они отображают данные в тексте и на картинке. Кнопка **Entry**, кнопка чекбокса, экранная кнопка предназначены для ввода данных, так как пользователь может их изменить.
3. Ситуация А соответствует выполнению функции в новом потоке. Здесь параллельно проходят два действия (не последовательно, а одновременно).
4. Оттенки серого имеют такие же числа, как и для красного, зелёного и синего цветов, например #111 или #777.

Решение задач

Пишем программу

```
#stopwatch.py
from tkinter import *
import time, _thread

def run():                                              #1
    while go:                                           #2
    time_value = round(time.time()-start_time, 3)       #3
    label.config(text=str(time_value)+' Секунд')        #4
    time.sleep(0.001)

def start():
    global start_time, go                               #5
    start_time = time.time()
    go = True
    _thread.start_new_thread(run, ())                   #6

def stop():
    global go
    go = False                                          #7

# Widget
window = Tk()

label = Label(master=window, font=('Arial',20),
              fg='white', bg='blue', text='0.000 Секунд',
              width=20, height=3)

start_button = Button(master=window, text='Пуск', command=start)
```

```
stop_button = Button(master=window, text='Стоп', command=stop)

# Layout
label.pack()
start_button.pack(side=LEFT, anchor=W)
stop_button.pack(side=LEFT, anchor=W)
window.mainloop()
```

Как это работает?

#1 Функция `run()` выполняется в отдельном потоке. Она позволяет отображать в метке текущее время.

#2 Цикл выполняется, пока у переменной `go` значение `True`.

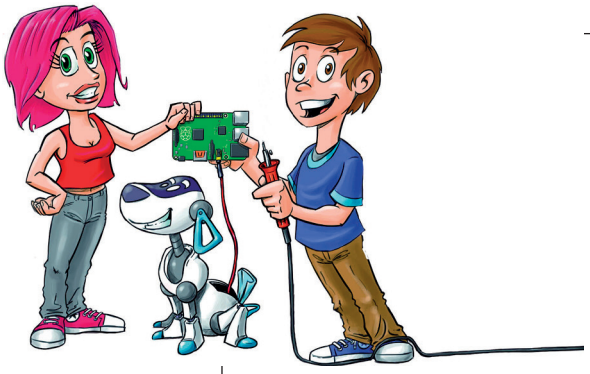
#3 Здесь высчитывается количество секунд с начала старта таймера, а потом округляется до трёх разрядов после запятой.

#4 В метке отображается текущее время.

#5 В этой функции изменяются значения для времени старта и управляющей переменной `go`. Поэтому они должны быть помечены как глобальные.

#6 Функция `run()` запускается в новом потоке. Она выполняется и в фоновом режиме. В остальных случаях использование кнопок пользовательского интерфейса было бы невозможно.

#7 Это позволяет приостановить выполнение функции `run()`.



17
Работа с камерой

В этой главе мы с тобой исследуем область цифровой фотографии. С помощью Raspberry Pi ты будешь делать кадры в режиме замедленной и ускоренной съёмки. Используя инфракрасный светодиод, реле и модуль инфракрасной камеры, ты соберёшь веб-камеру, которая с помощью мобильного телефона позволит тебе наблюдать за животными в ночном саду. Цифровые изображения могут быть обработаны с помощью *Python Imaging Library* (PIL). Ты создашь программу, которая сможет распознавать цветовой код на контрольной перфокарте.

Модуль видеокамеры

Существуют два модуля для Raspberry Pi – обычный и инфракрасный. Первый содержит фильтр для инфракрасного излучения (сокращённо ИК-лучи) и выдаёт изображение, как любая камера мобильного телефона. Второй модуль инфракрасным фильтром не оснащён, потому и называется NoIR (*noIR* = неинфракрасный). Это название является своего рода игрой слов, потому что «noir» на французском означает «чёрный» или «тёмный» и с его помощью можно делать снимки в ночное время. Впрочем, использовать модуль при дневном свете также не возбраняется. Изображения выходят такими же чёткими, но цветопередача немного отличается, поскольку, кроме видимого света, датчик воспринимает также и инфракрасное излучение. Если реальные цвета для тебя не столь важны и при этом ты лю-

бишь экспериментировать, то вместо обычного модуля видеокамеры купи себе модуль NoIR.

Камера Raspberry Pi по размеру не больше горошины и крепится на квадратную плату размером с почтовую марку. Обычно поверх линзы располагается маленькая пластиковая полоска для защиты, которую тебе перед использованием необходимо снять. Камера чувствительна к статическому электричеству. Прежде чем брать камеру в руки, чтобы не вывести её из строя, тебе стоит избавиться от статического электричества. Для этого ненадолго возьмись за водопроводный кран или батарею парового отопления. Водопровод и система парового отопления металлические (а металл – это проводник) и имеют контакт с землёй.

Для соединения с компьютером плата модуля оснащена 15-жильным шлейфом. Ты этот шлейф должен подключить к разъёму *Camera Serial Interface* (CSI) материнской платы Raspberry Pi. Разъём CSI находится рядом с выходом HDMI. Чтобы не произошло путаницы, CSI-разъём помечен словом *CAMERA* (рис. 17.1). У новых моделей Raspberry Pi этот разъём защищён пластиковой полосой, которую ты должен снять. Потом осторожно потяни вверх до упора пластиковую скобу, пока она полностью не освободится. Следи за тем, чтобы синяя сторона алюминиевой фольги была обращена к разъёму Ethernet, а после вставь кабель в этот слот. После этого вдави чёрную скобу обратно на место. В результате этого кабель оказывается зажатым и плотно зафиксированным (рис. 17.1 справа). Чтобы отсоединить его, нужно, конечно же, сначала вновь вытащить пластиковую скобу.

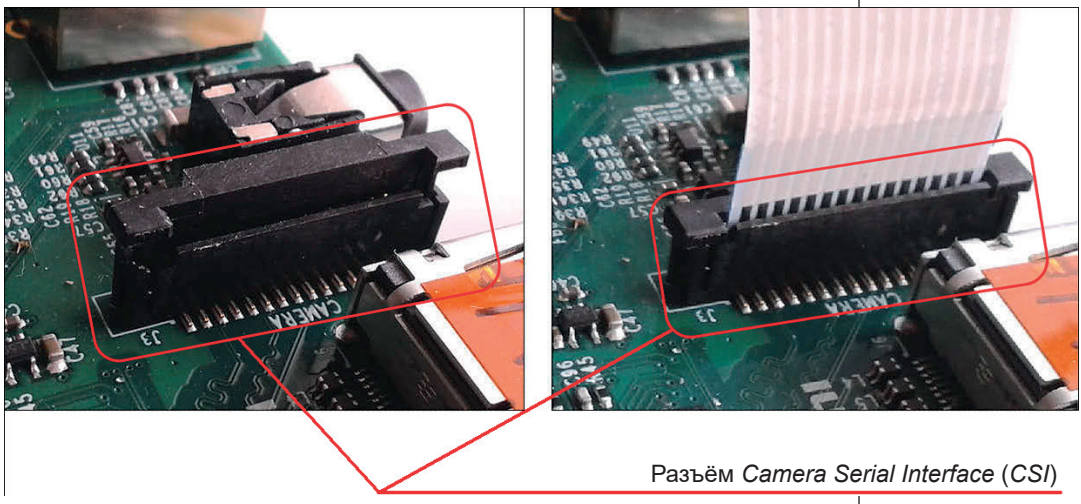


Рис. 17.1. *Подключение модуля камеры к разъёму CSI*

Точно так же ты можешь вытащить кабель из платы камеры (в случае если ты хочешь использовать более длинный шнур). Сначала вытаскиваешь до упора чёрную скобу и только затем кабель.

Тестируем модуль камеры

Перед использованием камеру необходимо активировать. В левом верхнем углу нажми кнопку главного меню и выбери команду меню **Параметры** ⇨ **Raspberry-Pi** ⇨ **Configuration**. Выбери вкладку **Interfaces** (Интерфейсы) и в правой части строки **Камера** (Camera) щёлкни мышью на кнопке **Enable** (Активировать). Далее нажми кнопку **OK** и перезапусти Raspberry Pi.

Теперь проверь, всё ли работает:

```
$ raspistill -v -o test.jpg
```

Если всё в порядке, твой дисплей в течение пяти секунд будет демонстрировать картинку камеры (функция **Preview** (Предварительный просмотр)). Затем автоматически будет сделана фотография и сохранена в файле *test.jpg*. Опция `-v` включает режим *verbose mode*.То есть после каждого снимка ты будешь получать набор технической информации.

Примечание

Функция предварительного просмотра не работает, если ты управляешь Raspberry с другого компьютера через соединение VNC (см. главу 2). Это значит, что хотя команда `raspistill` выполняется и сохраняет файл изображения, но в окне VNC в режиме live ты не увидишь картинку камеры.

Теперь некоторые технические данные о камере. Максимальное разрешение стоп-кадра – 2592×1944 пикселей в соотношении 4:3. Разрешение видео – 1080 пикселей при частоте 30 кадров в секунду. При выборе более низкого разрешения можно установить более высокую частоту кадров. Например, при разрешении 640×480 пикселей и частоте 90 кадров в секунду вполне реально даже делать замедленную съёмку.

Камера делает резкие снимки, только начиная с расстояния не менее 50 см. Крошечный объектив крепится с помощью

небольшого кусочка клейкой ленты. Для съёмки крупным планом следует изменить фокусное расстояние объектива. Для этого тебе необходимо с помощью иглы отсоединить эту клейкую ленту и изменить фокусное расстояние объектива. Правда, в этом случае гарантия на модуль камеры аннулируется, поэтому лучше ничего не трогать.

Программное обеспечение камеры

Для модуля камеры существует две программы. Одна программа обеспечивает покадровую съёмку, другая – позволяет снимать видео:

- `raspistill` выполняет покадровую съёмку и сохраняет снимки в различных форматах. Самый маленький размер файла с изображением обеспечивает алгоритм сжатия *jpeg*. Этот режим выбран по умолчанию;
- `raspivid` снимает видео (кадровая частота 2–90 изображений в секунду) и сохраняет в формате H.264.

Обе программы запускаются из командной строки. Есть большое количество опций, с помощью которых можно управлять обработкой изображений. В этом разделе представлены самые важные из них.

Каждую опцию в командную строку можно ввести в расширенном виде или в краткой форме. При вводе команды в расширенном виде команда начинается двух знаков «минус». При вводе в краткой форме команда начинается с одного минуса, например `-verbose` или `-v`.

Многие опции могут быть одинаково использованы как для функции `raspistill`, так и для функции `raspivid` (рис. 17.1).

Создаём покадровые изображения с помощью функции raspistill

Пять секунд – это же так долго! Теперь ты создашь фото за более короткий срок. Время ожидания выводится в миллисекундах после `-t`:

```
$ raspistill -t 2000 -o bild.jpg
```

Если ввести другой аргумент, например `-n` (или `--nopreview`), ты получишь фото без режима предварительного просмотра:

```
$ raspistill -o bild.jpg -n
```

Камера умеет делать и огромные фотоснимки (2592×1944 пикселей), но сама запись файла займёт много времени. Сохрани изображение в низком разрешении и понаблюдай, насколько быстро будет создан файл. Чтобы самостоятельно определить высоту и ширину создаваемого фотоснимка, с помощью аргумента `-w` задай нужное значение ширины, а аргументом `-h` – высоту изображения. Обрати внимание, высота и ширина снимка задаются в пикселях (пиксель – это отдельная точка, для которой определены значения яркости, насыщенности и цвета). Изображение состоит из отдельных пикселей (точек):

```
$ raspistill-w 499 -h 300 -obild.jpg
```

Возможно, ты захочешь сделать свой автопортрет, фотографируя своё отображение в зеркале. В этом случае сфотографированное изображение будет зеркальным. С помощью аргумента `-fv` ты можешь выполнить зеркальный поворот ранее полученного изображения. Сокращение *fv* образовано от «flipvertical». Это английское обозначение, буквальный перевод которого – «зеркало на вертикальной оси». То есть зеркальное отображение производится в горизонтальной плоскости относительно вертикальной оси.

Чтобы зеркальное изображение преобразовать в обычное (отобразить изображение по горизонтали), введи следующую команду:

```
$ raspistill -t 500 -fv
```

Аргумент `-fh` позволяет зеркально отображать изображение относительно горизонтальной оси. Это необходимо, если камера у тебя установлена на голове. В этом случае тебе нужно будет зеркально отображать изображение как по вертикали, так и по горизонтали.

```
$ raspistill -t 500 -fv -fh
```

Дальнейшие аргументы, с помощью которых ты можешь преобразовать вызов функции `raspistill`, ты найдёшь в этой таблице. Пробуй!

Таблица 17.1. Список нескольких опций для функций raspistill и raspistill

Опция	Значение	Пояснение
--brightness, -br	0–100	Яркость; 0 – это чёрный, 100 – белый. По умолчанию настроен на 50
--contrast, -co	–100...100	Контраст изображения, по умолчанию 07
--nopreview, -n		Без режима предварительного просмотра
--output, -o	Путь или –	Путь, по которому будет сохранено записанное изображение или видео. Если путь не указан, то сохранения не происходит. Если путь есть, то данные записываются в стандартный выход
--preview, -p	x, y, w, h	Расположение окна через указание позиции левого верхнего угла x, y, ширины w и высоты h

Таблица 17.2. Список опций для функции raspistill

Опция	Значение	Пояснение
--encoding, -e	jpg, bmp, gif, png	Формат данных для сохранённых файлов изображения; по умолчанию – jpg
--fliph, -fh		Отражение горизонтальное (вертикальное)
--flipv, -fv		Вертикальное (горизонтальное) отражение
--height, -h		Высота изображения (в пикселях); по умолчанию – 1944
--timeout, -t	Время в мс	Время ожидания до выполнения фото; по умолчанию – 5 с
--width, -w		Ширина изображения (в пикселях); по умолчанию – 2592

Модуль PIL

Чаще всего цифровые фотографии хранят в формате JPEG. Чтобы получить из файла формата JPEG объект PhotoImage, тебе понадобится такой инструмент, как *Python Imaging Library* (PIL).В настоящее время разработана версия PIL и для Python 3. Она есть и у тебя на твоём Raspberry Pi. Но это приложение на твоём Raspberry Pi есть в виде неустановленных пакетов. Чтобы этим приложением ты смог воспользоваться, выполни в командной строке терминала команду

```
$ sudo apt-get install python3-pil.imagetk
```

С помощью библиотеки PIL ты можешь обрабатывать изображения, сохранённые в форматах JPEG или PNG. Для этого вначале следует создать новый объект класса `Image`:

```
bild = Image.open(pfad)
```

Затем ты редактируешь объект с изображением. Наконец, ты из объекта `Image` создаешь объект `PhotoImage`, который можно отображать на виджете Tkinter. Подробнее об этом ты прочитаешь далее в примерах программы.

Проект 49. Распознаём движение

Что вообще происходит в саду или твоей комнате, пока тебя там нет? В этом проекте мы с тобой разработаем камеру, которая будет наблюдать за территорией и делать фото движущихся объектов. Кроме того, в левом верхнем углу цифровых фотографий будет указан временной промежуток, в течение которого было выполнено фото.

Рис. 17.2. *Фото с указанием времени происходящего*

В этом проекте компьютер должен обрабатывать цифровые фото для распознавания изменений, снятых камерой. Поэтому мы будем программировать с помощью *Python Imaging Library* (PIL). Каким образом можно засечь движение? Идея заключается в следующем. Компьютер в течение короткого промежутка времени делает маленькие (80×60) снимки. После каждого нового снимка компьютер попиксельно сравнивает полученное изображение со сделанным ранее. Значительные отличия указывают на то, что в кадре

появились изменения. В этом случае выполняются изменения в формате с большим разрешением и дополняются текстом.

Для сравнения двух изображений важны две константы (#3, #4):

- THRESHOLD (порог): это значение определяет, насколько сильно должны различаться два пикселя, чтобы считать их разными. В результате колебаний света пиксели двух изображений могут немного изменить свой цвет, находясь в одной и той же позиции, даже если никаких движений зафиксировано не было. Кстати, программа сравнивает только оттенки зелёного цвета пикселей (строка #8). Так процесс сравнения будет проходить быстрее;
- MIN: эта константа определяет количество пикселей двух последовательных изображений, которые должны быть разными. Если значение MIN слишком маленькое, тогда малые объекты (например, мухи) не будут распознаны, поскольку при движении изменяется недостаточное количество пикселей.

