МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

В. О. Адаменко

ВСТУП ДО СПЕЦІАЛЬНОСТІ
ЛАБОРАТОРНИЙ ПРАКТИКУМ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для здобувачів ступеня бакалавра за освітніми програмами «Інтелектуальні технології радіоелектронної техніки», «Інформаційна та комунікаційна радіоінженерія» та «Радіотехнічні комп'ютеризовані системи» спеціальності 172 Електронні комунікації та радіотехніка

Видання друге, перероблене та доповнене

Київ
КПІ ім. Ігоря Сікорського
2023

Рецензенти:

Шпилька О. О., к.т.н., доц.

За редакцією автора.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського
(протокол № 4 від 19.01.2023 р.)
за поданням Вченої ради радіотехнічного факультету
(протокол № 15/2022 від 19.12.2022 р.)

Наведено інструкції до виконання лабораторних занять, завдання яких сприяти підвищенню мотивації до навчання студентів спеціальності 172 Електронні комунікації та радіотехніка радіотехнічного факультету КПІ ім. Ігоря Сікорського.

Вступ до спеціальності: лабораторний практикум [Електронний ресурс]: навч. посіб. для студ. спеціальності 172 Електронні комунікації та радіотехніка / В. О. Адаменко; КПІ ім. Ігоря Сікорського. — Електронні текстові дані (1 файл 4,2 МБайт). — Київ : КПІ ім. Ігоря Сікорського, 2023. — 127 с.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЕК — електронний компонент;

ЕРЕ — електрорадіоелемент;

ДП — Друкована плата;

РЕА — радіоелектронна апаратура

САПР — Система автоматизованого проектування

ВСТУП

Навчальний посібник містить 16 занять, які можна розділити на дві частини: 1–5 заняття пов’язані з вивченням електронних компонентів, системи автоматизованого проектування друкованих плат DipTrace, одного із способів виготовлення друкованої плати та наведено необхідні теоретичні відомості для монтажу електронних компонентів на виготовлену плату; 6–16 заняття присвячені вивченню програмно-апаратної платформи Arduino з використанням різних електронних компонентів та модулів. Заняття побудовані так, щоб стимулювати творчі здібності студентів та зацікавити їх подальшим навчанням за спеціальністю 172 Електронні комунікації та радіотехніка.

Для проведення занять за даним навчальним посібником не потрібно попередньої підготовки студентів та вивчення ними інших дисциплін. Хоча базові знання в програмуванні будуть не зайвими, проте їх можна компенсувати не значними додатковими поясненнями під час занять.

Зверніть увагу! Більшість наведених в посібнику визначень та описів подано в спрощеному вигляді і, відповідно, можуть нести певну неповноту чи містити неточності. Це зроблено для кращого їх розуміння студентами, які не мають специфічних знань, які дозволять їм розібратися в складних термінах чи процесах!

ЛАБОРАТОРНА РОБОТА № 1
СХЕМА ЕЛЕКТРИЧНА ПРИНЦИПОВА. ЕЛЕМЕНТНА БАЗА РАДІОЕЛЕКТРОННОЇ АПАРАТУРИ.

Мета роботи: ознайомитися зі схемою електричною принциповою, схемними позначеннями типових електронних компонентів та їх основними параметрами. Навчитися визначати параметри електронних компонентів, відшукувати та працювати з Datasheet.

Теоретичні відомості

Схема електрична принципова

Схема електрична принципова дозволяє отримати найповніше уявлення про склад та принцип роботи радіоелектронного пристрою. На цій схемі зображуються всі електронні компоненти (ЕК), з яких складається пристрій та зв'язки між ними (часто використовується термін «електрорадіоелементи» або ж скорочено ЕРЕ). ЕК на принципових схемах позначаються умовно у відповідності до вимог діючого стандарту. На схемах обов'язково вказуються пронумеровані по порядку літерно-цифрові позначення ЕК (наприклад *R*1, *R*2 тощо, більш докладно див. далі), причому нумерація виконується зверху до низу та зліва на право окремо для кожного типу позначень.

На рис. 1.1 наведено схему електричну принципову контролера освітленості в приміщенні.

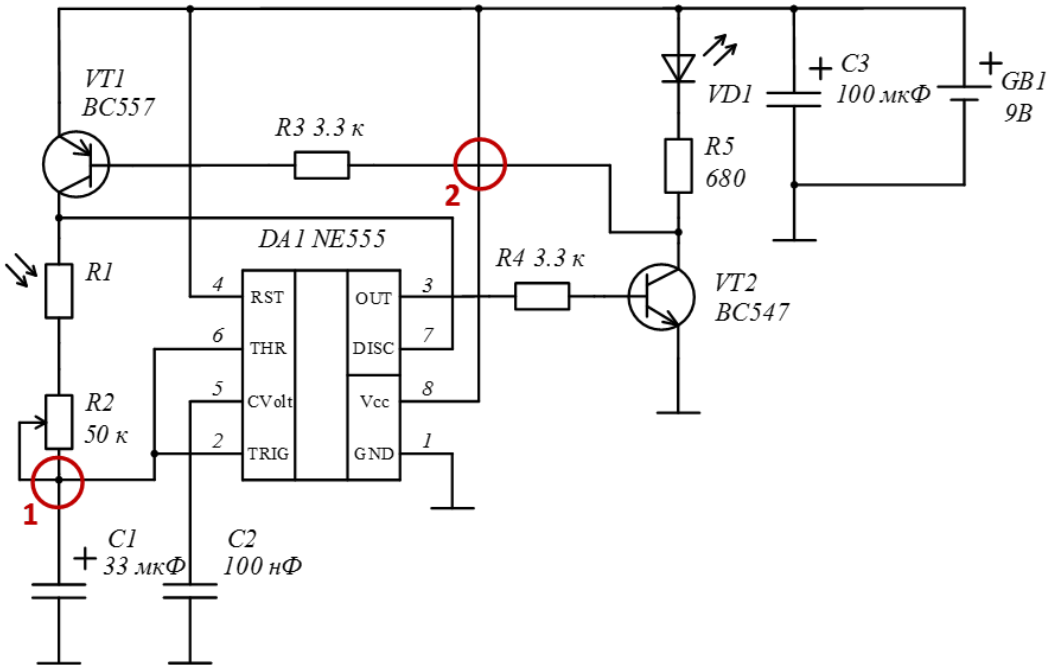


Рисунок 1.1 — Схема контролера освітленості

Лінії, які з’єднують схемні позначення ЕК відповідають за електричні з’єднання їх між собою. Тобто по таких з’єднаннях протікає електричний струм. Якщо ці зв’язки на схемі перетинаються і в точці перетину є крапка (рис. 1.1, червона мітка 1), то між цими лініями є електричний зв’язок. Якщо ж в точці перетину крапки немає (рис. 1.1, червона мітка 2), то електричний зв’язок відсутній. Більш докладно про напрям протікання та величину струму в таких з’єднаннях вивчається в курсах «Основи теорії кіл» та «Схемотехніка».

Типові електронні компоненти

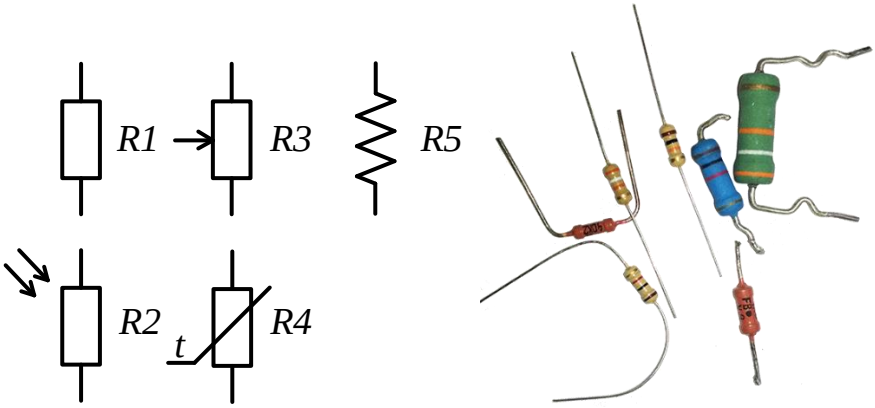
На сучасному етапі розвитку радіоелектроніки існує дуже велика кількість різноманітної елементної бази для створення радіоелектронних апаратів (РЕА). При чому одні і ті ж ЕК можуть виконуватися в різних корпусах, тобто їх зовнішній вигляд та спосіб встановлення на плату будуть відрізнятися. Наприклад, звичайні резистори можуть бути виконані з виводами для встановлення в отвори на друкованих платах (рис. 1.2б), та без виводів (часто їх називають SMD) для поверхневого монтажу (рис. 1.3а).

В межах курсу «Вступ до спеціальності» будуть використовуватися нижче перераховані типи ЕК.

Резистори

Схемне позначення та зовнішній вигляд резисторів зображено на рис. 1.2а. Резистори на схемах мають позначення «*R*». Резистори бувають постійного (рис. 1.2а, *R*1) та змінного опору (змінні або підстроювальні резистори), опір яких змінюється в залежності від положення механічного повзунка, рис. 1.2а, *R*3, також розрізняють фоторезистори (опір таких резисторів змінюється пропорційна до його освітленості, рис. 1.2а, *R*2) та терморезистори (опір змінюється в залежності від температури терморезистора, рис. 1.2, *R*4). На рис. 1.2а, схемне графічне позначення резистора *R*5 відповідає міжнародному стандарту.