Пишем программу

```
#! /usr/bin/python3
# movement.py
import os, time
from PIL import Image, ImageDraw                              #1

FOLDER = '/home/pi/Python-projects/garten/'                   #2
COMMAND ='raspistill -t 300 -w %i -h %i -n -o
THRESHOLD = 40                                                #3
MIN = 20                                                      #4

def difference(image1, image2):
    '''Возвращает количество различных пикселей'''
    different_pixel = 0                                       #5
    b1 = image1.load()
    b2 = image2.load()
    width, height = old_image.size                            #6
    for x in range(width):
        for y in range(height):
            pixel1 = b1[x, y]
            pixel2 = b2[x, y]                                 #7
            d = abs(pixel1[1] - pixel2[1])                    #8
            if d >THRESHOLD:
                various_child_pixel += 1                      #9
    return various_child_pixel
# Main_program
```

```
os.system(COMMAND%(80, 60, FOLDER + 'image.jpg'))
time.sleep(0.5)
bild = Image.open(FOLDER + 'image.jpg')                    #10
i = 0                                                      #11
while True:
    altes_image = image
    os.system(COMMAND %(80, 60, FOLDER + 'image.jpg'))     #12
    image = Image.open(FOLDER + 'image.jpg')
    if difference(image, old_image) > MIN:                 #13
      file_name = 'image _' + str(i) +'.jpg'               #14
      i += 1
      os.system(COMMAND % (400, 300, FOLDER + file_name))  #15
      foto = Image.open(FOLDER + file_name)                #16
      foto_with_time = ImageDraw.Draw(foto)                #17
      foto_with_time.text((0, 0), time.asctime())          #18
      foto.save(FOLDER + file_name)                        #19
      time.sleep(3)
```

Как это работает?

#1 Здесь подмодули импортируются из модуля PIL.

#2 В этой папке сохраняются все изображения. В следующей строке определяется образец с тремя элементами-заполнителями для вызова функции `raspistill`. Аргумент `-t 300`: 300 мс – это время задержки перед съёмкой, чтобы камера могла настроиться. Теперь разберёмся с аргументом `-w%i:`. После параметра `-w` стоит элемент-заполнитель. Перед форматированием элемент-заполнитель заменяется целым числом, определяющим ширину картинки. Аргумент `-h%i:` – параметр с элементом-заполнителем, определяющим высоту картинки. Аргумент `-n`: отключает режим предварительного просмотра. В аргументе `-o%i:` после параметра `-o` стоит заполнитель, он при форматировании будет заменён именем файла, под которым должен сохраниться снимок.

#3 Переменная `THRESHOLD` (порог) – это константа, определяющая разницу между сравниваемыми пикселями. Если при сравнении полученное значение будет меньше значения этой константы, значит, изменения не значительны. Если же при сравнении двух пикселей от предыдущего и текущего кадров значение будет больше значения константы, значит, разница значительна, и камера фиксирует движение.

#4 Переменная `MIN` определяет минимальное количество отличающихся друг от друга пикселей, показывающих, что два сравниваемых кадра отличаются друг от друга.

#5 В этой командной строке при сравнении двух кадров мы считаем количество отличающихся друг от друга пикселей. В начале цикла это количество равно нулю.

#6 Из объекта `PIL.Image` создаётся список пикселей для более быстрой обработки.

#7 В этих двух циклах идёт попиксельное сравнение изображений – пиксель за пикселем. Сначала сравниваются пиксели `b1[0.0]` и `b2[0.0]`, а потом пиксели `b1[0.1]` и `b2[0.1]` и т. д.

#8 Переменная `d` – это разница зелёных оттенков обоих пикселей, которые сейчас сравниваются. Функция `abs()` отвечает за то, что значение переменной `d` всегда будет положительным.

#9 Если значение переменной `d` больше значения переменной `THRESHOLD` (порог), значит, с помощью строки `various_kind_pixel += 1` определяется место, в котором между двумя кадрами наблюдается изменение.

#10 Идёт загрузка только что сделанного фото.

#11 Переменная `I` содержит номер следующего изображения, которое нужно сохранить. Этот номер используется для создания имени файла. У первого фотоснимка номер 0.

#12 Маленькая картинка (80×60) сохраняется в папке для изображений под именем *image.jpg*. С помощью оператора форматирования заполнители `%` заменяются в переменной `COMMAND` двумя числами и одной строкой символов.

#13 В случае значительного отличия предыдущей картинки от текущей выполняется и сохраняется снимок большего размера. Различия находятся в следующем (вложенном) блоке инструкций.

#14 Сначала создаётся имя файла для фото. При этом большую роль играет значение переменной `i`. У первой картинки имя *image.jpg*. Потом значение переменной `i` увеличивается на единицу.

#15 Выполняется и сохраняется большое фото (400×300).

#16 Считывается файл картинки, а потом создаётся Image-объект `фото`.

#17 Появляется объект `ImageDraw`. Создаётся Draw-объект `foto_с_указанием_времени`. Этот объект позволит вставить в изображение нужную нам надпись.

#18 В левом верхнем углу фото (позиция (0.0)) записывается отметка времени (дата и время).

#19 Фото сохраняется второй раз (уже с отметкой времени).

Проект 50. Покадровая замедленная съёмка

При такой съёмке скорость отображения медленных событий увеличивается. Это делается, чтобы можно было уловить происходящие с объектом съёмки изменения: как растёт лист салата, раскрывается цветочный бутон, ползёт улитка, исчезает лужа или поднимаются дрожжи.

Рис. 17.3. *Несколько снимков в режиме замедленной съёмки, показывающих изменения в росте дрожжей. В верхней части стакана можно увидеть красный отсвет модуля камеры. Если тебе мешает это «всевидящее око», то можешь накрыть камеру кусочком клейкой ленты*

Провести этот эксперимент с дрожжами ты можешь так.

- Насыпь в стакан ложку сухих дрожжей, ложку сахара и добавь туда немного воды. Тщательно всё перемешай и поставь перед камерой.
- Открой в Raspberry окно LXTerminal, перейди в папку проекта и создай там каталог для своего фильма:

```
$ mkdir yeast
```

- Затем перейди в новую папку:

```
$ cd yeast
```

- Используя следующую команду, в течение двух часов с интервалом в 30 с создай серию снимков и сохрани их под именем *image0001.jpg*, *image0002.jpg* и т. д.

```
$ raspistill -o image%04d.jpg -tl 30000 -t 7200000
```

У тебя есть мысли по поводу того, что означают эти параметры? В табл. 17.3 ты найдёшь пояснения.

Таблица 17.3. Список опций для режима замедленной съёмки, выполняемой с помощью `raspistill`

Параметр	Пояснение
`-oimage%04.jpg`	Изображение сохранилось под именем, состоящим из трёх частей: `image`, автоматического четырёхразрядного числа и разрешения `jpg`
`-t l 30000`	Время интервала между двумя снимками составляет 30 000 мс = 30 с
`-t 7200000`	В общей сложности фото выполняется в течение 72 000 000 мс. Это 7200 с, или 2 ч

Теперь тебе нужно собрать все изображения в единый видеофайл. Сначала установи программное обеспечение для обработки аудиовизуальных данных:

```
$ sudo apt-get install libav-tools
```

Следующая команда должна выполняться в каталоге, в котором находятся твои раздельные снимки. Команда создаёт фильм в формате MP4 с размером кадра 800×600. Здесь команда разделяется на несколько строк. В своём окне Terminal ты должен записать *одной строкой*:

```
$ avconv -r 10 -i bild%04d.jpg -r 10 -vcodec libx264
         -crf 20 -g 15 -vf scale=800:600 lapse _1.mp4
```

Видеоформат не должен быть большим, иначе Raspberry будет перегружен. Сама загрузка длится достаточно долго и может быть прервана. При формате 800×600 Raspberry требуется несколько секунд на снимок.

Если тебе нужен фильм с кадром размером, скажем, 1280×720, то ты можешь использовать лишь фрагмент из оригинальных изображений. Вызови программу, используя дополнительную опцию:

```
$ avconv -r 30 -i bild%04d.jpg -r 30 -vcodec libx264 -crf 20
         -g 15 -vf crop=2592:1458,scale=1280:720 lapse_1.mp4
```

В конце команды стоит название файла, в котором нужно сохранить видео. Таблица 17.4 объяснит остальные опции.

Таблица 17.4. Список нескольких опций для программы avconv

Опция	Пояснение
`-r 30`	Количество кадров в секунду (кадровая частота). Частота должна быть указана дважды. Первый раз для ввода файла, а второй – для вывода видео. В противном случае есть риск, что похожие снимки будут пропущены
`-I картинка%04.jpg`	Образец имени для файла картинки, которую нужно обработать (inputfiles). У нас это: *image0001.jpg*, *image0002.jpg* и т. д.
`-vcodec lib×264`	Выбор видеокодирования `lib×264`
`-crf 20`	Настройка устойчивого качества изображения
`-g 15`	Размер группы изображений устанавливается на отметке 15. При видеоформате MP4 для экономии пространства всегда кодируются группы нескольких изображений
`-vf crop=2592:1458.`	Видеофильтр. Используя `crop` = *ширина*:*высота*, выбирается фрагмент изображения.
`scale=1280:720`	Используя `scale` = *ширина*:*высота*, настраивается формат фильма

С помощью уже установленного на Raspberry плеера OMX ты можешь просматривать видео в режиме замедленной съёмки (как и другие отснятые с помощью `raspivid` видеофайлы):

```
$ omxplayer lapse_1.mp4
```

Используя опцию `--fps`, ты можешь настроить скорость (количество кадров в секунду):

```
$ omxplayer -fps 25 lapse_1.mp4
```

Проект 51. «Цветной ключ»

Внимание, контроль! Незнакомец кладёт лист бумаги с шестью цветными квадратиками под камеру (расстояние от камеры до листа минимум полметра). Это его ID-карта. На экране видно, как он двигает листок туда-сюда продолжительное время, пока цветная поверхность не окажется внутри рамки. Затем он нажимает кнопку **Check**. Компьютер проверяет изображение, распознаёт цветную комбинацию и приветствует человека по имени, которое было

сохранено. В дополнение к этому он может ещё включить привод, открывающий дверь или другое устройство. Но этот момент мы пропустили в программе, чтобы сделать процесс короче.

Основная идея программы заключается в следующем. На снимке камеры цвет пикселя определяется примерно в центре каждого из шести квадратов. У каждого из этих шести пикселей проверятся RGB-значения (аббревиатура от *red*, *green*, *blue* – «красный», «зелёный», «синий») и определяется собственно цвет. Мы допускаем только четыре цвета (чтобы не возникло сложностей) и кодируем их буквами: *красный* – red (`r`), *зелёный* – green (`g`), *синий* – blue (`b`) и *чёрный* – black (`s`). Обрати внимание: хоть чёрный цвет на английском – это black, чтобы не было повторения букв, мы этот цвет обозначим буквой `s`. Каждая ID-карта цвета соответствует слову из шести букв, например `rgbbsr` для карты, как показано на рис. 17.4. В словаре сохраняются несколько комбинаций цвета и присваиваются именам собственным, например `'rgbbsr': 'David'`. Только те люди, чьи имена сохранены в словаре, имеют так называемое «разрешение на доступ».

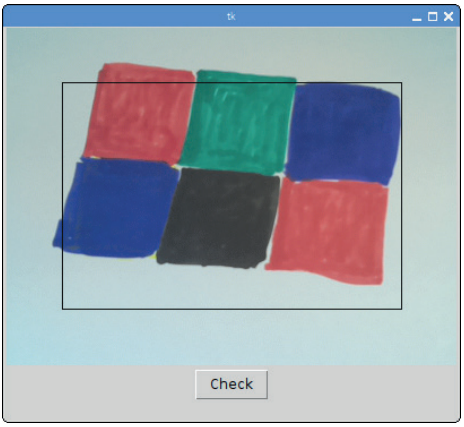


Рис. 17.4. *Графический пользовательский интерфейс шкалы оттенков*

```
>>> ================================ RESTART ======
>>>
(194, 55, 32)
(24, 135, 66)
(51, 50, 90)
(59, 57, 94)
(74, 58, 22)
(212, 82, 48)
```

Рис. 17.5. *Распознанные RGB-значения для отсканированных полей цвета (красный, зелёный, синий, синий, чёрный, красный)*

Пишем программу

```
#! /usr/bin/python3
# colour_lock.py
import os, time
from tkinter import Tk, Label, Button, Canvas, NW           #1
from PIL import Image, ImageTk
BBOX = (200, 150, 600, 450)                                 #2
PEOPLE = {'rgbbsr':'Вася','rrbbsg':'Аня','sgbrrb':'Гриша'}  #3

def display_image():
    '''Показать текущее изображение камеры'''
    global image, foto                                      #4
    COMMAND='raspistill -t 500 -w 800 -h 600 -o image.jpg -n'
    os.system(command)
    total_image = Image.open('image.jpg')                   #5
    middle = total_image.crop(BBOX)                         #6
    foto = ImageTk.PhotoImage(middle)                       #7
    canvas.create_image(0, 0, anchor=NW, image=foto)        #8
    canvas.create_rectangle(50, 50, 350, 250)               #9
    window.after(500, show_image)                           #10

def farbe(x, y):
    '''Цвет точки x, y'''
    r, g, b = image.getpixel((x, y))                        #11
    print(r, g, b)                                          #12
    if r > 0.95 * (g + b):
        return 'r'                                          #13
    if g > 0.8 * (r + b):
        return 'g'
    if b > 0.7 * (r + g):
        return 'b'
    else:
        return 's'

def check():
    colors = color(100, 100) + color(200, 100) \
           + color(300, 100) + color(100, 200) \
           + color(200, 200) + color(300, 200)              #14
    if colors in PEOPLE.keys():                             #15
        text = colors + ' ------ Здравствуй, ' \
               + PEOPLE[colors] + '!'
    else:
        text = colors + ' ------ неизвестный'
    label.config(text=text)

# Основная программа
window=Tk()
window=Tk()
canvas = Canvas(master=windows, width=400, height=300)
canvas.pack()
```

```
button = Button(master=window, command=check,text='Check')
button.pack()
label = Label(master=window, text='')
label.pack()
show_image()
window.mainloop()
```

Как это работает?

#1 Внимание! Функции `PIL` и `tkinter` содержат класс `Image`. Поэтому при импортировании с помощью `from` здесь используется не `*` («звёздочка»), а в целях безопасности – только необходимые имена.

#2 Переменная `BBOX` – это bounding-box (ограничивающая рамка). Она даёт описание длины и общего размера прямоугольного фрагмента изображения.

#3 Переменная `PEOPLE`. Словарь даёт имя комбинации цвета, описанной шестью буквами, и связывает каждую цветовую комбинацию с именем конкретного человека.

#4 Переменные `image` и `foto` должны изменяться, поэтому они становятся глобальными.

#5 В файле *image.jpg* только что сохранилось изображение камеры. Далее этот файл считывается и создаётся объект `PIL.Image`.

#6 Здесь есть один важный совет. Камера должна находиться на сравнительном отдалении от листка бумаги, чтобы сделать изображение чётким. С близкого расстояния это сделать, к сожалению, невозможно (если только не изменить фокусное расстояние в настройках самой камеры). Тогда цветные поверхности в центре становятся маленькими и вокруг них можно увидеть подложку, на которой лежит лист бумаги с нарисованными цветными квадратиками (например, стол, на котором лежит бумага). Но нам нужна только центральная часть цветной поверхности, а не окружающие предметы. Поэтому здесь мы вырежем прямоугольник из середины картинки с помощью метода `crop()`.

#7 Из объекта `PIL.Image` создаётся `Tkinter.PhotoImage` и присваивается переменной `foto`. Такой объект (*а не* `PIL.Image`) может быть представлен в окне приложения.

#8 На Canvas-объекте `canvas` отображается картинка.

#9 На одном Canvas рисуется прямоугольник. Левый верхний угол находится в позиции (50, 50), правый нижний угол – в позиции (350, 250).

#10 Через полсекунды функция вызывается сама (бесконечная рекурсия). Таким образом, показанное изображение будет постоянно обновляться.

#11 Вызов функции `getpixel((x, y))` возвращает кортеж из трёх цифр между 0 и 255. Они дают часть оттенков красного, зелёного и синего для пикселя в указанном месте (x, y).

#12 Эта строка важна только для разработки программы. Если программа работает хорошо, ты эту строку можешь удалить. RGB-значения пикселей выводятся в окне Shel в виде цифр (рис. 17.5).

#13 Здесь находится «интеллект» программы. RGB-значения определяют, о каком цвете идёт речь. Если, например, оттенок красного (первая цифра) превышает все остальные, тогда, скорее всего, будет выбран красный. Значит, возвращается буква `'r'`. Если доминирует зелёный оттенок, значит, пиксель зелёный и возвращается буква `'g'`. Обрати внимание, что выполнение программы после функции `return` будет прервано. Почему именно эти коэффициенты: 0.95, 0.8 и 0.7? Я выполнил ряд тестов, просмотрел значения цветов и заметил, что в красном цвете оттенки выражены сильнее, чем, например, синий оттенок на синем.

#14 Здесь появляется последовательность из шести букв. Каждый вызов функции `color()` возвращает одну букву. А эти буквы прикрепляются друг за другом знаком +.

#15 Если строка символов `colors` встречается в ключах словаря, то система распознаёт эту комбинацию цвета и приветствует владельца ID-карты по имени. Его имя в словаре связано с цветом.