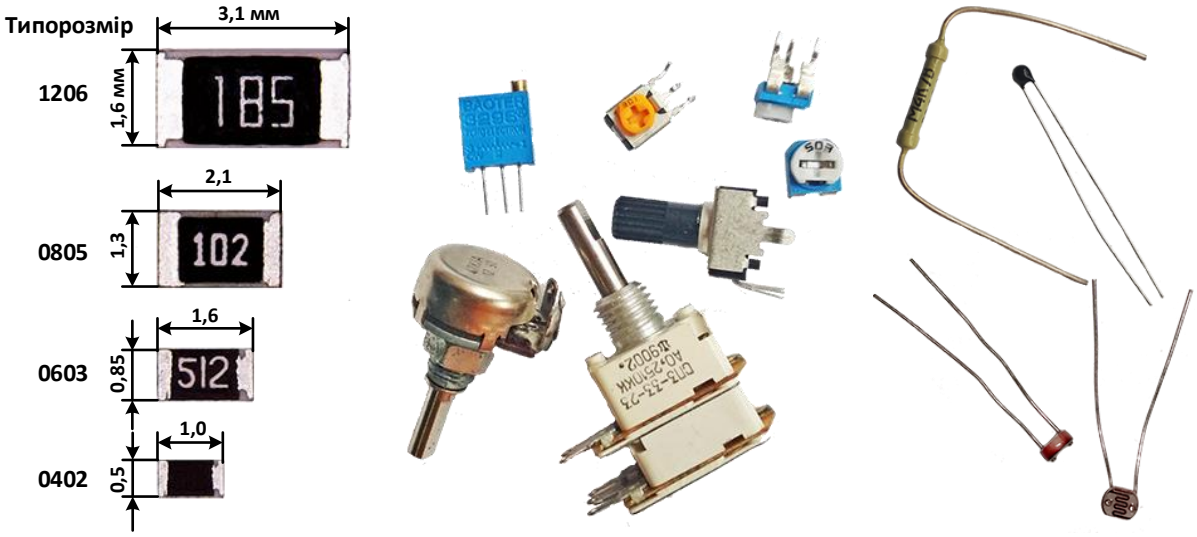
Основним параметром резисторів є значення їх опору, яке вимірюється в Омах. Опори резисторів, які випускаються промисловістю відповідають ряду номінальних значень [1]. Для позначення великих опорів використовують приставки кіло (к) та мега (М). Таким чином, якщо на схемі написано 50к, то це означає, що номінал резистора 50 кОм (кілоОм). Часто на схемах літеру «к» (або «М») ставлять замість десяткової крапки, таким чином позначення 3к3 відповідає значенню 3300 Ом або ж 3,3 кОм.



а б

Рисунок 1.2 — Схемне позначення (а) та зовнішній вигляд (б) резисторів

Раніше номінал вивідних резисторів вказувався на корпусі у вигляді відповідних цифр та літер, зараз в основному використовується кольорове маркування. Більш докладно про кольорове маркування розповідається в Додатку А. SMD резистори маркуються цифровими або цифро-літерними кодами. Найпоширеніші цифрові коди, де перші дві (або три) цифри вказують на значення опору, а остання цифра вказує на степінь, в яку потрібно піднести 10, щоб отримати множник, на який множиться значення опору. Наприклад, якщо на корпусі SMD резистора написано 152, то це означає $15 \cdot 10^2 = 1500$ Ом або ж 1,5 кОм.



а б в

Рисунок 1.3 — Зовнішній вигляд SMD (а), змінних, підстроювальних (б), термо- та фоторезисторів (в)

Резистори постійного опору неполярні, тобто немає різниці який вивід

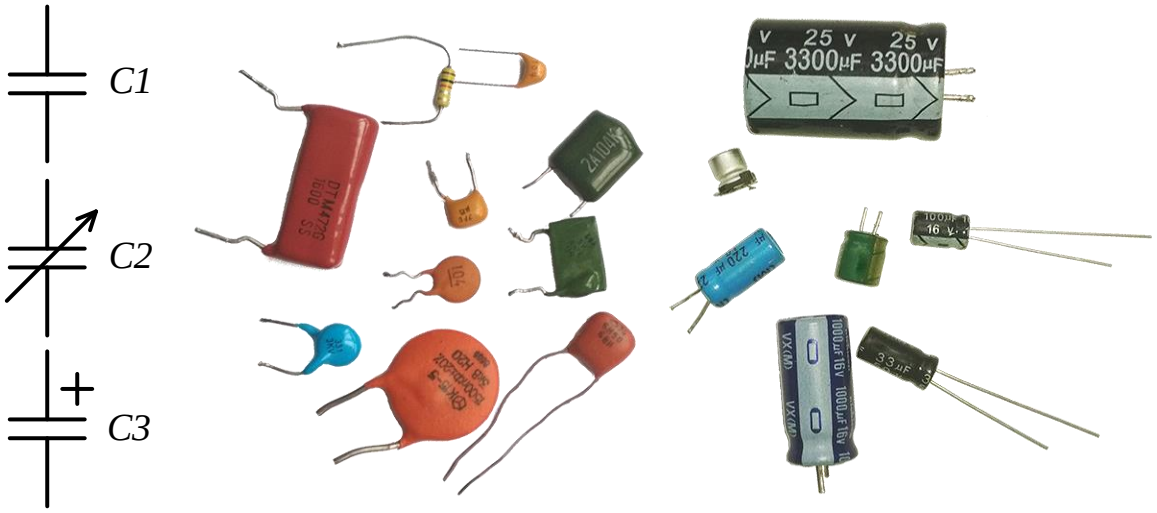
ЕК відповідає його схемному позначенню. При використанні змінних резисторів потрібно враховувати правила їх підключення, вивід позначений стрілочкою на схемі вказує на вивід резистора, який приєднаний до рухомого повзунка. Положення інших двох виводів впливає виключно на зручність їх експлуатації. Зовнішній вигляд резисторів змінного опору наведено на рис. 1.3б.

На рис. 1.3в зверху наведено приклад зовнішнього вигляду терморезисторів, а знизу — фоторезисторів.

Зверніть увагу! Термо- та фоторезистори зазвичай не мають маркування на корпусі, тому визначити їх тип дуже складно, якщо вони знаходяться поза пакувальною тарою.

Конденсатори

Схемне позначення та зовнішній вигляд конденсаторів зображено на рис. 1.4. Конденсатори мають на схемах позначення «*C*», бувають постійної (рис. 1.4 *C*1) та змінної (рис. 1.4 *C*2) ємності, також розрізняють полярні конденсатори (електролітичні) для яких на схемі використовують позначення полярності (рис. 1.4 *C*3).



а б в

Рисунок 1.4 — Схемне позначення (а) та зовнішній вигляд не полярних (б) та полярних (в) конденсаторів

Основним параметром конденсаторів є їх ємність, яка вимірюється в Фарадах, проте така ємність є великою, в радіотехніці використовуються конденсатори, ємність яких вимірюється у мікрофарадах (мкФ), нанофарадах (нФ) та пікофарадх (пФ). На схемах частіше у позначенні ємності не

вказують літеру Ф, а залишаю тільки префікс, так наприклад ємність в мікрофарадах може позначатися, як «мк» або «μ» (часто для електролітичних конденсаторів даний префікс пропускають, тобто може бути вказана виключно числове значення ємності); для нанофарадів — «н» або «n» (даний префікс вказується завжди); для пікофарадів — «п» або «р», проте в більшості випадків ємності в пікофарадах позначаються взагалі без літери.

Другим важливим параметром конденсатора є його робоча напруга. Особливо важливим цей параметр є для електролітичних конденсаторів, так як вони мають низьку, порівняно зі звичайними конденсаторами, робочу напругу.

Зверніть увагу! Заборонено експлуатувати конденсатори з перевищенням їх робочої напруги, адже це може призвести до їх вибуху!

Звичайні конденсатори не полярні, а електролітичні конденсатори мають полярність, тобто на корпусі є позначка «+» або «–», часто вивід «+» роблять довшим. Для таких конденсаторів в схемі обов'язково потрібно вказувати полярність підключення (на схемі традиційно вказують «+»).

Зверніть увагу! Полярні конденсатори потрібно підключати зі збереженням полярності, адже в іншому випадку вони можуть вийти з ладу.

На великогабаритних конденсаторах зазначають ємність та префікс, наприклад, 3300 μF відповідає ємності в 3300 мікрофарад, а 3n3 — 3,3 нанофаради або ж 3300 пікофарад. Якщо ж префікс не вказано, то ємність вказана в пікофарадах. Для малогабаритних конденсаторів на корпусі вказують три цифри: перші дві номінал, третя — степінь десяткового множника, так якщо на конденсаторі вказано 331, то це відповідає ємності в 330 пікофарад.

Зверніть увагу! *SMD* конденсатори не мають маркування!

Напівпровідникові електронні компоненти

Окремо варто відзначити напівпровідникові ЕК, для виготовлення яких використовуються напівпровідникові матеріали з різним типом провідності (більш докладно це вивчається в межах курсу «Елементна база»). До напівпровідникових ЕК відносяться діоди, транзистори, тиристори, мікросхеми тощо. Для напівпровідникових ЕК характерне вказання на схемах електричних принципових їх типу. Основні ж їх параметри можна знайти за вказаним типом у відповідних довідкових матеріалах. Для сучасних напівпровідникових ЕК такими довідковими матеріалами є *Datasheet*.

Даташіт, дейташіт (англ. Datasheet, data sheet) — документ, який

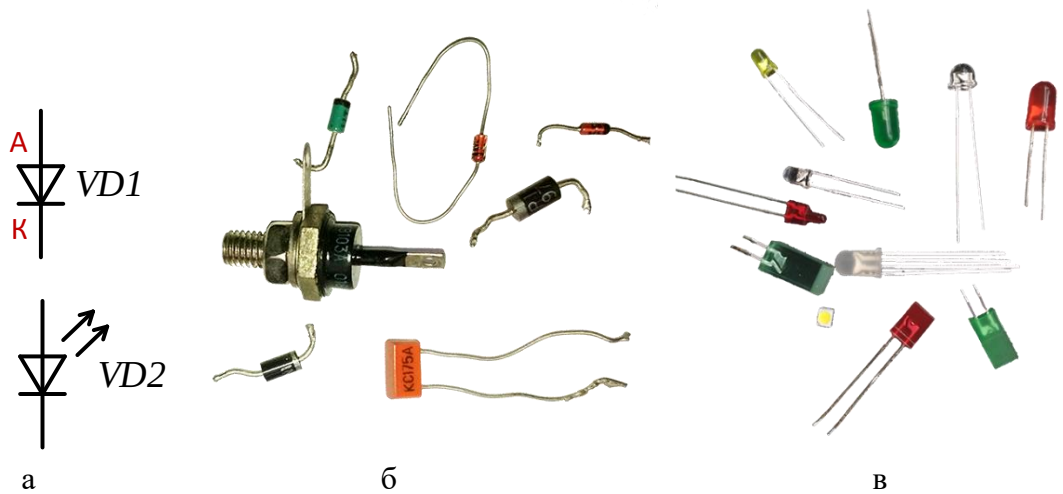
підсумовує технічні характеристики електронного компоненту, що призначений для використання інженером-конструктором. Як правило, *Datasheet* створюється компанією-виробником і починається зі сторінки з основними характеристиками, далі йдуть розширені характеристики, типові схеми включення, параметри друкованого рисунку тощо [2].

Вміння відшуковувати та працювати за *Datasheet* є дуже важливим для сучасного інженера!

Зверніть увагу! Більшість напівпровідникових ЕК має різне функціональне призначення кожного виводу. Тому виводи зі схеми повинні співпадати з виводами на корпусі ЕК. Розташування виводів на корпусі («розпіновка», від англ. *pinout*) наводяться в *Datasheet*.