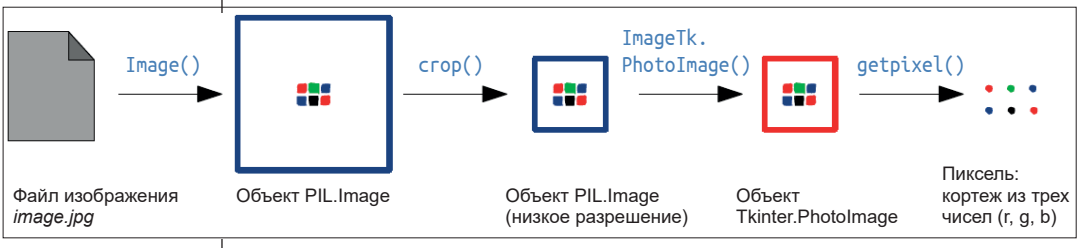


Рис. 17.6. *От файла изображения до пикселя: этапы обработки*

Расширения

Распознавание цветов (#13) может быть запрограммировано ещё на интеллектуальном уровне. Например, сюда мож-

но включить значение яркости (сумма всех компонентов цвета), тем самым сделав чёрный цвет более отличимым от других. Анализ изображений – важная область информатики, имя которой – компьютерная визуализация. С помощью этой техники возможно распознавание предметов, фруктов или лиц.

Вопросы

1. Как ты узнаешь, что камера в настоящий момент активна и снимает фото?
2. Какой элемент отсутствует у модуля камеры NoIR?
3. Почему изображение, снятое с помощью функции `raspistill`, хуже, если время ожидания равно нулю (опция `-t`)?
4. У функции `raspivid` опция `-t` имеет иное значение, нежели у функции `raspistill`. Поясни, почему!
5. Как можно настроить скорость проигрывания видеофайла, открытого в плеере `Omxplayer`?

Задания

Задание 1. Замедленная съёмка с использованием управляемого освещения

Прежде чем яблоко начнёт портиться, а семена кресс-салата прорастут и взойдут, пройдет несколько дней. Чтобы эти изменения были хорошо заметны, можно делать снимки каждые 30 мин. В течение недели это будет $2 \times 24 \times 7 = 336$ изображений. Основной проблемой является то, что при съёмке каждого кадра должно быть примерно одно и то же освещение. Лучше всего, если ты создашь маленькую фотостудию – из бумаги в качестве фона и маленького источника света. Для этой маленькой студии достаточно будет белого светодиода, управляемого через GPIO. Подключи светодиод к GPIO Raspberry. Соедини через резистор номиналом 68 Ом анод (длинный вывод) с контактом 1, а катод (короткий вывод) соедини с контактом 10. Схему подключения светодиода ты найдёшь на рис. 8.8 главы 8. Обрати внимание: на этом рисунке номинал сопротивления 130 Ом. Сейчас тебе понадобится сопротивление номиналом 68 Ом.

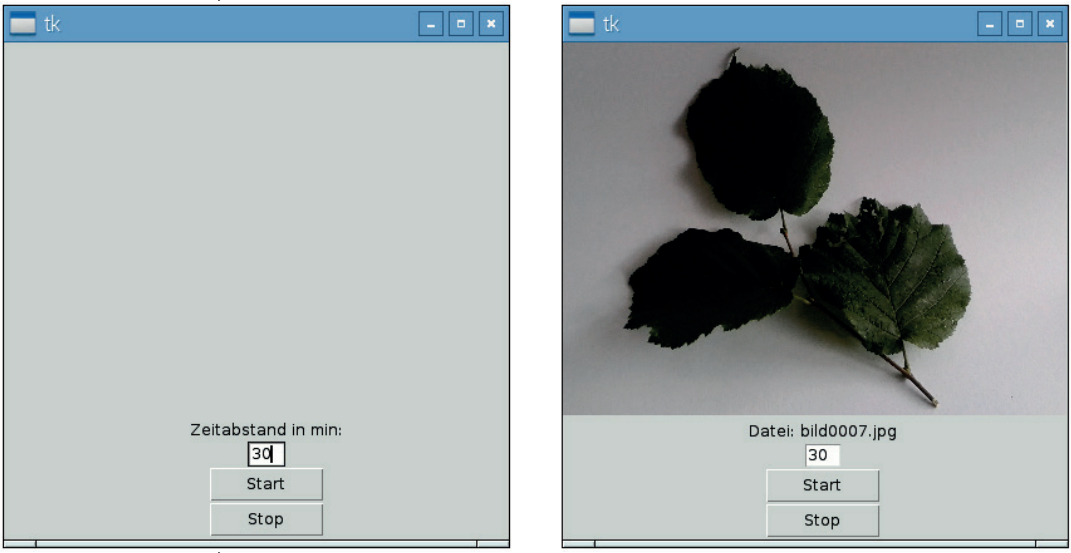


Рис. 17.7. *Использование снимков, сделанных в режиме замедленной съёмки*

Разработай программу с пользовательским интерфейсом, как показано на рис. 17.7. Следует предусмотреть возможность настройки пользователем временного интервала между двумя снимками (в минутах). Чтобы начать фотосъёмку, следует нажать кнопку **Пуск**. Перед съёмкой каждого кадра ненадолго включается светодиод, а после съёмки выключается. Для завершения работы программы следует нажать кнопку **Стоп**.

Структура программы

```
#! /usr/bin/python3

def start():
    '''Начало съемочной серии '''
    global stop
    stop = False
    ...

def stop():
    '''Завершение съёмки'''
    global stop
    stop = True
    ...

def run(time):
```

```
    '''Фотосъемка'''
    ...
    while not stop:
        ...

def init_GPIO():
    ...

# Основная программа
...
```

Задание 2. Следы движения

Компьютеры способны автоматически определять движение и фиксировать его. Это полезная функция, если хочешь, к примеру, изучить маршрут движения или характер колебания упругих предметов.

В каком направлении движутся муравьи, и каких мест они избегают? Где на рыночной площади чаще всего происходят события, а где не происходит вообще ничего? В каких местах звери переходят дорогу? Если попытаешься посидеть спокойно, какие части тела все равно останутся подвижными? Разработай программу, позволяющую вести наблюдения за движением и регистрирующую их. Пользовательский интерфейс должен выглядеть как на рис. 17.8. В окне приложения ты увидишь живую картинку камеры. Примерно каждые две секунды будет выполняться съёмка изображения. Места, где происходило какое-то движение, будут помечаться красной точкой. Со временем все места с происшествиями будут окрашены в красный цвет.

Рис. 17.8. *Профиль движения палисадника*

На рис. 17.8 представлен обзор палисадника. Несколько зон окрашено красным цветом, поскольку ветви и листва на ветру приходят в движение. Чем интенсивнее красный цвет в определённое время, тем выше скорость ветра.

Ответы на вопросы

1. Если камера активна, загорается красный светодиод.
2. Здесь отсутствует инфракрасный фильтр, чтобы камера (наряду с видимым светом) также могла принимать инфракрасное излучение.
3. Автоматический режим камеры не имеет времени для оптимальной настройки изображения.
4. Обычно у функции `raspistill` происходит задержка перед съёмкой. А в режиме замедленной съёмки (если установлена `-tl`) у функций `raspivid` и `raspistill` параметр `-t` указывает общую продолжительность записи.
5. С помощью параметра `-fps` можно настроить скорость (количество кадров в секунду).

Решение задач

Задание 1. Режим замедленной съёмки с управляемым освещением

Эта программа использует **PIL**, поэтому вместо Python 3 её нужно разрабатывать в Python 2.7. Кроме того, должен быть доступ к GPIO и наличие управляемого светодиода, поэтому здесь потребуются права администратора. Запусти IDLE (не IDLE 3) из командной строки:

```
$ sudo idle
```

Пишем программу

```
#! /usr/bin/python3
# lapse.py

import os, time
from PIL import Image, ImageTk
from tkinter import Canvas, Tk, Button
from tkinter import NW, Label, Entry
from thread import start_new_thread
from RPi import GPIO                                        #1
COMMAND= 'raspistill -t 500 -w 800 -h 600 -n -o '

def start():
    global stop                                             #2
    time = int(time_input.get()) *60                        #3
```

```
    stop = False
    start_new_thread(run, (time,))                          #4

def stop():
    global stop
    label.config(text='Интервал времени в мин.:')           #5
    stop = True

def run(time):
    i = 0                                                   #6
    while not stop:                                         #7
        file_name = 'bild%04d.jpg' %i                       #8
        label.config(text='Datei: '+file_name)
        i += 1
        GPIO.output(10, False)                              #9
        os.system(COMMAND + file_name)                      #10
        GPIO.output(10, True)                               #11
        image = Image.open(file_name)                       #12
        image_center = image.crop((200, 150, 600, 450))
        image_center.save(file_name)                        #13
        image_tk = ImageTk.PhotoImage(image_center)         #14
        canvas.create_image(0, 0,
                            anchor=NW,image=image_tk)       #15
        time.sleep(time)

def init_GPIO():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(10, GPIO.OUT)
    GPIO.output(10, True)

# Основная программа
init_GPIO()
window = Tk()
canvas = Canvas(master=window,width=400, height=300)
canvas.pack()
label = Label(master=window,text='Интервал времени в мин.: ')
label.pack()
time input = Entry(master=window, width=3)
time input.pack()
Button(master=window, command=start,width=8, text='Пуск').pack()
Button(master=window, command=stop,width=8, text='Стоп').pack()
window.mainloop()
```

Как это работает?

#1 В следующей строке стоит образец для команды `raspistill`. Через полсекунды будет сделано изображение в формате 800×600 без режима предпросмотра. Не забудь поставить пробел после `-o`! Позже здесь будет добавлено имя файла, под которым сохранится изображение.

#2 Глобальная переменная `stop` управляет выполнением функции `run()`.

#3 Здесь выбирается строка символов из поля ввода, преобразовывается в целое число и из него вычисляется количество секунд между двумя снимками.

#4 Функция `run()` выполняется в новом потоке в виде параллельного процесса (*см.* раздел 16.5.2).

#5 Здесь опять записывается начальный текст в метке.

#6 Переменная `i` – это номер следующего фото.

#7 Переменная `stop` – это своего рода блокировка для камеры в режиме замедленной съёмки. Пока у переменной значение `False`, выполняется повтор, т. е. съёмка продолжается.

#8 Здесь создаётся имя файла: *image0000.jpg*, *image0001.jpg* и т. д.

#9 Включение светодиода.

#10 Выполняется съёмка фото и проигрывается под созданным именем.

#11 Выключение светодиода.

#12 Новое фото загружается в виде `PIL.Image`-объекта, а потом из центра копируется прямоугольник. Таким образом мы решаем проблему, когда камера не может сделать съёмку крупным планом. Если у тебя маленький объект, то, конечно, ты можешь выбрать фрагмент ещё меньше.

#13 Изображение сохраняется с центральным прямоугольником.

#14 На `PIL.Image`-объекте мы получаем `tkinter.PhotoImage`-объект…

#15 … который отображается на Canvas.

Больше света!

Для студии большего размера тебе понадобится больше света. Используй розетку с подключённым к ней прожектором и один передатчик. Как управлять розеткой с помощью передатчика, подробно говорилось в главе 15.

Задание 2. Следы движения

Пишем программу

```
#! /usr/bin/python3
# traces.py
```

```
import os, time
from PIL import Image, ImageTk
from tkinter import Canvas, Tk, Button, NW

COMMAND = 'raspistill -t 300 -w 400 -h 300 -n -o image.jpg'
THRESHOLD = 30                                                  #1

def run():
    global altes_image, image
    altes_image = image
    os.system(COMMAND)
    image = Image.open('image.jpg')
    b1 = altes_image.load()
    b2 = image.load()
    width, height = image.size
    for x in range(breite):                                     #2
        for y in range(height):
            pixel1 = b1[x, y]
            pixel2 = b2[x, y]
            d = abs(pixel1[1] - pixel2[1])
            if d >THRESHOLD:
              canvas.create_line(x, y,x, y+1,fill='red')        #3
    window.after(2000, run)                                     #4

# Основная программа
window = Tk()
canvas = Canvas(master=window,width=400, height=300)
canvas.pack()
os.system(COMMAND)
image = Image.open('image.jpg')
image_tk = ImageTk.PhotoImage(image)
canvas.create_image(0, 0, anchor=NW,image=image_tk)
run()
window.mainloop()
```

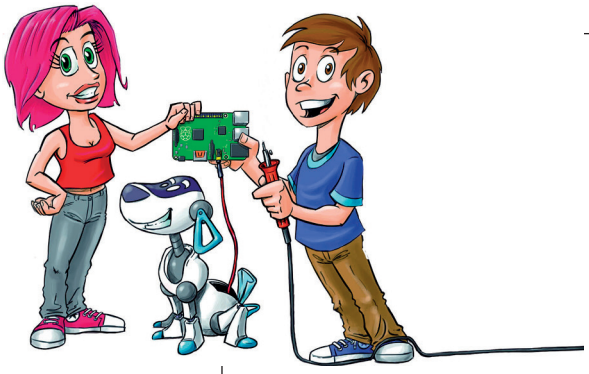
Как это работает?

#1 Переменная `THRESHOLD` (порог) – это константа, которая указывает, как сильно должны отличаться пиксели, чтобы считаться разными.

#2 В обоих циклах все пиксели обоих изображений (нового и старого снимков) сравниваются друг с другом.

#3 Если пиксели на предыдущем и текущем изображениях, которые находятся в одной и той же позиции, явно различаются, значит, в этом месте было какое-то движение, а где именно – покажет красная линия (точка).

#4 Функция вызывается самостоятельно через две секунды.



18

Raspberry Pi в качестве веб-сервера – всегда к вашим услугам

Сервер – это постоянно работающая и выполняющая задачи компьютерная система. Он обеспечивает доступ к веб-страницам через локальную сеть или интернет. Raspberry Pi потребляет меньше электроэнергии, поэтому он является идеальным аппаратным обеспечением. В этой главе ты узнаешь, как установить сервер Apache и использовать его в качестве веб-сервера. Речь пойдёт о динамических сайтах, отображающих время, живую картинку камеры или температуру. Кроме того, мы разработаем интерактивный сайт с кнопками и полями ввода, через которые можно управлять устройствами.

Как настроить Raspberry Pi в виде сервера?

Глобальная всемирная сеть www (world wide web) состоит из многих миллиардов веб-страниц, соединённых друг с другом ссылками. Ты уже знаешь, что ссылка (или гиперссылка) – это картинка или кусочек текста в виде ссылки, где ты можешь щёлкнуть мышью. Если навести указатель мыши на такую ссылку, курсор изменится и примет вид маленькой ладошки. Если же щёлкнуть мышью по такой ссылке, появится новое окно. Такую сеть из связанных страниц называют гипертекстом.

А что происходит, когда ты запрашиваешь адрес сайта в интернете? Твой браузер отправляет запрос на веб-сервер, работающий на удалённом компьютере в интернете. Этот сервер обрабатывает твой запрос и в качестве ответа посылает тебе запрошенную веб-страницу, которая впоследствии отобразится в браузере.

Пересылка веб-страницы и работа сервера регулируются через *Hyper Transfer Protocol* (HTTP). По-русски этот термин означает «протокол передачи гипертекста».

Для создания сайта существует особый формат, вид языка, именуемый HTML (от англ. *Hypertext Markup Language* – «язык гипертекстовой разметки»).

Если хочешь использовать Raspberry Pi в качестве сервера, то для начала тебе необходимо выполнить две вещи, которые я последовательно объясню. Итак:

- установи HTTP-сервер (например, Apache) и проследи, чтобы он постоянно работал;
- сохрани HTML-файлы в специальном каталоге, содержащем информацию о твоем веб-сервере.

Из этого раздела ты узнаешь, как это сделать.

Устанавливаем Apache-сервер

Apache – это, пожалуй, самый известный HTTP-сервер. Установи его, введя в окне LXTerminal следующую команду:

```
$ sudo apt-get install apache2
```

После установки программа сразу же запустится. Так после каждой перезагрузки RPi Apache будет автоматически работать в фоновом режиме. Работа программы в фоновом режиме не заметна. С этого момента твой Raspberry Pi становится веб-сервером.

Попробуй его в работе. Нажми в верхнем левом углу Рабочего стола и запусти браузер *Хромиум*. Введи в поле ввода адреса адрес http://localhost/index.html. Появится сайт, как показано на рис. 18.1. Сайт гласит: «Всё работает! Эта страница установлена по умолчанию для серверов. Программное обеспечение работает, но содержимое пока не добавлено».