Діоди та світлодіоди

Найпростішим напівпровідниковим ЕК є діод, завдання якого пропускати струм в одному напрямку. Діоди полярні елементи, тому їх підключення потребує уваги. На рис. 1.5а, *VD*1 зображено схемне позначення діода (літери червоного кольору тут вказані для зручності, на схемах не вказуються), його виводи називаються анод та катод. Основними параметрами діода є робоча напруга та струм. На корпусі діода вказується його тип (іноді використовується кольорове маркування) та позначається анод (в більшості випадків), рис. 1.5б.



а б в

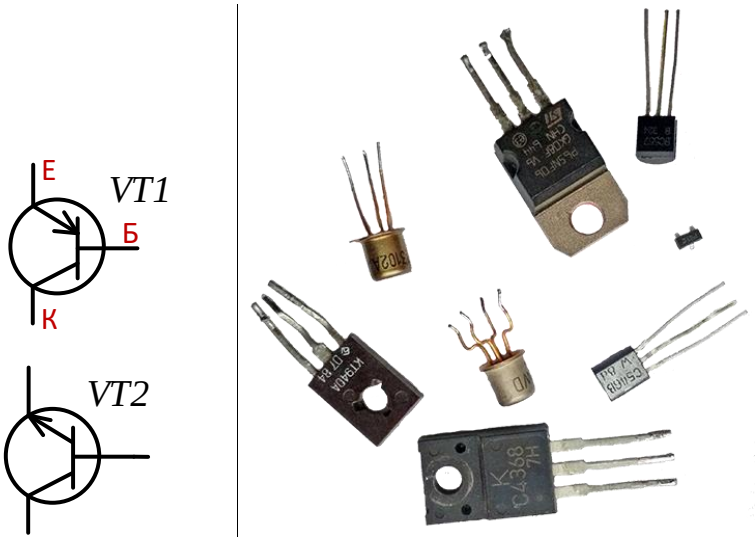
Рисунок 1.5 — Схемне позначення (а) та зовнішній вигляд діодів (б) та світлодіодів (в)

Окремо варто відзначити світлодіоди — це такі діоди, які випромінюють світло, коли через них протікає струм. Схемне позначення світлодіода наведено на рис. 1.5а, *VD*2. Світлодіоди використовуються для індикації та

освітлення. Вони бувають різних кольорів, розмірів та форм (рис. 1.5в). Основні параметри світлодіодів та правила їх ввімкнення розглядаються на наступних заняттях. Вивід світлодіода, який є анодом зазвичай довший за вивід катоду.

Транзистори

Транзистор — напівпровідниковий елемент, який дозволяє керувати струмом, який протікає через нього за допомогою зміни напруги (струму) на керуючому виводі. Транзистори поділяються на два великі класи: біполярні та польові. На рис. 1.6а, зображено схемне позначення біполярного транзистора прямої провідності або ж *p-n-p* транзистор (*VT*1) та зворотної провідності, або ж *n-p-n* транзистор (*VT*2). В чому різниця і як працюють транзистори вивчається на старших курсах. Виводи біполярного транзистора називаються «колектор», «емітер» та «база» (див. червоні позначки на рис. 1.6а). Традиційно схему розташування виводів транзистора можна знайти в *Datasheet*.



а б

Рисунок 1.6 — Схемне позначення (а) та зовнішній вигляд (б) транзиторів

Транзистори мають багато важливих параметрів, причому часто в залежності від схеми включення основними стають різні з них. Так, наприклад, при роботі в ключовому режимі основними параметрами є напруга та струм емітер–колектор. А якщо транзистор працює в режимі підсилення, то на перше місце виходить коефіцієнт підсилення, струм бази-емітер, характер вольт-амперної характеристики тощо.

Мікросхеми

Найскладнішим напівпровідниковим ЕК є мікросхеми. Різноманіття та

призначення різних типів мікросхем дуже значне. Мікросхеми не мають єдиного схемного позначення, хоча і існують загальні правила, які використовуються для позначення їх на схемі. На рис. 1.7а показано приклад схемного зображення мікросхеми, а на рис. 1.7б зовнішній вигляд корпусів деяких з них. Тип мікросхеми вказується на корпусі, там же повинна бути мітка, яка вказує на перший вивід мікросхеми. Нумерація виводів на схемі та в корпусі повинні співпадати (в більшості випадків).

В цілому мікросхеми поділяються на два великі класи: аналогові (на схемах позначаються як *DA*) та цифрові (*DD*). Проте зараз вже є мікросхеми, які не можна віднести ні до одного з цих класів. В залежності від призначення мікросхеми вона має відповідні параметри. В більшості випадків мікросхеми, на відміну від інших ЕК, не є взаємозамінними. Тобто використовувати при збиранні пристрою потрібно ту мікросхему, яка вказана на схемі. Виключенням є мікросхеми, які виконують однакові функції, але при цьому виготовляються різними фірмами. В такому випадку вони можуть бути повністю взаємозамінними або частково взаємозамінними (коли функціональне призначення мікросхем співпадає, але є відмінності в схемі розміщення виводів).