Данные, которые ты вводишь в адресное поле браузера, называются *Uniform Resource Identifier* (единый идентификатор ресурсов), или, проще говоря, URI. Этот идентификатор веб-сайтов состоит из нескольких частей:

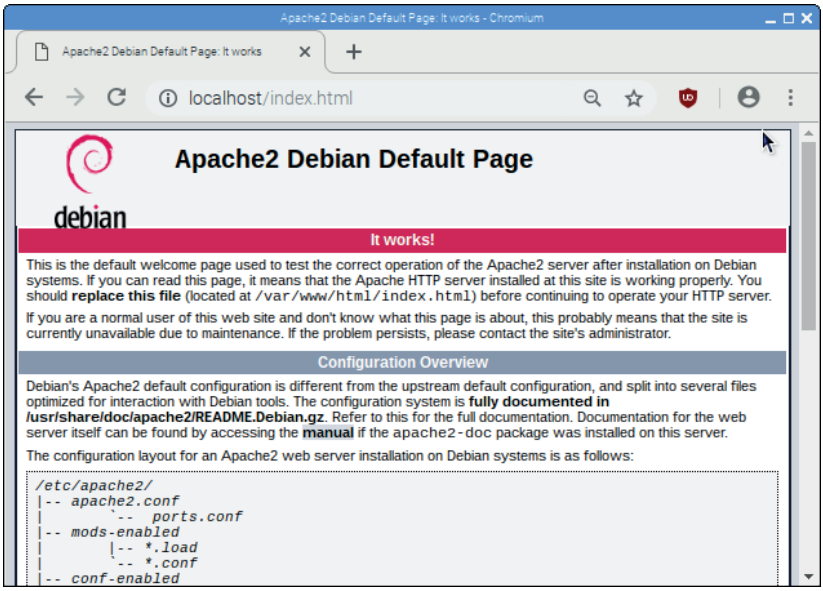


Рис. 18.1. *Так выглядит веб-страница для сервера Apache*

- **http://** – буквы представляют собой *гипертекстовый транспортный протокол*. Эта часть указывает на то, что это веб-сайт;
- имя сервера или IP-номер. Имя *localhost* всегда означает сервер, который работает на том же компьютере, что и браузер;
- каталог и имя (путь) файла.

Имя *index.html* имеет одну особенность. Его можно опустить. То есть ты можешь просто ввести в поле браузера *http://localhost/*, как показано на рис. 18.1. Страница *index.html* является стартовой для сайта, состоящего из нескольких отдельных страниц, которые соединены ссылками.

Но где же сохранён этот файл *index.html*? Программа установки Apache-сервера создала несколько новых каталогов. Самый важный из них – это каталог */var/www/html*. Там хранятся все HTML-страницы, которые отображает Apache-сервер. И именно там ты найдёшь (пока что единственный файл) *index.html*.

Собственная веб-страница в локальной сети

Замени установленную страницу Apache-сервера на свою собственную страницу! Веб-серверы служат для распространения информации. Если твой Raspberry Pi встроен в локальную компьютерную сеть, то любой пользователь

локальной сети (и никакой другой более) может читать твою страницу. Об этом чуть позже. Радиус действия локальной сети ограничен. Твой сайт *нельзя* увидеть в интернете.

На своей странице ты можешь создавать публикации, которые интересны людям твоего окружения: события, фото или актуальные новости. На рис. 18.2 наглядно показан простой пример с использованием заголовка, нескольких строк текста и гиперссылки, по которой для перехода нужно щёлкнуть мышью.

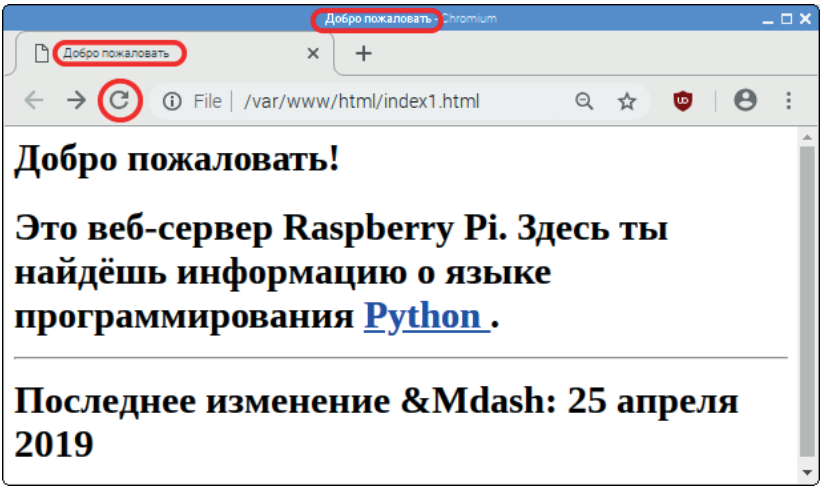


Рис. 18.2. *Новая домашняя страница, в которой красной рамкой выделен заголовок страницы*

Новый владелец

В каталоге */var/www/html* содержатся сайты, которые отображает Apache-сервер. В настоящий момент изменить каталог */var/www/html* и файл */var/www/html/index.html* может только администратор (`root`). Используя `ls`-команду, ты можешь это проверить:

```
$ ls -l /var/www/html
```

Здесь после `ls`-команды ещё добавлен *аргумент* `-l` (маленькая буква L). Это позволит немного изменить команду `ls`. Она возвращает дополнительную информацию о каталоге, который (в виде второго аргумента) находится за ней. При нажатии ↵ в следующей строке появится список всех файлов в каталоге */var/www/html*. Поскольку там сейчас находится всего один файл, список состоит лишь из одной-единственной строки:

```
-rw-r--r-- 1 root root 177 Nov 8 11:47 index.html
```

На рис. 18.3 объясняется значение этих букв и знаков. Обрати внимание: в ответе на твой запрос дата будет другой, показывающей текущее время.

В первой части описываются права на файл */var/www/html/index.html*. В целом существуют три вида прав на доступ к файлам:

- право чтения `r` (read = читать) означает, что содержимое файла (картинку или текст) можно прочитать или как-то отобразить. Изменять содержимое нельзя;
- право записи `w` (write = записывать) означает, что файл можно изменить или удалить;
- право выполнения `x` (execute = выполнять) означает, что файл можно выполнить. Эта функция, конечно же, имеет смысл только для программных файлов.

В каталогах права на доступ `r`, `w`, `x` имеют немного другой смысл:

- право чтения `r` (read = читать) означает, что содержимое каталога может быть перечислено;
- право записи `w` (write = записывать) указывает, что созданные в этом каталоге файлы могут быть изменены или удалены;
- право выполнения `x` (execute = выполнять) означает, что можно переместиться в каталог, но это не значит, что можно перечислить его содержимое.

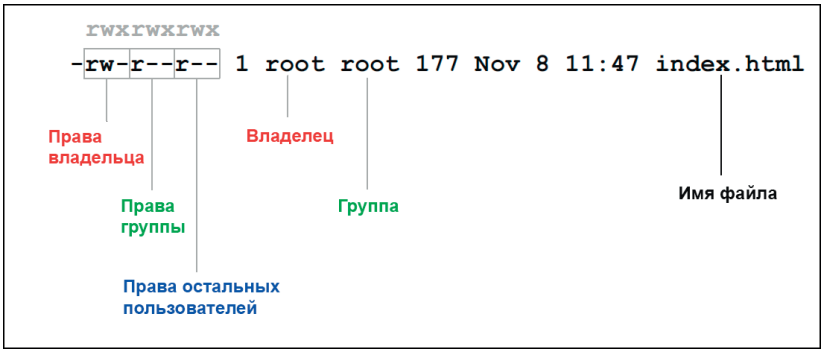


Рис. 18.3. *Право на доступ*

Существуют только три группы лиц, которым присвоено право доступа: владелец (единоличный пользователь), группа (несколько пользователей компьютером) и остальные пользователи.

Любому участнику этой группы присваивается буква `r`, `w` или `x`, с помощью которой пользователь получает право доступа. Если право предоставлено, тогда ставится соответствующая буква. Если нет – ставится черта.

У файла */var/www/html/index.html* имеется владелец с правом чтения и записи. Участники группы и прочие пользователи могут только читать.

Цифра по описанию прав на доступ (у нас это 1) – это количество ссылок в файловой системе на эту запись.

Первый обладатель `root` означает владельца файла. Это администратор. Второй `root` – это группа, к которой этот файл относится. Здесь волею случая группа также зовётся `root`.

Если ты, будучи пользователем `pi`, создашь сайт, тебе понадобятся права на изменение файла (права записи). Но у остальных такого права быть не должно. Поэтому будет лучше, если пользователь `pi` станет новым владельцем и каталога */var/www/html*, и файла */var/www/html/index.html*. С помощью Unix-команды `chown` (*change owner* = сменить владельца) ты можешь изменить отношения владения. Тебе нужно запустить эту команду от лица администратора (`sudo`):

```
$ sudo chown pi /var/www/html /var/www/html/index.html
```

Проверь, кто является владельцем файла:

```
$ ls -l /var/www/html
-rw-r--r-- 1 pi root…
```

Обрати внимание: если ты не получишь права на изменение файла, в следующем шаге у тебя не получится сохранить создаваемый тобой HTML-файл.

HTML-страница

HTML-страница – это не более чем текстовая страница. Запусти редактор Leafpad (**Старт ⇨ Стандартные ⇨ Текстовый редактор**), введи следующий текстовый код:

```
<html>
  <head>
    <title>Добро пожаловать</title>
  </head>
  <body>
    <h1>Добро пожаловать!</h1>
    Это веб-сервер Raspberry Pi.
```

```
    Здесь ты найдёшь информацию о языке программирования
    <a> href="http://www.python.org"> Python </a>.
    <hr/>
    Последнее изменение &Mdash: 25 апреля 2019
  </body>
</html>
```

При вводе текстового кода обрати внимание на отступы.

Сохрани эту страницу по адресу */var/www/html/index.html*. Это позволит перезаписать старую стартовую страницу. Сделать можно так:

- нажми на **Файл** ⇨ **Сохранить файл как**;
- в пустом поле введи имя: путь */var/www/html/index.html*.

В браузере щёлкни мышью по кнопке **Reload this page** (Перезагрузи эту страницу), расположенной левее от поля адреса (рис. 18.2 слева, обведён). Страница *index.html* будет обновлена, и ты увидишь только что созданную веб-страницу.

Как это работает? HTML-текст содержит так называемые теги, которые заключены в угловые скобки. Это обозначения, определяющие формат области текста. Обычно теги встречаются парно. Вместе с открывающим тегом <…> идёт закрывающий тег, который начинается со знака косой черты </…>. Весь текст HTML начинается с `<html>` и заканчивается как `</html>`. Он состоит из двух частей:

- «шапка» текста обозначается через `<head> ... </head>`. Head в переводе с английского обозначает «голова». Она содержит скрытую информацию о документе. Здесь, в «шапке», между тегами `<title>` и `</title>` указан заголовок. В переводе `title` – это «заглавие». Этот заголовок отображается только в строке заголовка браузера (на рис. 18.2 обведён красной рамкой);
- тело текста обозначается тегами `<body> ... </body>`. Body в переводе – «тело». Здесь находится текст, ссылки и изображения, которые отображаются в окне браузера.

А вот ещё несколько указаний по тегам в теле HTML-документа.

- С помощью тегов `<hl> ... </hl>` создаётся заголовок первого уровня. Всё, что находится между этими двумя тегами, пишется большими буквами.
- Теги `<a href="..."> ... </a>` определяют гиперссылку. Всё, что находится между этими двумя тегами, появляется в браузере, окрашено другим цветом и/или выделено подчёркиванием. Если щёлкнуть мышью по такой гиперссылке, в окне браузера отобразится страница по

адресу, введённому в атрибуте `href="..."`. В этом примере ссылка ведёт к домашней странице Python.

- Тег `<hr/>` отображает горизонтальную линию.

Использование веб-сервера в локальной сети

Raspberry Pi соединён с локальной сетью (с помощью проводного подключения LAN или беспроводного подключения WiFi). Ты также можешь получать доступ к серверу и запрашивать информацию с других компьютеров этой локальной сети.

Веб-сервер Raspberry Pi называется localhost. На других компьютерах ты будешь использовать IP-номер своего Raspberry Pi. Как найти этот номер? Открой LXTerminal и введи в командную строку следующую команду:

```
$ ifconfig
```

IP-номер Raspberry Pi указан после inet adresse. Подробнее об этом рассказано в главе 2.

Проект 52. Который час? Создаём динамические веб-страницы

Первые веб-страницы были полностью статичными, т. е. представляли собой HTML-документы, которые сохранялись в каталоге */var/www/html* и отправлялись с Apache-сервера по запросу. Сайты были статичными, потому что всегда оставались такими, какие они есть, и никогда не менялись.

Но иногда информация, например время, отображаемая на сайте, должна изменяться. Значит, как только запрос попадёт на сервер, HTML-страница должна стать *динамичной*.

Что такое CGI-скрипт?

CGI-скрипт – это программа по созданию динамичных HTML-сайтов. Аббревиатура CGI означает *Common Gateway Interface* (общий шлюзовой интерфейс) и является стандартом, используемым для создания CGI-скриптов и управления серверов. HTTP-сервер не отправляет сохранённый текстовый файл, а разрешает выполнение CGI-скрипта и отсылает его вывод (данные). В этом разделе мы разработаем веб-страницу, которая по запросу будет отображать дату и время.

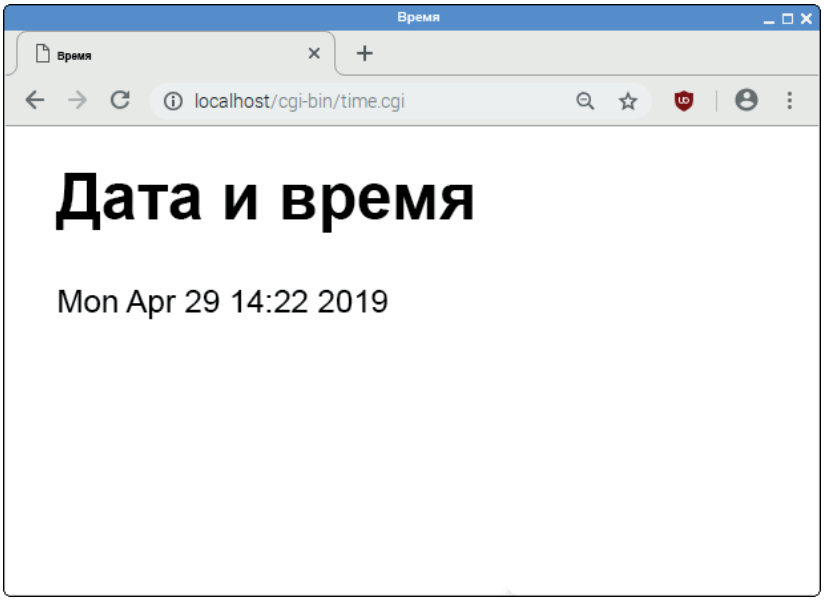


Рис. 18.4. *Так выглядит динамический сайт с указанием даты и времени*

HTML-страница создаётся с помощью скрипта Python, сохранённого под именем *time.cgi*. Да, это скрипт Python, но он будет иметь окончание не *.py*, а *.cgi*. Таким образом, программа Python становится CGI-скриптом.

Каталог cgi-bin

Все CGI-скрипты сохраняются в особом каталоге. На твоём Apache-сервере таким местом будет */usr/lib/cgi-bin*. Но следующее действие относится уже к функциям администратора `root`. Пользователь `pi` не имеет права записи и не может ничего сохранять в этом каталоге. Что же делать? Лучше всего изменить право на доступ. Команда `chown` разрешает пользователю `pi` вносить изменения:

```
$ sudo chown pi /usr/lib/cgi-bin
```

Подготовка Apache-сервера к CGI-скриптам

Чтобы твой Apache-сервер смог обработать CGI-скрипты, тебе необходимо ввести в LXTerminal следующую команду:

```
$ sudo a2enmod cgi
```

После этого нужно вновь перезапустить сервер:

```
$ sudo service apache2 restart
```

Программирование

В левом верхнем углу окна щёлкни мышью по кнопке главного меню и выбери команду меню **Программирование ⇨ Python 3(IDLE)** (Programming ⇨ Python 3(IDLE)), выбери команду меню **File ⇨ New file** (Файл ⇨ Новый файл). Введи нижеприведенный программный текст и сохрани его по пути:

```
/usr/lib/cgi-bin/time.cgi
```

Убедись, что ты действительно пишешь расширение файла *.cgi*, а не *.py*.

Строка формата с заполнителями

При программировании CGI-скриптов особенно удобна строка формата с заполнением. Строка формата – это что-то вроде формул, в определённых местах которых нужно вписать ещё что-то. Заполнители начинаются со знака %. Заполнитель %i предназначен для целого числа, а %s – для строки. С оператором форматирования % заполнители заменяются конкретными значениями, например:

```
>>> шаблон = '%S имеет %i друзей.'
>>> шаблон % ('Вася', 89)
'У Васи 89 друзей.'
```

Программный текст

```
#! /usr/bin/python3                    #1

from time import asctime
HTML = '''Content-type: text-html; char-set=utf-8

<html>
  <head>
       <title>Время</title>
  </head>
  <body>
   <h1> Дата и время </h1>
   %s
  </body>
</html>'''                             #2

print(HTML % asctime())                #3
```

Обрати внимание, в строке `%s` буква **s** должна быть маленькой, а не большой.

Как это работает?

`#1` Первая строка очень важна. Здесь вводится Python-интерпретатор, управляющий скриптом.

`#2` Здесь определяется длинная строка, через которую идут несколько строчек (тройные кавычки). Это образец для HTTP-пакета, который должен отправить Apache-сервер. В первой строке определяется, что это HTML-текст. Также указывается кодировка символов (utf-8). Внимание! Вторая строка *должна* остаться пустой. После неё следует нормальный HTML-текст с заполнителем `%s`.

`#3` Инструкция `print()` выводит HTTP-пакет, который далее должен быть отправлен Apache-сервером. Заполнитель `%s` заменяется на строку с датой и временем.

Тестируем скрипт с помощью IDLE 3

Если ты прописал и сохранил этот скрипт, то тебе нужно запустить его с помощью команды меню **Run** ⇨ **Run Module** (или нажав кнопку **F5**), чтобы проверить, правильно ли скрипт сгенерирован.

```
>>> ================ RESTART ================
>>>
Content-type: text-html; char-set=utf-8
<html>
 <head>
      <title>Время</title>
 </head>
 <body>
  <h1> Datum und Uhrzeit</h1>
  Fri Apr 26 20:44:26 2019
 </body>
<html>
```

Делаем скрипт рабочим

Чтобы скрипт действительно работал, тебе необходимо сделать его активным. С помощью команды `chmod +x` каждый пользователь системы получит право выполнения (`x`, *execute*).

Введи в командную строку терминала команду

```
$ sudo chmod +x /usr/lib/cgi-bin/time.cgi
```

Проверь, правильно ли настроены права доступа. Для этого введи в командную строку терминала команду

```
ls -l /usr/lib/cgi-bin/time.cgi
```

Теперь можно открыть на Raspberry Pi браузер Хромиум и протестировать динамический сайт. Введи в поле адреса следующий URI:

```
http://localhost/cgi-bin/time.cgi
```

Если ты несколько раз подряд вызовешь этот сайт, то всякий раз будет появляться новое время. Ты наверняка заметил, что в URI отсутствует */usr/lib*. Этот внутренний путь отличается от того адреса, который использует посетитель сайта.

Если ты хочешь войти на сайт, используя свой мобильный телефон или другой компьютер, тогда укажи вместо *localhost* IP-адрес своего Raspberry Pi. И URI будет таким:

```
http://192.168.178.40/cgi-bin/time.cgi
```

Найти свой реальный IP-адрес вместо приведённого 192.168.178.40 ты сможешь с помощью команды

```
$ ifconfig
```

Больше информации об этом в первой главе.

Проект 53. Шпион в саду

В HTML-страницу также можно вставлять изображения. Если на сайте изображение постоянно обновляется в реальном времени, значит, это изображение, получаемое с веб-камеры, которая у тебя есть.

Для этого проекта, кроме Raspberry Pi, тебе понадобятся:

- мобильный блок питания;
- модуль камеры.

Всё это можно настроить где угодно в рамках радиуса действия беспроводной сети. К примеру, вблизи кротовой норы в саду, чтобы понаблюдать, появится ли маленький землекоп на поверхности земли.

Если всё работает, то можешь запросить веб-страницу с картинкой в реальном времени с любого компьютера локальной сети или со своего мобильника.

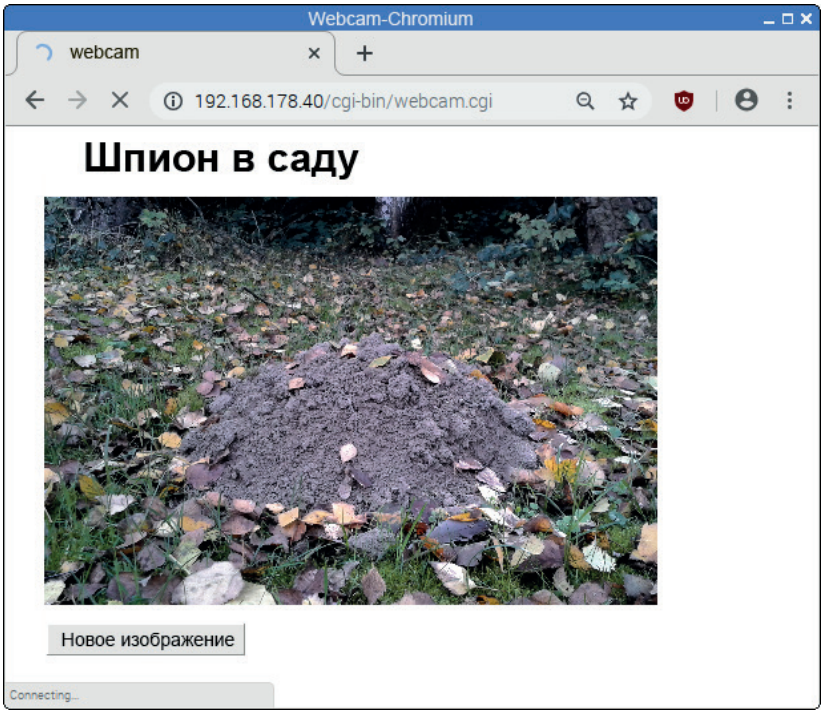


Рис. 18.5. *Веб-камера, установленная в саду*

Программное обеспечение – это CGI-скрипт, создающий сайт с кнопками. Нажав на одну из них, можно с помощью raspistill сделать новый снимок.

Расширенные права для CGI-скриптов

Чтобы сделать фото с помощью raspistill, понадобится Python-интерпретатор, выполняющий CGI-скрипт, и права `root`. Сейчас мы создадим вызов программы Python-интерпретатора, обладающей правами администратора. У профессиональных серверов, доступных в интернете, CGI-скрипты имеют заведомо ограниченные права доступа для защиты системы. Это сделано для того, чтобы CGI-скрипты, имеющие слабые места в системе безопасности, не позволили злоумышленникам изменить в системе то, что менять нельзя. В наших проектах такой риск невелик, тем более что твой Raspberry Pi недоступен в интернете.

Действуй так.

С помощью Unix-команды cp создай копию вызова интерпретатора под другим именем:

```
$ sudo cp /usr/bin/python3 /usr/bin/python3_root
```

Установи suid-Bit для Python3_root, чтобы получить права администратора:

```
$ sudo chmod u+s /usr/bin/python3_root
```

Контролируй эти права!

```
$ ls -l /usr/bin/python3_root
```

В следующей программе мы будем использовать этот особый интерпретатор в «Магической линии».

Пишем программу

```
#!/usr/bin/python3_root
#webcam.cgi
import os


COMMAND = 'raspistill -t 100 -w 400 -h 300 \
          + ' -o /var/www/html/bild.jpg -n'       #1
HTML='''Content-type: text/html; char-set=utf-8

<html>
 <head>
 <title>Webcam</title>
 </head>
 <body>
  <h1> Шпион в саду </h1>
   <p>
    <img src="/image.jpg" alt="Вид в сад" />
  </p>
  <form action=http://192.168.178.40/cgi-bin/webcam.cgi
       method="POST">
  <input type="Submit" value="Новое изображение" />
 </body>
</html>'''                                        #2

os.system(COMMAND)                                #3
print(HTML)                                       #4
```

Как это работает?

#1 Переменная `COMMAND` содержит команду, позволяющую камере сделать новый снимок, который всегда будет сохраняться под одним и тем же именем файла. Каталог – это корневой каталог Apache-сервера.

#2 Эта константа является HTML-страницей, которую постоянно выдаёт CGI-скрипт. Тег `<img .../>` помещает изображение с указанным адресом на это место. Атрибут `alt="Вид в саду"` означает, что в качестве альтернативы изображению будет появляться текст *с видом в сад*. Текст должен описывать содержание картинки и использоваться в программах чтения для слабовидящих. Но текст также должен показываться, если по каким-то причинам файл изображения недоступен.

С помощью `<form ...> ... </form>` определяется CGI-формуляр (форма). У всех интерактивных страниц, где вы можете что-то ввести или нажать, есть формуляры. В первом теге атрибут `action` определяет, что произойдёт при нажатии кнопки. В этом случае просто снова вызывается CGI-скрипт. (Следи, чтобы у тебя был введён «правильный» IP-номер.) А также определяется метод переноса данных (`get` или `post`). Об этом поговорим чуть позже.

В теге `<input type-"Submit" .../>` назначается кнопка – **Submit button**. С помощью атрибута `value` определяется надпись на кнопке.

#3 Выполняется новое фото.

#4 Вывод сайта. Текст всегда один и тот же, но отображаемая картинка каждый раз новая, так как содержимое файла изображения изменяется.

Сохрани скрипт под именем файла *webcam.cgi* в папке */usr/lib/cgi-bin* и активируй его с помощью ввода следующей команды:

```
$ sudo chmod +x /usr/lib/cgi-bin/webcam.cgi
```

Теперь тебе нужно подключить мобильный Raspberry Pi к источнику питания. Потом запусти операционную систему. Apache-сервер автоматически начнёт свою работу. Веб-камера готова к использованию.

Что делать в случае ошибки сервера?

Иногда твой CGI-скрипт работает не совсем так, как ты этого ожидаешь. Во время вызова скрипта браузер выдаёт сообщение **Internal Server Error** (Внутренняя ошибка сервера). Что делать? Примечание, где искать ошибку, ты найдёшь в файле журнала Apache-сервера. Это текстовый файл, в котором сервер Apache хранит сообщения об ошибках. Открой этот файл в текстовом редакторе Leafpad. Для этого введи в командную строку команду

```
$ sudo leafpad /var/log/apache2/error.log
```

Лёгким движением руки мобильник превращается... в модем

Использовать интернет для экспериментов с Raspberry Pi можно не везде. Если у тебя есть айфон или смартфон на базе Android, ты можешь создать точку доступа и войти с помощью телефона на сайт своего Raspberry Pi. Здесь в качестве примера приводится инструкция для мобильного телефона Samsung Galaxy. Для всех остальных смартфонов порядок действий такой же.

- Создай в папке *Python_project* папку *Flags*.
- Выбери функцию **Настройки** (Settings). Ярлык этой функции напоминает шестерёнку ⚙.
- Выбери **Беспроводные сети** (Wireless network), далее – **Привязка** (Binding), после чего – **Режим модема** (Tethering and modem mode).
- Нажми кнопку **Точка мобильного доступа** (Mobile access point).
- Проследи, чтобы все устройства получили доступ.
- В поле **Безопасность** выбери WPA 2 PSK. Это сокращение означает Wi Fi Protected Access с Preshared Key.
- Возможно, ты уже настроил пароль для соединения со своим ноутбуком. Для этого проекта лучше ввести новый пароль, который неизвестен другим людям.
- После того как ты всё подготовил, осталось активировать точку доступа. У телефонов Samsung для этого существует переключатель в виде ползунка, который надо сдвинуть вправо так, чтобы надпись на сдвижном

переключателе изменилась с 0 на I и переключатель принял вид  .

Теперь у тебя есть модем, управляемый с мобильника. Но ты можешь воспользоваться браузером на этом же телефоне, чтобы перейти на свою веб-страницу.

Проект 54. Совершенно секретно! Создаём сайт с защитой доступа

Может быть, ты не хочешь, чтобы твоя камера или какой-то другой веб-сервис стали объектом пристального внимания всех, имеющих доступ к локальной сети. В таком случае в этом разделе мы разработаем страницу, требующую авторизацию при входе. Введи в браузере своё имя, как на рис. 18.6. Сайт содержит поле ввода, в которое пользователь вносит своё имя. После этого нажми кнопку **Login**.

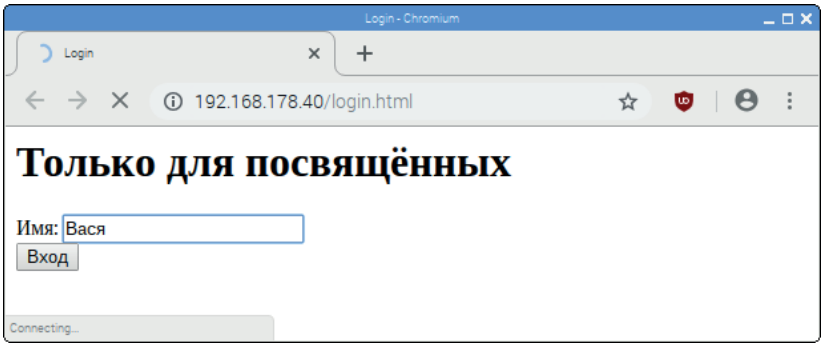


Рис. 18.6. *Внешний вид страницы входа*

Изображение камеры будет показано только при условии, что вводимое имя есть в сохранённом списке имён (рис. 18.7).

Если имя системе незнакомо, появляется соответствующее уведомление (рис. 18.8).

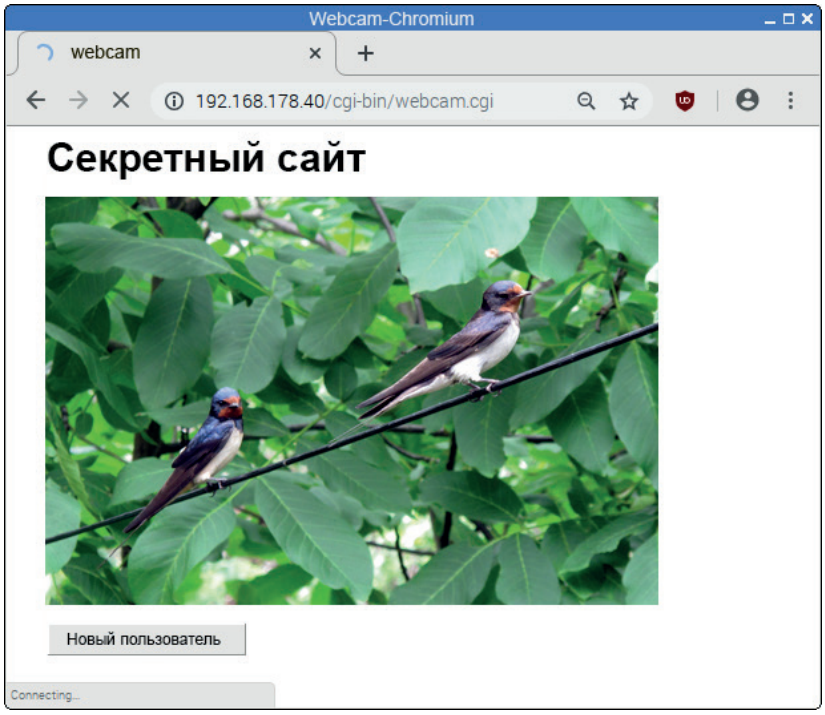


Рис. 18.7. *Динамический веб-сайт с картинкой в реальном времени*

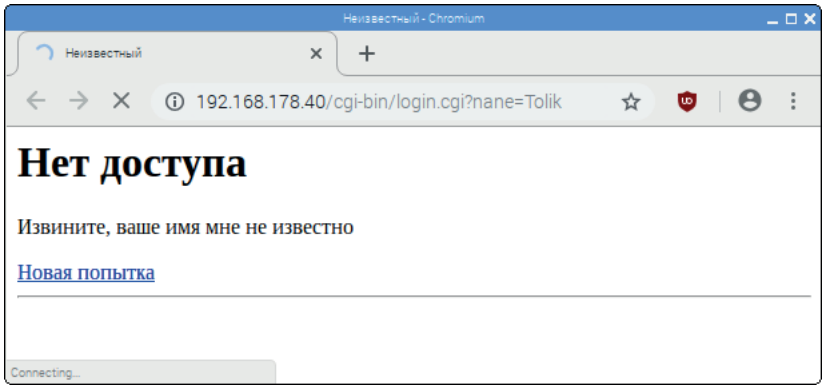


Рис. 18.8. *Динамически созданная страница ответа: ссылка ведёт к странице входа*

Проект состоит из двух частей:

- статичной HTML-страницы, содержащей форму с полем ввода для имени;
- статичной страницы входа, вызывающей CGI-скрипт. Он проверяет имя и возвращает динамически созданную страницу в качестве ответа.

Страница входа

Страница входа является статичной HTML-страницей. Назови её *login.html* и сохрани её по адресу:

```
/var/www/html/login.html
```

Теперь она находится в корневом каталоге Apache-сервера. Очень важно сохранить эту страницу в правильной папке.

Статичный HTML-текст

Это текст страницы входа.

```
<html>
  <head>
    <title> login </title>
  </head>
  <body>
    <h1>Только для посвящённых</h1>
    <from method="get"
       action=http://192.168.178.40/cgi-bin/login.cgi>
      Имя: <input type="text" name="name"> <br/>
      <input type="Sumbit" value="Вход"/>
    </from>
  </body>
</html>
```

Как это работает?