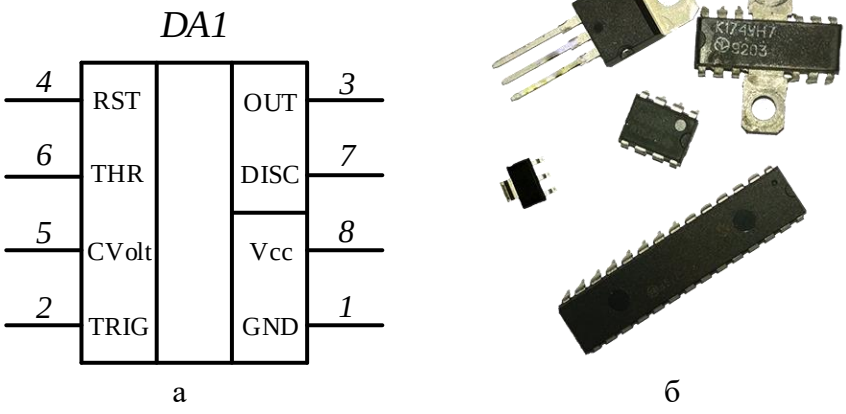


Рисунок 1.7 — Приклад схемного позначення (а) та зовнішнього вигляду (б) мікросхем

Часто в *Datasheet* крім параметрів та варіантів корпусу мікросхеми наводяться і типові схеми ввімкнення, що допомагає в розробленні апаратури на основі мікросхем.

На схемі також присутня батарейка та символ спільного проводу або ж

заземлення (рис. 1.8). Часто символів заземлення може бути декілька на схемі, як і в даному випадку. Проте всі частини схеми, які закінчуються символом заземлення по факту з'єднані одна з одно, тобто між ними є електричний зв'язок. Якщо ж на схемі присутні два типи позначення заземлення, то це більш складний випадок, який називається цифрове та аналогове заземлення і як їх з'єднувати між собою вивчається на старших курсах.

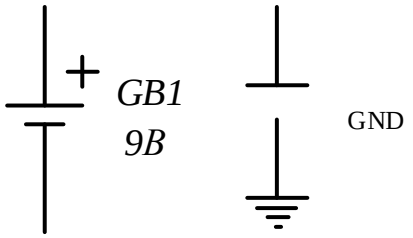


Рисунок 1.8 — Схемне позначення батарейки (*GB*)
та заземлення (*GND*)

Зверніть увагу! В процесі виконання робіт, описаних в даному посібнику будуть використовуватися і інші ЕК, проте їх опис буде наведено в теоретичних відомостях відповідних занять!

***Принцип роботи контролера освітленості**

Зважаючи на вищевикладене можна проаналізувати схему контролера освітленості (рис. 1.1), вона складається з:

- фоторезистора *R*1, опір якого змінюється під дією світла, яке потрапляє на його лінзу;
- підстроювального резистора *R*2, опір якого (між виводом зі стрілочкою та будь-яким іншим виводом) змінюється в залежності від кута повороту повзунка;
- резисторів *R*3 – *R*5, які мають постійний опір;
- електролітичних конденсаторів *C*1 та *C*3, які є полярними (електролітичні конденсатори);
- конденсатора *C*2, який є неполярним (керамічний конденсатор);
- мікросхеми–таймера *DA*1;
- транзисторів прямої провідності *VT*1 (або ж *p-n-p* транзистор) та зворотної провідності *VT*2 (*n-p-n* транзистор);
- світлодіода *VD*1;
- батарейки *GB*1.

Схема працює наступним чином: у момент включення конденсатор *C*1

розряджений і тому на 3 виводі мікросхеми *DA*1 високий рівень (відповідає напрузі живлення), який відкриває транзистор *VT*2 (тобто через перехід колектор-емітер протікає струм), відповідно світлодіод *VD*1 засвічується та відкривається транзистор *VT*1 і напруга живлення потрапляє на резистор *R*1. Конденсатор через резистори *R*1 та *R*2 починає заряджатися і при досягненні певної напруги переводить таймер *DA*1 у вимкнений стан (низький рівень на 3 виводі), транзистори *VT*1 та *VT*2 закриваються, світлодіод згасає, а конденсатор *C*1 починає розряджатися через резистори *R*1 та *R*2 та внутрішній транзистор таймера (вивід 7). Процес повторюється знову і знову, тобто світлодіод блимає. Так як швидкість заряду та розрядку конденсатора залежить від ємності конденсатора *C*1 та сумарного опору резисторів *R*1 та *R*2 то, відповідно, ця швидкість змінюється в залежності від інтенсивності світла, яке потрапляє на фоторезистор *R*1 (Чим більше світла потрапляє на фоторезистор, тим нижчий у нього опір, і відповідно, конденсатор заряджається швидше).

Порядок виконання роботи

1. Отримати у викладача набір ЕК.

2. Ознайомитися з теоретичними відомостями та розібратися з виданими електронними компонентами.

3. Розділити ЕК за групами та визначити їх основні параметри.

4. Для напівпровідникових ЕК знайти відповідні *Datasheet* та визначити їх схему розташування виводів.

5. Продемонструвати викладачу результат розподілу за групами та визначення параметрів ЕК.

6. Відібрати ті ЕК, які необхідні для реалізації схеми рис. 1.1

7. Повернути викладачу роздані ЕК.

Контрольні питання

1. Яке умовне графічне позначення на схемі електричній принциповій мають резистори, конденсатори, світлодіоди, транзистори тощо?

2. Чим на схемі електричній принциповій відрізняється позначення *n-p-n* та *p-n-p* транзистора?

3. Як на схемі електричній принциповій розрізнити емітер, колектор та базу транзистора?

4. Як визначити правильну схему розташування виводів світлодіода?

5. Який вигляд має корпус *TO*-92 та *DIP*-8?

6. Як визначити номінал резистора за його кольоровим маркуванням?