Документ опять содержит формуляр `<form ...> ... </form>`. После нажатия на кнопку отправки будет вызван скрипт CGI (*/cgi-in/login.cgi*). В области формы определяется однострочное поле ввода:

```
<input type=»text» name=»name»>
```

Когда кнопка отправки нажата, веб-сервер вызывает URI, который указан в теге `<form ...>`, в атрибуте `action="..."`. Поскольку был выбран метод `get`, то прикрепляется ещё одна строка запроса. Эта строка начинается со знака вопроса и отображает содержимое переменных формуляра. У нас представлена единственная переменная содержимого поля ввода. Если в поле ввода ввели имя `Вася`, то вызываемая строка запроса будет выглядеть так:

```
/cgi-bin/login.cgi?name=Вася
```

CGI-скрипт

Сохрани следующий CGI-скрипт по адресу:

```
/usr/lib/cgi-bin/login.cgi
```

Не забудь активировать файл. Для этого введи в командную строку терминала следующую строку:

```
$ sudo chmod +x /usr/lib/cgi-bin/login.cgi
```

Пишем программу

```
#!/usr/bin/python3_root
#login.cgi

import cgi, cgitb, os
cgitb.enable()                                          #1

COMMAND = 'raspistill -t 100 -w 400 -h 300 \
          + ' -o /var/www/html/image.jpg -n'            #2
FRENDS = ['Вася', 'Гриша', 'Таня']
UNKNOWN = '''Content-type: text/html                    #3

<html>                                                  #4
  <head>
    <title> Неизвестный </title>
  </head>
  <body>
    <h1> Доступа нет </h1>
    <a href="/login.html"> Новая попытка </a>
  </body>
</html>'''

WEBCAM ='''Content-type: text/html; char-set=utf-8

<html>                                                  #5
  <head>
    <title>Webcam</title>
  </head>
  <body>
    <h1> Секретный сайт </h1>
    <p>
     <img src="/image.jpg" alt="Вид в сад" />
    </p>
    <a href="/login.html"> Новый пользовтель </a>

  </body>
</html>'''
```

```
form = cgi.FieldStorage()                          #6
person = form.getvalue('name')                     #7

if person in FRENDS:                               #8
    os.system(COMMAND)
    print(WEBCAM)
else:
    print(UNKNOWN)
```

Как это работает?

#1 Модуль `cgitb` предлагает помощь в поиске ошибок. Обычно у CGI-скриптов ты получаешь уведомление об ошибке от Python-интерпретатора. Но если твоя программа содержит эту инструкцию, то будет по-другому. Если что-то не так, в браузере появится веб-страница с сообщениями об ошибке.

#2 В этой строке содержится команда для фотографирования изображения. Размер этого изображения 400×300 пикселей, а имя, под которым оно сохраняется в корневом каталоге Apache-сервера, – *image.jpg*.

#3 Список с именами людей, имеющих доступ к секретной странице.

#4 Веб-страница, которая будет отображаться на экране в случае *неудачного* входа в систему. Тег `<a href= ...> ... </a>` создаёт ссылку, по которой можно снова вернуться на страницу входа (статичная HTML-страница).

#5 Веб-страница, которая будет отображаться на экране в случае удачного входа в систему. Она показывает текущее изображение камеры. Внизу, под изображением, находится кнопка **Новый пользователь**, нажав на которую, ты вернёшься на страницу входа.

#6 Модуль `cgi` требуется для обработки входных данных сайта с помощью формы. Здесь создаётся объект `cgi.FieldStorage`. Он содержит все данные формы, переданные в строке запроса. В данном случае их не слишком много – лишь одно введённое имя.

#7 Здесь запрашивается значение переменной `name` из строки запроса и присваивается переменной `person`.

#8 Если имя, введённое в поле ввода **Имя** страницы входа в списке **FRENDS**, есть, на экране отобразится веб-страница **Секретный сайт** с изображением с камеры (рис. 18.7). Если же введённое имя не соответствует ни одному имени списка **FRENDS** (#3), на экране появится страница **Нет доступа** (рис. 18.8), где находится ссылка

Новая попытка, с помощью которой ты вернёшься на страницу входа **Только для посвящённых** (рис. 18.6), и у тебя появится новая возможность зайти на страницу **Секретный сайт** (рис. 18.7).

Проект 55. Управление светодиодом через сайт

Ты можешь управлять устройствами, подключёнными к GPIO Raspberry, с другого компьютера через интерактивную веб-страницу. Это могут быть такие устройства, как:

- мотор, двигающий камеру;
- электрический дверной замок;
- звонок;
- светодиод.

Для примера возьмём что-то совсем простое – управление светодиодом. Интерактивная веб-страница вызывается в браузере с URI, который выглядит примерно так:

```
http://192.168.178.40/cgi-bin/led.cdi
```

Имя файла показывает нам, что эта веб-страница была создана с помощью CGI-скрипта. С помощью экранной кнопки ты выбираешь желаемое состояние и нажимаешь кнопку **Включить** (рис. 18.9). Через короткое время в зависимости от выданной команды светодиод включится или выключится.

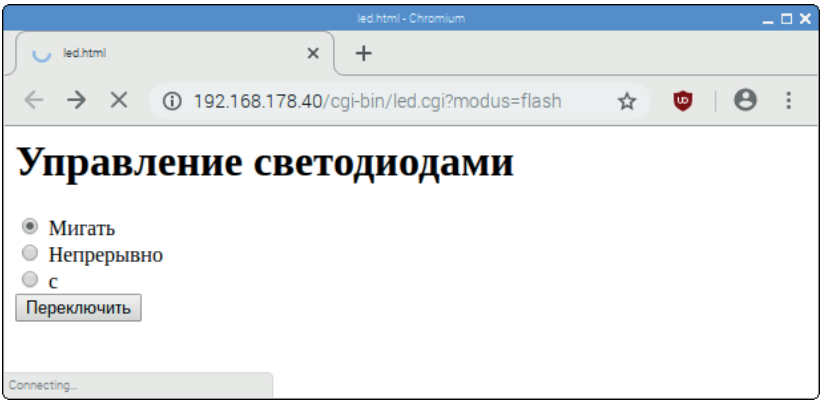


Рис. 18.9. *Интерактивная веб-страница для управления светодиодом с помощью экранной кнопки*

Аппаратное обеспечение

Анод светодиода (длинный вывод) соединён через резистор 100 Ом с контактом 1 GPIO (3,3 В). Катод (короткий вывод) подключается к контакту 10. Мы управляем контактом 10 с помощью CGI-скрипта. Когда контакт 10 соединяется с массой (команда управления контактом `False`), ток протекает через контакт 1 GPIO, анод светодиода, катод светодиода, контакт 10 GPIO, и светодиод излучает свет.

Сохрани следующий CGI-скрипт по адресу:

```
/usr/lib/cgi-bin/led.cgi
```

Активируй программный файл для всех пользователей:

```
$ sudo chmod +x /usr/lib/cgi-bin/led.cgi
```

Пишем программу

```
#!/usr/bin/python3_root
# led.cgi

import cgi, cgitb, os
# led.cgi
import cgi, cgitb
from RPi import GPIO
from time import sleep
HTML='''Content-type: text/html; charset=utf-8

<html>
<body>
 <h1>Управление светодиодами</h1>
 <form action="/cgi-bin/led.cgi" method="GET">
  <input type="radio" name="modus" value="flash" %s>
  Мигать <br/>
  <input type="radio" name="modus" value="duration" %s>
  Непрерывно <br/>
  <input type="radio" name="modus" value="from" %s>
  С <br/>
  <input type="submit" value="Переключить"/>
 </form>
</body>
</html>'''                                #1

def flash():
   for i in range(10):
      GPIO.output(10, False)
      sleep(0.2)
      GPIO.output(10, True)
      sleep(0.2)
```

```
cgitb.enable()
GPIO.setmode(GPIO.BOARD)
GPIO.setup(10, GPIO.OUT)
form = cgi.FieldStorage()
modus = form.getvalue('modus','of')   #2
if modus =='flash':
   flash()
   select = ('checked', '', '')       #3
else:
   if modus =='duration':
       GPIO.output(10, False)
       select = ('', 'checked', '')
   else:
       GPIO.output(10, True)
       select = ('', '', 'checked')
print(HTML % select)                  #4
```

Как это работает?

#1 Это шаблон для веб-страницы, возвращающей скрипт Apache-серверу. Он содержит три заполнителя `%s`, в которые позднее (#4) вносится одна строка символов.

Форма `<form ...> ... </form>` содержит три `input`-тега типа radio. Все три тега имеют одно и то же имя (атрибут `name="modus"`), но разные значения (атрибут `value`). С помощью экранного переключателя делается выбор. То есть можно выбрать только один переключатель, имеющий состояние `checked`. После нажатия на кнопку **Переключить** в строке запроса передаётся значение выбранного переключателя (рис. 18.9).

#2 Из объекта `FieldStorage` считывается значение переменной `modus`. Имя переменной – это первый аргумент вызова метода. Второй аргумент `'of'` – это значение, которое используется в случае отсутствия строки запроса. Помни, что во время *первого* вызова веб-страницы строка запроса не прикрепляется к URI. Такое происходит, только если ты уже заходил ранее на страницу и нажимал кнопку **Переключить**.

#3 Если выбран переключатель **Мигать** и, следовательно, переменная `select` получает значение `'flash'`, то происходят две вещи. Во-первых, вызывается функция `flash()`. Она обеспечивает десятикратное мигание светодиода. Во-вторых, здесь образуется кортеж из строки символов `'checked'` и две пустые строки (символов). Кортеж прикрепляется в строке #4 к шаблону веб-страницы. Это означает, что в теге стоит `'checked'` для первого экранного переключателя. Это действие приводит к тому, что

на веб-странице выбирается первый верхний переключатель. То есть состояние переключателя сохраняется.

#4 Здесь три заполнителя в тегах переключателя заменяются либо пустой строкой, либо `'checked'`.

Кортеж

Кортеж – это неизменяемая секвенция. Ты узнаешь её по круглым скобкам. Пример `('Вася', 'Иванов')`. Элементы кортежа – это предметы, прочно связанные между собой, например имя и фамилия человека. Кортеж также использует форматирование `Formatstrings` с заполнителями. Пример:

```
>>> шаблон = 'Фамилия %s - это %s'.
>>> шаблон % ('Вася', 'Иванов')
'Фамилия Васи - это Иванов'.
```

Проект 56. Управление домашними устройствами через беспроводную сеть

С помощью Python легко запрограммировать свой собственный HTTP-сервер. Иногда такой способ предпочтительнее, так как Apache-сервер должен быть настроен для специальных приложений, а это не так просто.

В этом проекте мы будем управлять двумя беспроводными розетками через интерактивную веб-страницу. К одной может быть подключена лампа, а к другой – вентилятор.

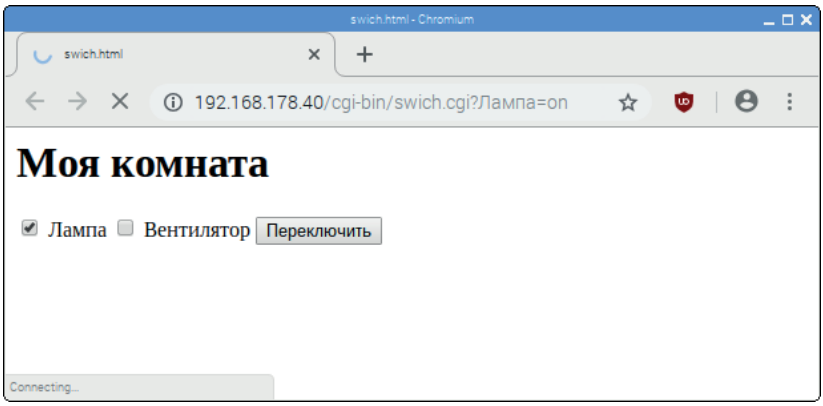


Рис. 18.10. *Интерактивная страница для управления устройствами*

На интерактивной странице имеется два чекбокса (множественный выбор). Таким образом можно определить отдельно для каждого прибора, должен ли он быть включённым или выключенным. На рис. 18.10 в качестве примера включён только вентилятор. При нажатии на кнопку состояние розетки каждый раз обновляется.

Веб-сервер

Создай в своём домашнем каталоге папку проектов, например *webserver*. В этой папке ты будешь хранить сервер под именем *httpd.py*. Эта папка проектов является корневой папкой программируемого сервера.

Пишем программу

```
#!/usr/bin/python3
# httpd.py
from http.server import HTTPServer, CGIHTTPRequestHandler
serveradress =('', 8080)                                    #1
server=HTTPServer(server_adress, CGIHTTPRequestHandler)     #2
server.serve_forever()                                      #3
```

Как это работает?

#1 Адрес сервера – это кортеж, который состоит из имени (здесь идёт пустая строка символов) и номера порта. Каждый HTTP-сервер использует порт. Стандартным портом является 80. Такой же порт использует Apache-сервер, всегда работающий в фоновом режиме. Соответственно, наш новый порт будет использовать другой номер. Обычно это четырёхзначный номер, который начинается с 8. Для больших номеров, как этот, серверу не требуются права суперпользователя.

#2 Создаём объект `HTTPServer`…

#3 … и запускаем его.

CGI-скрипт

Подготовка

Ты должен знать системный и UNIT-номер беспроводных розеток. Предположим, что в следующем скрипте розетка для лампы имеет системный номер 1 и Unit-номер 8. Розетка для вентилятора также имеет системный номер 1 и Unit-номер 4.

Перед тестированием этого скрипта попробуй вручную управлять розетками в командной строке, например:

```
$ sudo pilight-send -p mumbi -s 1 -u 8 -t
```

Возможно, возникнут проблемы, и тебе придётся приостановить pilight-сервис, а затем перезапустить его. Подробности этого ты найдёшь в главе 15.

В своей папке проекта с сервером *httpd.py* (корневой каталог) ты создашь новый каталог с именем *cgi-bin/*. В этом каталоге ты сохранишь следующий скрипт с именем *switch.cgi*.

Этот скрипт также нужно сделать активным:

```
$ sudo chmod +x /home/pi/webserver/cgi-bin/switch.cgi
```

Пишем программу

```
#!/usr/bin/python3_root
# switch.py
import cgi, cgitb, os

COMMAND = 'sudo pilight-send -p mumbi -s %i -u %i %s'    #1
HTML='''Content-type: text/html; charset=utf-8

<html>
<body>
 <h1>Моя комната</h1>
 <form action="http://192.168.178.40:8080/cgi-bin/schalter.cgi"
      method="GET">
  <input type="checkbox" name="lamp" value="on" %s>
  Лампа
  <input type="checkbox" name="ventilator" value="on" %s>
  Вентилятор
  <input type="submit" value="Переключить"/>
 </form>
 </body>
</html>'''                                                  #2

cgitb.enable()
form = cgi.FieldStorage()
lamp = form.getvalue('lamp','of')                          #3
ventilator = form.getvalue('ventilator','of')
if lamp =='on':
   os.system(COMMAND %(1, 8, '-t'))                        #4
   checked_lamp = 'checked'
else:
   os.system(COMMAND %(1, 8, '-f'))
   checked_lamp = ''
if ventilator =='on':
   os.system(COMMAND %(1, 4, '-t'))
```

```
    checked_ventilator = 'checked'
else:
    os.system(COMMAND %(1, 4, '-f'))
    checked_ventilator = ''

print(HTML % (checked_lamp, checked_ventilator))        #5
```

Как это работает?

#1 Эта константа содержит шаблон для команды `pilight-send`. В ней находятся три заполнителя. Два заполнителя `%i` предназначены для системного и Unit-номера. Оба они являются целыми числами (`integer`). Заполнитель `%s` предназначен для аргументов включения (`-t`) и выключения (`-f`).

#2 Длинная строка символов – это возвращаемая HTML-страница. Форма содержит два тега для чекбоксов. У каждого чекбокса свое имя, но оба имеют одинаковое значение. Если выбран какой-то чекбокс, его имя и значения будут перечислены в строке запроса, например `lamp=on`.

#3 Переменная `lamp` получает значение из объекта `FieldStorage`. Если `lamp=on` не присутствует в строке запроса, `form.getvalue()` возвращает значение `'of'`. В конечном счёте это означает, что переменная `lamp` имеет значение либо `'on'`, либо `'of'`.

#4 Здесь форматируется шаблон для команды (т. е. заполнители заменяются двумя числами и одной строкой), и эта команда выполняется. В этом случае лампа включается.

#5 В заключение идёт форматирование шаблона для веб-страницы. Это значит, в тегах для чекбокса вносится `'checked'` в случае, если он был выбран.

Запускаем HTTP-сервер в «безголовом» режиме

Apache-сервер запускается самостоятельно в случае включения RPi. А вот новый сервер ты должен запустить вручную. Теперь частенько можно управлять веб-сервером в «безголовом» режиме, т. е. без клавиатуры и монитора. Убедись, что на твоём Raspberry Pi активирован протокол *SSH* (см. главу 1) и что ты используешь *PuTTY* или подобный SSH-клиент для соединения с другого компьютера. Войди под именем пользователя `pi`.

Теперь можешь запускать сервер. Правда, есть одна проблема. При завершении сеанса сервер останавливается, но тебя это не устраивает. Ты хочешь, чтобы он продолжил работу, даже если ты закроешь SSH или PuTTY-сеанс. Решить это можно с помощью программы `nohup`. Поступай следующим образом.

Используя команду `cd`, перейди в каталог сервера:

```
$ cd webserver
```

Теперь введи команду

```
$ sudo nohup python3 httpd.py 2>/dev/null 1>/dev/null &
```

Таким образом ты запустишь команду `nohup` на правах администратора. Этот процесс, в свою очередь, запустит Python-интерпретатор, который выполняет HTTP-сервер. Обычно `nohup` пишет сообщения в файле *nohup.out*. Со временем этот процесс будет всё длиннее, пока в какой-то момент не займёт всю память. Аргументы `2>/dev/null 1>/dev/null` означают, что никакие данные не сохранились.

Не забудь, что знак `&` должен находиться в конце строки!

Это позволит HTTP-серверу работать дальше даже в случае завершения SSH- или PuTTY-сеанса.

Вопросы

1. Назови три типа `<input>`-тегов, которые можно использовать для ввода данных в форме.
2. Какие процессы происходят, если нажать на кнопку **Переключить**?
3. Что такое строка запроса?
4. В каком каталоге дерева каталогов Linux сохраняются по умолчанию статичные HTML-страницы для Apache-сервера?
5. В каком каталоге дерева каталогов Linux сохраняются CGI-скрипты для Apache-сервера?
6. Какой порт обычно использует Apache-сервер?

Задание. Измерение температуры через сеть

Бывают разные ситуации, когда хочется узнать, какая температура царит в том или ином месте. Например, виноделы проверяют температуру в бочках (иногда днём или ночью), поскольку температура показывает, насколько быстро проходит процесс брожения. Чем теплее, тем активнее винные дрожжи, отвечающие за производство алкоголя. В химической лаборатории стоит бутылка для брожения вина с сахарным раствором и дрожжами. Там установлен цифровой датчик температуры типа DS1820. В соответствии с инструкцией он подключён к 1-Wire-Bus Raspberry Pi (рис. 15.1). И у тебя должна быть возможность считывать температуру, используя свой мобильный телефон.

Разработай CGI-скрипт, возвращающий динамическую веб-страницу, как показано на рис. 18.11. Ты нажимаешь нижнюю ссылку, обновляя тем самым данные измерения и значение температуры.

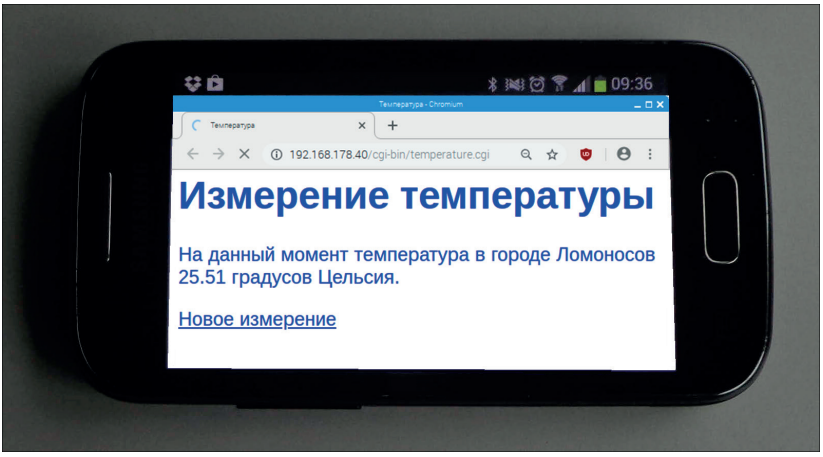


Рис. 18.11. *Интерактивная веб-страница на экране смартфона*

Ответы на вопросы

1. Чекбокс `<input type="checkbox" ...>`, кнопка `<input type="radio" ...>`, однострочное поле ввода `<input type="text" ...>`.

2. При нажатии на **Переключить** вызывается новый веб-ресурс, обычно это CGI-скрипт. URI находится в начале тега формы `<form action="..." ...>`. Кроме этого, отправляются данные, которые пользователь вводил в форму.
3. Если выбран метод переноса `get`, отправляются данные формы в виде строки запроса. Эта строка начинается с вопросительного знака. Содержимое переменных отображается в форме *name=значение*. Между несколькими значениями переменной стоит знак `&`. Пример:

```
?lamp=on&ventilator=on
```

4. Статичные HTML-страницы: */var/www/html/*.
5. CGI-скрипты: */usr/lib/cgi-bin/*.
6. HTML-сервер использует порт 80.

Решение задачи

Подготовка

Подключи датчик температуры и узнай его ID-номер. Сделать это можно так.

Чтобы запустить программу интерфейса для работы термодатчика, введи в окне LXTerminal следующие команды:

```
$ sudo modprobe w1-gpio
$ sudo modprobe w1-therm
```

Затем открой в файловом менеджере папку

```
/sys/bus/1-wire/w1/devices -
```

и найди там каталог термодатчика. Его имя – это ID-номер. Тебе потребуется имя каталога в твоей программе, потому что интерфейс записывает текст с температурными данными в файл в этом каталоге (рис. 15.1).

Сохрани следующий CGI-скрипт по адресу:

```
/usr/lib/cgi-bin/temperature.cgi
```

Активируй программный файл для всех пользователей:

```
$ sudo chmod +x /usr/lib/cgi-bin/temperature.cgi
```

Пишем программу

```
#!/usr/bin/python3_root
#temperature.cgi

import os, time, cgitb
cgitb.enable()
os.system('sudo modprobe wire')
os.system('sudo modprobe w1-gpio')
os.system("sudo modprobe w1-therm")

FILE = '/sys/bus/w1/devices/xxx/w1_slave'      #1
HTML = '''Content-type: text/html; char-set=utf-8

<html>
  <head>
    <title>Температура</title>
  </head>
  <body>
    <font face="Arial" color="blue" size=8>
    <h1>Измерение температуры</h1>
    <p>На данный момент температура в городе
       G&auml;rflasche %s Grad Celsius.</p>
    <a href="/cgi-bin/temperatur.cgi"> Новое измерение </a>
    </font>
  </body>
</html>'''                                      #2

def temperature():
    ok = False                                 #3
    while not ok:
        f = open(FILE, 'r')
        line_1, line_2 = f.readlines()
        f.close()
        if line_1.find("YES") != -1:
            ok = True
    part_1, part_2 = line_2.split("=")
    return int(part_2)/1000

print(HTML % temperatury())                    #4
```

Как это работает?

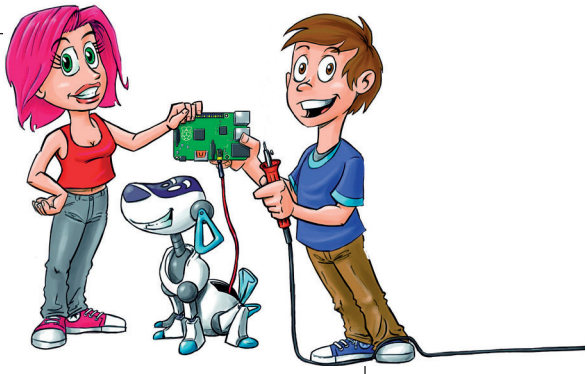
#1 Здесь находится путь к файлу с данными температуры. *xxx* заменяется на ID-номер термодатчика.

#2 Возвращаемый шаблон для HTML-страницы, которая содержит заполнители для температурных значений. Особенностью является тег `<font ...> ... </font>`. Атрибуты определяют некоторые характеристики шрифта. Размер имеет значение. Шрифт `size=8` в восемь раз

больше обычного: так веб-страница будет хорошо читаться и с маленького экрана мобильного телефона.

#3 Функция `temperature()` считывает файл с температурными данными, ищет в тексте температуру и возвращает её в виде целого числа (тип `int`).

#4 Эта инструкция содержит несколько шагов. Сначала вызывается функция `temperature()`, которая считывает температуру. Это число вносится вместо заполнителя `%s` в шаблон HTML-страницы. При этом десятичное число, показывающее значение температуры, автоматически преобразовывается в строку символов. Наконец, весь текст (веб-страница) с помощью функции `print()` выводится на экран.



Примечания для родителей и преподавателей

Мы живём в цифровом мире. И нам постоянно приходится пользоваться техническими устройствами с клавиатурой, камерой, динамиком и дисплеем. Смартфоны, планшеты и ноутбуки стали нашими незаменимыми помощниками и применяются теперь повсеместно: в школе, в профессиональной среде, на досуге. Такими же помощниками являются и такие высокотехнологичные устройства, как автомобили, печи, пылесосы, стиральные машины, в которых используется компьютерное управление, – так называемые встроенные системы.

Наш мир – это цифровая вселенная, но особенностью цифровой техники является то, что она невидима. Систему управления старой паровой машины, работающей через клапаны, рычаги и железные колёса, можно потрогать, посмотреть и послушать. Можно даже проследить принципы её работы, просто наблюдая за её работой или подетально разбирая и вновь собирая.

Что касается планшета или смартфона, то мы можем увидеть только его внешнюю оболочку. Даже если устройство разобрать, будут видны лишь впаянные в печатную плату микросхемы. Ну а как устройство работает, мы увидеть не сможем, потому что вся его работа скрыта. Цифровая техника запрятана в капсулу, как в чёрный ящик самолёта.

Raspberry Pi позволяет этот ящик немного приоткрыть. В Raspberry Pi мы видим печатную плату, токопроводящие дорожки и распаянные электронные компоненты. Камера,

периферийная память, клавиатура и все другие устройства, принадлежащие компьютерной системе, собираются и настраиваются самостоятельно.

Кроме того, можно подключить самостоятельно изготовленные электронные схемы. С помощью Raspberry Pi, изучая цифровую технику, создаем проекты, пробуем различные варианты и экспериментируем.

Что нам нужно для работы?

Любому, кто работает с Raspberry Pi, требуется множество деталей. Каких именно деталей, зависит от того, что мы хотим создать. Для начала я рекомендую следующую конфигурацию: Raspberry Pi, карта микро-SD (8 Гб), блок питания, USB-клавиатура, USB-мышь. Также необходимо интернет-подключение. Существует своего рода «стартовый набор» для новичков (туда входит обычно Raspberry Pi, корпус, кабель HDML, блок питания). Но стоимость его не всегда ниже, чем покупка всех элементов по отдельности. Если есть желание собрать Raspberry Pi самостоятельно, от корпуса можно и отказаться.

В первых главах Raspberry Pi используется как недорогой компьютер. Мы его конфигурируем, устанавливаем программное обеспечение и рассматриваем использование операционной системы Linux. Далее начинается программирование. Сначала в программе Scratch, а после и в Python. На этом этапе потребуется дополнительное оборудование.

Концепция книги навеяна проектом «Сделай сам» («Build it Yourself» http://build-it-yourself.com). «Build it Yourself» – группа преимущественно американских подростков, увлекающихся цифровой техникой и программированием. Основная идея заключается в том, что для проекта требуется несколько уже работающих устройств, а также компьютер и PicoBoard (монтажная плата). Все остальные детали – чистой воды импровизация из подручных материалов («first quality junk»).

Детали, которые требуются для работы над проектами, можно разделить на четыре группы:

- приборы особого назначения, которые нужно купить новыми: Raspberry Pi, модуль камеры для Raspberry и монтажная плата (PicoBoard, ранее ScratchBoard). Всё это можно заказать в интернете. Больше информации об этом в следующем разделе;

- устройства, которые, возможно, уже имеются дома или те, что используются для иных целей: карта micro-SD, монитор, клавиатура, мышка, блок питания, наушники, колонки, цифровой универсальный измерительный прибор или тестер универсальный (для измерения напряжения, силы тока и сопротивления), беспроводные розетки и светодиодные лампы с изменяемым цветом. Самым дорогим в этом наборе является монитор. Если не хочется вкладывать в него средства, можно поискать для этих целей подержанную модель. В крупных городах имеются магазины подержанной электроники. Но лучше всего (и не так уж дорого) было бы приобретение HD-монитора с входом HDMI. В случае если интерес к Raspberry Pi со временем ослабеет, его можно либо использовать в качестве второго монитора, либо просто продать;
- электронные детали и аксессуары. К ним относятся разъёмы, шлейфы с разъёмами «папа-мама» на концах (джампер-кабели), резисторы, потенциометр, светодиоды, температурные и ультразвуковые датчики. Они стоят недорого. А если их заказать оптом в большом количестве, то стоимость одного экземпляра будет ещё меньше. Заказывать товары сообща можно, если объединиться с другими любителями Raspberry Pi. Может быть, даже стоит скооперироваться с кем-то в школе или организовать рабочую группу. Больше информации об этих специальных группах вы найдёте в «Списке покупок» в следующем разделе;
- предметы и инструменты, используемые в хозяйстве. Хорошими материалами для экспериментальных моделей собираемых самостоятельно устройств являются картон, алюминиевая фольга, клей, звонковый провод и клейкая лента. С этими предметами уже можно смастерить очень многие узлы, например штурвал для симулятора гонок, раму для светодиодной матрицы, фотометр и переключатели всех видов.

Список покупок

В этом разделе вы найдёте описание некоторых деталей, которых обычно нет дома и которые нужно заказать в интернете. Информация о закупках есть в разделе «Почта со всех концов света, или Как заказать онлайн» на стр. 556.

Набор для начинающих

Для проектов с Raspberry Pi существуют различные «Наборы для начинающих», которые можно приобрести по цене от 6000 руб. Такой пакет обычно содержит современную модель Raspberry Pi, маленький корпус, блок питания и соответствующие элементы. Важным пунктом здесь является корпус. Его внешний вид, конечно же, строго индивидуален. Я же считаю, что он должен быть прозрачным, так как идея работы с RPi заключается как раз в том, чтобы открыть «чёрный ящик» компьютера. Кроме того, все разъёмы (в особенности GPIO с большим количеством контактов) должны быть в свободном доступе.

Съёмная плата и простые электронные компоненты

- Монтажная плата (PicoBoard или ScratchBoard) от 200 руб.
- Резисторы:
  - добавочные резисторы для светодиодов по пять штук номиналом 100 и 130 Ом;
  - дополнительные резисторы номиналом каждый 330 Ом, 470 Ом, 560 Ом, 1 кОм, 1,7 кОм и 4,7 Ом.
- Три или четыре светодиода различных цветов для разных проектов.
- 16 белых светодиодов для светодиодной матрицы (глава 11). Очень выгодно их приобретать набором (от 20 шт.). Качество светодиодов не должно быть слишком высоким, так как они будут использоваться не для освещения помещения.
- Примерно десять штук джампер-кабелей (соединительных проводов) «мама-мама» (male-male) и десять джампер-кабелей (соединительных проводов) «папа-папа» (female-female). Можно также приобрести ассортимент различных перемычек в пластиковой коробке.
- Поворотный потенциометр номиналом 10 кОм с линейной зависимостью без дополнительных выводов для тон-компенсации. Потенциометр с линейной зависимостью обозначается или буквой А (кириллица, отечественный стандарт), или буквой B (латиница, западный стандарт). Его стоимость от 80 руб. С его помощью можно соорудить штурвал для симулятора гонок. Резистор с линейной зависимостью позволит равномерно управлять поворотом машинки при любых отклоне-

ниях руля без скачков и провалов. Если взять резистор с логарифмической зависимостью (буква Б – кириллица, отечественный стандарт, или С – латиница, западный стандарт) или обратно логарифмической зависимостью (буква В – кириллица, отечественный стандарт, или А – латиница, западный стандарт), то управление поворотом машины будет сопровождаться скачками и провалами. То есть, чтобы повернуть, например, направо, придётся слегка повернуть руль вправо, а для поворота налево поворачивать руль влево на большой угол.

- LCD-дисплей размером 16×2 с широко известным контроллером HD44780 фирмы «Хитачи» (Hitachi) от 600 руб. С ним можно разработать таймер и другие приложения с использованием двухстрочного дисплея (глава 13).
- Светодиодная матрица TA07-11HWA фирмы «Кингбрайт» (Kingbright), от 200 рублей. Преимуществом такого элемента является то, что с его помощью можно запрограммировать светодиодную матрицу без необходимости паять её. Проекты по этой теме можно найти в главе 11.

Датчики

- Один или несколько термодатчиков (DS18S20 или DS18B20). Если купить сразу несколько, то можно сэкономить, заплатив не более 80 руб. за штуку (глава 15).
- Ультразвуковой датчик для измерения расстояния (US 020 стоимостью от 300 руб.). Существует несколько проектов с использованием этого датчика, например будильник или цифровой тренер по гимнастике, который отслеживает и контролирует движения участника.