7. Як визначити номінал конденсатора за цифровим кодом?

*8. Як працює контролер освітленості?

ЛАБОРАТОРНА РОБОТА № 2
ДРУКОВАНІ ПЛАТИ. СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ

Мета роботи: ознайомлення з призначенням та методами виготовлення друкованих плат. Вивчення системи автоматизованого проектування *DipTrace*. Перенесення схеми до модулю *Schematic*

Теоретичні відомості

Друковані плати

Друкована плата, ДП (англ. *Printed circuit board, PCB*) — пластина, виконана з діелектрика (склотекстоліт, текстоліт, гетинакс, ситал тощо), на якій або всередині якої сформований хоча б один шар з провідними доріжками. На друковану плату монтуються електронні компоненти, які з'єднуються своїми виводами з елементами провідного рисунка паянням (рідше зварюванням), у результаті чого складається електронний модуль — друкований вузол [3].

Друковані плати бувають: односторонні — провідники розміщуються з одного боку діелектричної пластини, двосторонні — провідники розміщуються з двох боків пластини та багатошарові — провідники розміщуються з боків та в товщі діелектрика.

Типові методи виготовлення друкованих плат вивчаються на старших курсах в межах дисциплін «Проектування друкованих плат» або «Конструювання РЕА».

Під час виконання лабораторних робіт буде використано субтрактивний метод хімічного травлення, який полягає в тому, що рисунок провідників буде отримано шляхом витравлювання в реактивах міді з незахищених ділянок склотекстолітової заготовки з нанесеною мідною фольгою.

Типовий процес проектування друкованих плат складається з декількох етапів:

- Створення схеми електричної принципової в системі автоматизованого проектування (САПР);
- Закріплення за кожним схемотехнічним компонентом креслення відповідного корпусу ЕК (патерну) з прив'язуванням виводів;
- Визначення майбутніх габаритів друкованої плати та інших технологічних параметрів (діаметри отворів, параметри друкованих провідників тощо);

– Ручне або автоматичне розміщення посадочних місць компонентів на друкованій платі;
– Ручне або автоматичне трасування провідників (процес заміни електричних з'єднань провідниковими доріжками);
– При потребі повтор двох попередніх пунктів до досягнення оптимального результату;
– Перевірка на помилки та підготовка отриманих файлів до виробництва.

Етапи виробництва друкованої плати в домашніх умовах:

– Нанесення дзеркального відображення провідникового рисунку на спеціальний папір (або його замінник), зазвичай виконується за допомогою лазерного принтеру;
– Перенесення рисунку з паперу на спеціально підготовлений фольгований склотекстоліт (за допомогою праски ☺);
– Витравлювання міді з незахищених ділянок за допомогою хлорного заліза ($FeCl_3$) або іншими реактивами;
– Зняття захисного покриття;
– Висвердлювання отворів в потрібних місцях;
– Монтаж електронних компонентів на плату;

Звичайно це спрощених процес виготовлення друкованої плати, який можна просто повторити в домашніх умовах. На виробництві все відбувається трохи складніше, але це вже вивчається в дисципліні «Проектування друкованих плат».

Система автоматизованого проектування *DipTrace*

Система автоматизованого проектування (САПР) — автоматизована система, призначена для автоматизації технологічного процесу проектування виробу, результатом якого є комплект проектно-конструкторської документації, достатньої для виготовлення та подальшої експлуатації об'єкта проектування [4].

Існує багато САПР для проектування друкованих плат: *OrCAD*, *P-CAD*, *Altium Designer*, *DipTrace* тощо. Порівняно простою, але при цьому повноцінною САПР є *DipTrace*, який як і більшість професійних САПР складається з наступних модулів:

– *Schematic Capture* — використовується для створення та редагування принципових схем;
– *PCB Layout* — модулю проектування друкованої плати;

– *Component Editor* — модуль для створення та редагування компонентів;
– *Pattern Editor* — модуль для створення корпусів ЕК.

Офіційну інструкцію користувача програмою розміщено на сайті компанії «Новарм» [5].

Порядок виконання роботи

1. Перенести схему контролера освітленості рис. 1.1 до модулю *Schematic* програми *DipTrace*, виконавши наведену нижче інструкцію.

Запустити *Schematic* можна в меню «Пуск», в розділі *DipTrace*.

Перед початком роботи варто ознайомитися з інтерфейсом програми (рис. 2.1) та провести попередні налаштування, змінивши одиниці вимірювання на міліметри: Меню «Вигляд/Одиниці вимірювання».