Беспроводные розетки и передатчик

Для управления крупными электронными устройствами требуются беспроводные розетки. Получить набор из нескольких розеток и пульта управления (например, марки Mumbi) можно от 1300 руб. Для управления с помощью Raspberry требуется ещё и передатчик 434 МГц (WRL-10534 от 200 руб.). Больше информации об этом в главе 15.

Модуль для камеры RPi

Модуль камеры представляет собой маленькую плату с камерой высокого разрешения, одним светодиодом и одним

кабелем для подключения к RPi. Существует два типа такого модуля – обычный, с инфракрасным фильтром, и модуль NoIR, который содержит инфракрасный фильтр. Модуль первого типа делает снимки лучше, чем модуль NoIR, который может принять инфракрасное излучение и сделать его видимым. Модуль камеры стоит примерно 2000 руб. Проекты с его использованием можно найти в главе 17.

Монтажная плата PicoBoard

Наиболее простым является программирование с использованием языка Scratch. В этой книге есть проекты, в которых компьютер обрабатывает звук и свет. Например, управление летящей уткой посредством звуков и хлопков в ладоши. Фотометр позволяет по цвету определить содержание фруктового сока в лимонаде. В программе симуляции управление гоночным автомобилем на трассе происходит с помощью самодельного руля и педали газа. Для таких проектов необходима монтажная плата PicoBoard. Здесь есть датчики, кабель с зажимами типа «крокодил» и рычаг. Больше информации об этом в главе 5. Устройство PicoBoard можно приобрести онлайн от 2600 руб. (например, здесь: http://www.sparkfun.com).

Почта со всех концов света, или Как заказать онлайн

Материалы из первой и третьей групп обычно заказывают в интернете. В последнее время многочисленные компании по типу «товары почтой» открыли продажу специализированных изделий. Самыми выгодными оказались торговцы с азиатских площадок (например, www.aliexpress.com). Обычно товары оттуда приходят в течение трёх недель. Компания даёт полную защиту прав потребителя и два месяца, для того чтобы вернуть покупку. Но если что-то действительно не работает, то можно прождать довольно долго, прежде чем получишь замену. Однако и в России существуют онлайн-магазины электроники, предлагающие товары для работы с Raspberry Pi. Среди них такие, как «Онпад» (https://onpad.ru), «Вольтик» (https://voltiq.ru), «Чипстер» (https://chipster.ru), «Ультраробокс» (https://www.ultrarobox.ru).

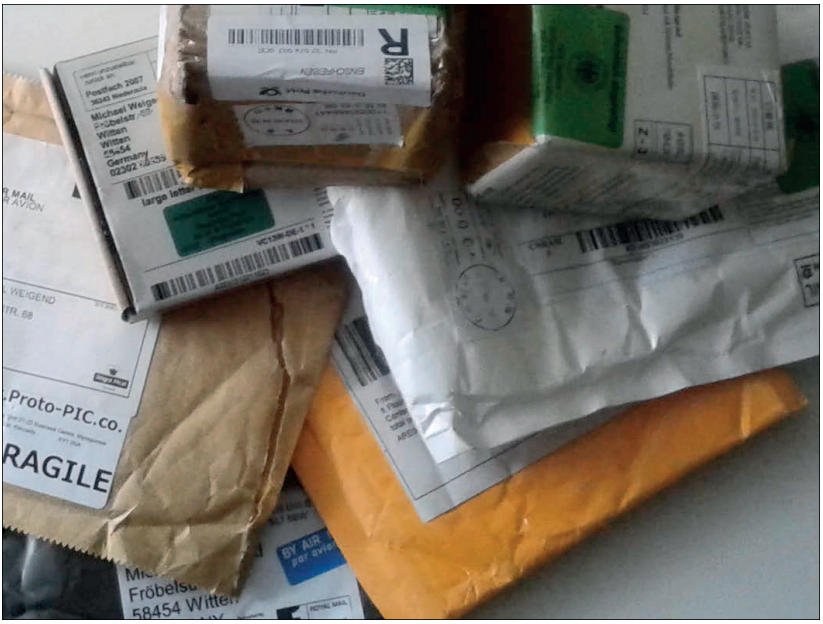


Рис. А.1. *Заказывая онлайн, будь готов к получению посылок со всех концов света*

Как работать с книгой?

В первых главах речь пойдёт о настройках Raspberry Pi и использовании его в качестве медиацентра. В этой же главе детям понадобится помощь взрослых.

- На карте SDHC установлена операционная система. Она имеется не только у Raspberry, но и у любого другого работающего компьютера. К такому компьютеру, конечно же, всегда должен быть доступ. Обычно при установке системы требуются знания экспертов.
- Для работы с другими программами и с другими проектами Raspberry также потребуется доступ к электросети и в интернет. Дети должны знать названия ваших сетей и пароли от них. Лучше всего записать эти кодовые слова в секретном месте. Листок с информацией не должен покидать пределов дома.

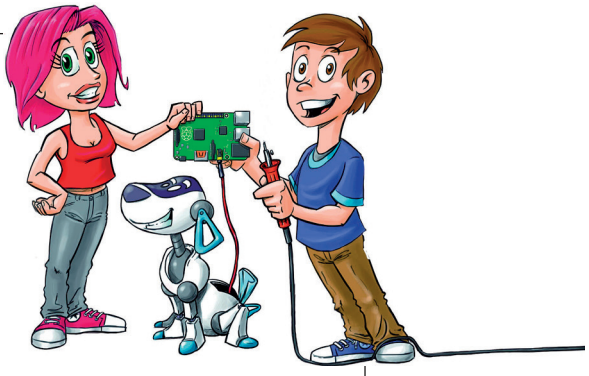
В остальной части книги представлены «Начальные проекты» с использованием Scratch и Python. Это очень простые и бесхитростные программы. Сами кодовые тексты должны быть по возможности краткими, простыми, тогда их можно будет создавать быстрее. Коды программ должны побуждать детей к тому, чтобы продолжать работу,

«оформлять» программное обеспечение и реализовывать свои собственные идеи.

Перед тем как приступить к творчеству и отправиться в свой собственный путь, очень важно следовать предписанным правилам, продвигаясь, как минимум, на шаг вперёд. Не каждый ребёнок и подросток способен сделать это сразу. Часто им не хватает терпения. Некоторые из них, начав читать, убеждают себя, что «так я никогда не сумею» или «это слишком сложно для меня». А есть и такие, кто предпочитает, чтобы технику им «по-быстрому» объяснил кто-то другой. Это создаёт зависимость и не приносит должного удовольствия от работы. Чтение инструкции шаг за шагом является своего рода искусством. Если вы поймёте, что ваш ребёнок или ученик не в состоянии двигаться дальше, то обычно ему или ей достаточно объяснить три «правила чтения» из вступительной главы.

По своему опыту скажу, что это является основой изучения. Начинать порой сложно, но после первого успеха всё пойдёт по накатанной и будет приносить удовольствие от процесса. Новичкам частенько помогают пометки о проделанной работе в инструкции. Так они всегда видят, где находятся и каким будет их следующий шаг.

Когда первые трудности пройдены, назад пути уже нет. Raspberry Pi увлёк миллионы людей во всём мире, а ведь это не только дети и подростки, отдающие своё свободное время работе с «малинкой», но и студенты и ученые с серьёзными исследовательскими намерениями.



Указатель

Символы

1-Wire-Bus, 439
--kiosk, параметр, 68
--noerdialog, параметр, 68

А

Автозапуск, 68
Автосимулятор, 95
Аддон, 77
Аналоговый аудиовыход, 21
Аргументы командной строки, 456

Б

«Безголовый» Raspberry, 59
Бесконечный цикл, 304
Беспроводная розетка, 450
Блокнот IDLE, 247
Блуждающие звёзды, 401

В

Ввод и вывод данных, 256
Веб-сайт, 17
  образцы программ, 17
Веб-сервер, 518
Велик для чтения (команда), 187
Виджет, 470
Внутренний цикл, 301
Возведение числа в степень, 242

Г

Гимнастика с ультразвуком, 482
Гороскоп, 385
Графический пользовательский интерфейс, 251, 468
  виджеты, 470
  разметка, 472

Д

Датчик температуры DS18B20, 439
Дверной звонок, 319
Двужильный кабель, 179
Дерево выбора, 297
Дерево файлов, 42
Динамические веб-страницы, 525
Диоды, 278
ДИП-переключатель, 451
Дополнительный резистор, 280
Дортмундер U, 338

Ж

Жидкокристаллический индикатор, 388

З

Загрузка, 31
Замедление, 257

Замедленная съёмка, 504
Запуск программы Python, 250
Затемнение (диммирование), 354
Звуковой файл, 316

И

Изменение пароля, 33
Изменяемая последовательность элементов, 152
Измеритель уровня звука, 187
Имена, 263
Импортирование модуля, 245
Импортирование функции, 244
Индекс, 152, 361
Инструкция, 292
  for, 366
  if, 292
  while, 300
  условная, 292
Интерфейс конфигуратора, 59
Итерация, 366

К

Калькулятор, 241
Карта памяти, 19
Коллекция, 359
Команда
  adduser, 50
  alsamixer, 58
  cd, 47
  chmod, 429
  cleanup(), 283
  exit, 62
  ifconfig, 59, 525
  input(), 255
  ls, 47
  mocp, 56
  nohup, 546
  passwd, 50
  велик для чтения, 187
Комментарии, 258
Комплементарные цвета, 201
Консоль LXTerminal, 46
Контекстное меню, 130
Контроллер MaKey MaKey, 222
Контрольная переменная, 487
Кортеж, 360

Л

Лексика, 377

М

Магическая линия, 253
Макетная плата, 278
Математическое выражение (терм), 241
Материнская плата, 18
  Raspberry Pi 3 модель b, 18
Мегагерц, 451
Медиацентр, 71
Меню-консультант, 489
Метаданные, 76
Метка, 471
Метод, 369
Множество, 361
Модуль
  NoIR, 494
  PIL, 499
  time, 244
  видеокамеры, 494
  ультразвуковой, 431
Монитор, 19
Монтирование файловой системы, 45
Музыкальный центр, 65
Мультяшные герои, 129

Н

Напряжение, измерение, 282
Нулевое время, 424

О

Окно Python Shell, 249
Оператор
  ветвления, 291
  сравнения, 298
  условный с двумя ветвями, 293
Охлаждение при испарении, 449
Ошибка сервера, 533

П

Параллельные процессы, 196
Параметр
  --kiosk, 68
  --noerdialog, 68

Педаль газа, создание, 224
Передатчик WRL-10534
RF-Link, 451
Переключатель, 308
Переменная управления, 489
Переменные, 263
  контрольные, 487
Перфолента, 323
Перфорированная карта, 323
Пинг-понг, 191
Планеты, 365
Подсказка, 253
Предустановленные параметры значений, 414
Преобразователь, 417
Приветствие, 253
Протокол, 454
Процессор, 21
Пустые объекты, 298

Р

Рабочий стол, 35
Разъём
  HDMI, 22
  RJ45, 21
Распознавание движения, 500
Распознавание речи, 182
Растр, 479
Руль управления, 224

С

Сбой программы, 422
Световой сигнал, 333, 379
Светодиод, 280
Светодиодная матрица, 339
Светодиодный матричный индикатор, 349
Секвенция, 360
  длина, 363
  кортеж, 542
  объединение, 364
  слайсинг, 395
Сенсор (датчик), 181
Сигнал SOS, 284
Сигнализация, 320
Сигнальная азбука, 477
Симуляция высадки на Луну, 231
Синхронизация через сообщения, 140
Системные номера, 450
Системный администратор (root), 34
Скрипт, 245
Скрытые каталоги и папки, 69
Следы движения, 513
Случайные функции, 371
Смешивание цветов, 476
Содержание переменной, 256
Сопротивление, 215
  измерение, 273
Списки, 152, 360
  изменяемые, 368
Список фрагментов, 395
Средний уровень громкости, 190
Строка документации, 414
Строка символов, 361
Строка формата с заполнителями, 527

Т

Тайм-аут, 423
Таймер, 397
Температурные данные, 440
Терм (математическое выражение), 241
Тормозной путь, 257
Трансмиттер, 417

У

Управление разными устройствами, 272
Условие, 298
Условный повтор, 300
Установка программного обеспечения, 22
Устройство для приготовления лимонада, 197

Ф

Формат
  CSV, 445
  MP3, 56
  OGG, 56
Форматирование SD-карты, 26
Функция, 405
  asctime(), 244
  float(), 259
  min(), 244

os.system(), 443
print(), 249, 255
range(), 374
round(), 243, 244
блок инструкций, 405
возвращение значения, 409
вызов, 408
заголовок, 405
имя, 406
определение, 408
тело, 405
Функция времени (тестирование), 244

Ц

Цветной ключ, 506
Цифровой датчик прямого назначения, 438
Цифровые часы, 396

Ч

Частота звука, 415
Чекбокс, 486

Ш

Шестнадцатеричные числа, 475
Шуточный мультфильм, 128

Э

Эдисон, Томас, 151
Экранный переключатель, 486
Эксперимент с дрожжами, 504
Экспозиметр, 205

A

adduser, команда, 50
Advanced Linux Sound Architecture (ALSA), 58
alsamixer, команда, 58
Amixer, 66
Aplay, 316
Application Launch Bar, 36
ARM (Advanced RISC Machines), 21
asctime(), функция, 244

C

Camera Serial Interface (CSI), 495
cd, команда, 47
CGI-скрипт, 525
chmod, команда, 429
cleanup(), команда, 283
Common Gateway Interface (CGI), 525
Creative Commons, 129
CSV, формат, 445

D

Dictionarys (словари), 375

E

entry, виджет, 476
exit, команда, 62

F

False, состояние контакта GPIO, 309
float(), функция, 259
for, инструкция, 366

G

GPIO (интерфейс ввода-вывода), 21, 275

H

HD44780, контроллер, 389
history (история), 241
HTML-страница, 523
Hypertext Markup Language (HTML), 519
Hyper Transfer Protocol (HTTP), 519

I

ifconfig, команда, 59, 525
if-else, условный оператор с двумя ветвями, 293
input(), команда, 255
IP-адрес, 59

L

LibreOffice, 447
Linux, 18
ls, команда, 47
LXTerminal, 46

M

MaKey MaKey, контроллер, 222
Mathematica, 39
min(), функция, 244
MOC (Music on Console), 56
mocp, команда, 56

N

nohup, команда, 546
NOOBS, 23

O

OSMC (Open Source Media Center), 72
os.system(), функция, 443

P

passwd, команда, 50
Picoboard, 178
Pilight, 452
print(), функция, 249, 255
PuTTy, 60
PuTTY, 545
Python
  IDLE, 239
  две важные комбинации клавиш, 240
  интерактивный режим, 240
  интерпретатор, 238

Q

QWERTY-раскладка, 32

R

range(), функция, 374
Raspbian, дистрибутив, 18
raspistill, программа, 497
raspivid, программа, 497
RGB-значения, 507
round(), функция, 243, 244
RPi.GPIO, модуль, 281

S

Scratch, 82
  астероиды, 123
  версии программы, 85
  время, 112
  загрузка проекта, 122
  координаты сцены, 88
  костюмы, 90
  лабиринт, 124
  метеориты, 106
  мокрица, 124
  объект, 83
  переменная, 102
  сайт, 120
  скрипт, 86
  создание ремикса, 123
  студии, 120
  сцена, 83
  фон, 94
  цвет, 101
  числа, 112
ScratchBoard, 181
SD-card (карта памяти), 19
SD-карта, форматирование, 26
SOS, сигнал бедствия, 284
SSH (Secure Shell), 59, 60
SSH, протокол, 545
Strings (строчки), 360

T

time, модуль, 244
True, состояние контакта GPIO, 309

U

Unit-номера, 450

W

WinRAR, архиватор, 24

Книги издательства «ДМК Пресс» можно заказать
в торгово-издательском холдинге «Планета Альянс» наложенным платежом,
выслав открытку или письмо по почтовому адресу:
115487, г. Москва, 2-й Нагатинский пр-д, д. 6А.
При оформлении заказа следует указать адрес (полностью),
по которому должны быть высланы книги;
фамилию, имя и отчество получателя.
Желательно также указать свой телефон и электронный адрес.
Эти книги вы можете заказать и в интернет-магазине: **www.a-planeta.ru**.
Оптовые закупки: тел. **(499) 782-38-89**.
Электронный адрес: **books@alians-kniga.ru**.

Михаэль Вайгенд

Raspberry Pi для детей

Главный редактор *Мовчан Д. А.*
dmkpress@gmail.com
Научный редактор *Герасименко А. С.*
Перевод *Энглерт Ю. Ю.*
Корректор *Синяева Г. И.*
Верстка *Чаннова А. А.*
Дизайн обложки *Мовчан А. Г.*

Формат 70×100 1/16.
Гарнитура «PT Serif». Печать офсетная.
Усл. печ. л. 45,83. Тираж 200 экз.

Веб-сайт издательства: www.dmkpress.com