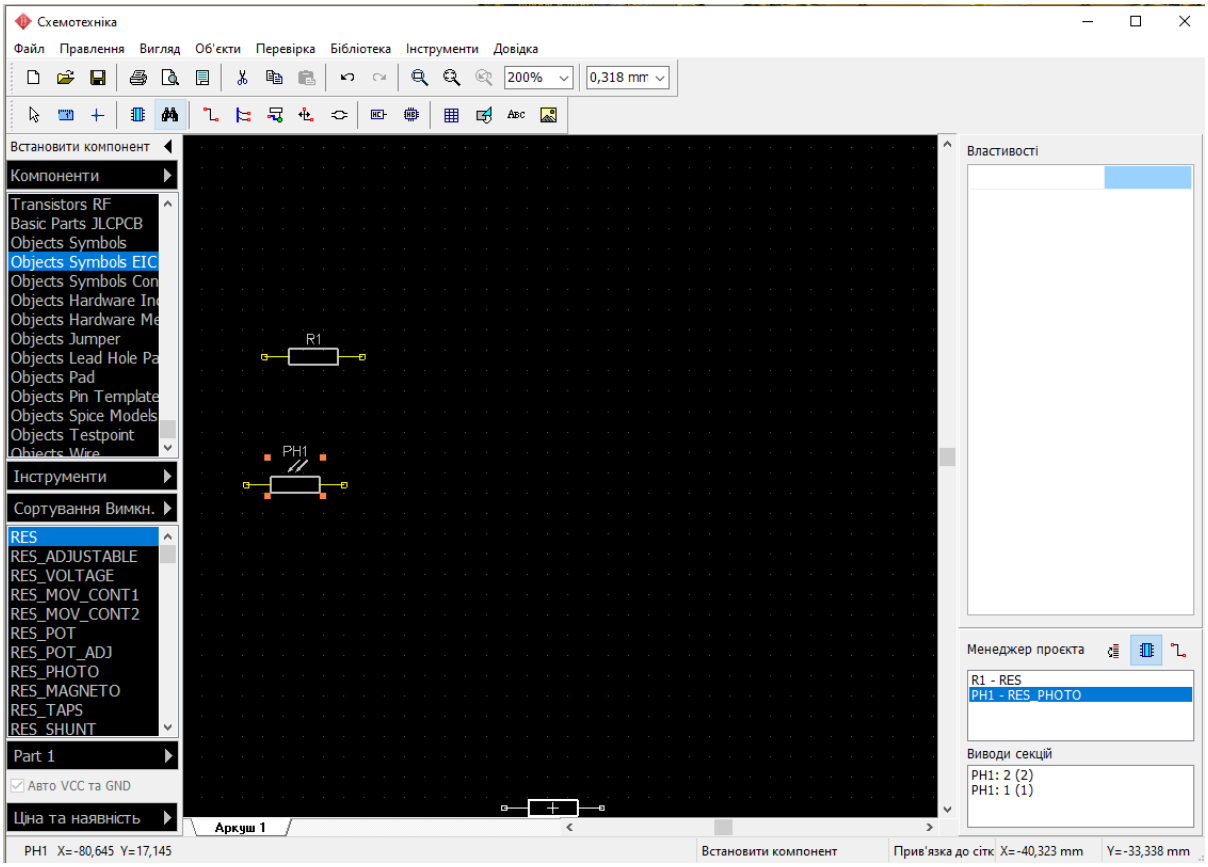


Рисунок 2.1 — Зовнішній вигляд модуля «Схемотехніка»

Компоненти, необхідні для створення схеми, обирати з бібліотеки «*Objects Symbols EIC*» (якщо не вказано іншу бібліотеку), так як там схемні зображення максимально наближені до стандартних графічних зображень

(правда повністю ним не відповідають, що варто враховувати при подальшому використанні програми). До кожного схемного компоненту потрібно приєднати відповідний корпус (точніше зображення корпусу).

Зверніть увагу! Всі схемні компоненти можна обертати навколо осі та віддзеркалювати за допомогою контекстного меню (клік правою кнопкою миші на компоненті). Для обертання можна використати клавішу — «*R*». Також всі компоненти та лінії зв'язку можна переміщати звичайним перетягуванням. В процесі роботи нумерація міток може не співпадати з наведеною на рис. 1.1, адже вона визначається порядком додавання компонентів до програми, тому не варто поки звертати на неї увагу.

Для резисторів *R*3–*R*5 обрати схемний компонент «*RES»*, для цього в панелі «Встановлення компонентів» обрати загальну бібліотеку «*Objects Symbols EIC*» (клік мишкою по назві) та зі списку нижче обрати «*RES*» та встановити схемний компонент, який автоматично потягнеться за курсором в будь-якому місці загального вікна. Далі потрібно його прив'язати (тобто забезпечити зв'язки між виводами схемного компонента та контактними площинками посадочного місця його корпусу) до корпусу «*RESAD1120W55L680D260»*, для цього необхідно двічі клацнути на схемному компоненті лівою кнопкою миші та натиснути «Прив'язування до корпусу», в отриманому вікні (рис. 2.2) в колонці «Бібліотека корпусів» обрати *«Res Axial»*, а нижче в «Корпуси» «*RESAD1120W55L680D260»*, в полі «Таблиця прив'язування» необхідно вказати 1 – 1 та 2 – 2 (Вивід – Номер контактної площинки), якщо зв'язки не з'явилися автоматично.

Для кожного схемного компонента необхідно вказати номінал або тип, та відобразити його на схемі. Для цього клік правою кнопкою миші на компоненті, обираємо «Властивості» та на вкладці «Параметри», див. рис. 2.3 записуємо номінал у відповідному полі «Значення». Якщо це необхідно, то в цьому ж полі вказуються одиниці вимірювання номіналу (кОм, мкФ тощо). Можна змінити місце розміщення цих написів (вкладка «Написи»). Також переміщення написів можна проводити безпосередньо у головному вікні, для цього потрібно натиснути «F10» та обрати мишкою потрібний напис. Переміщення відбувається звичайним перетягуванням, а обертання натисканням «R» або пробілу.

Якщо після внесення даних в поле «Значення» в робочій області не відображаються номінали, то необхідно змінити глобальні налаштування програми, для цього в основному Меню вибираємо «Вигляд/Написи компонентів» і ставимо «галочку» біля поля «Значення».

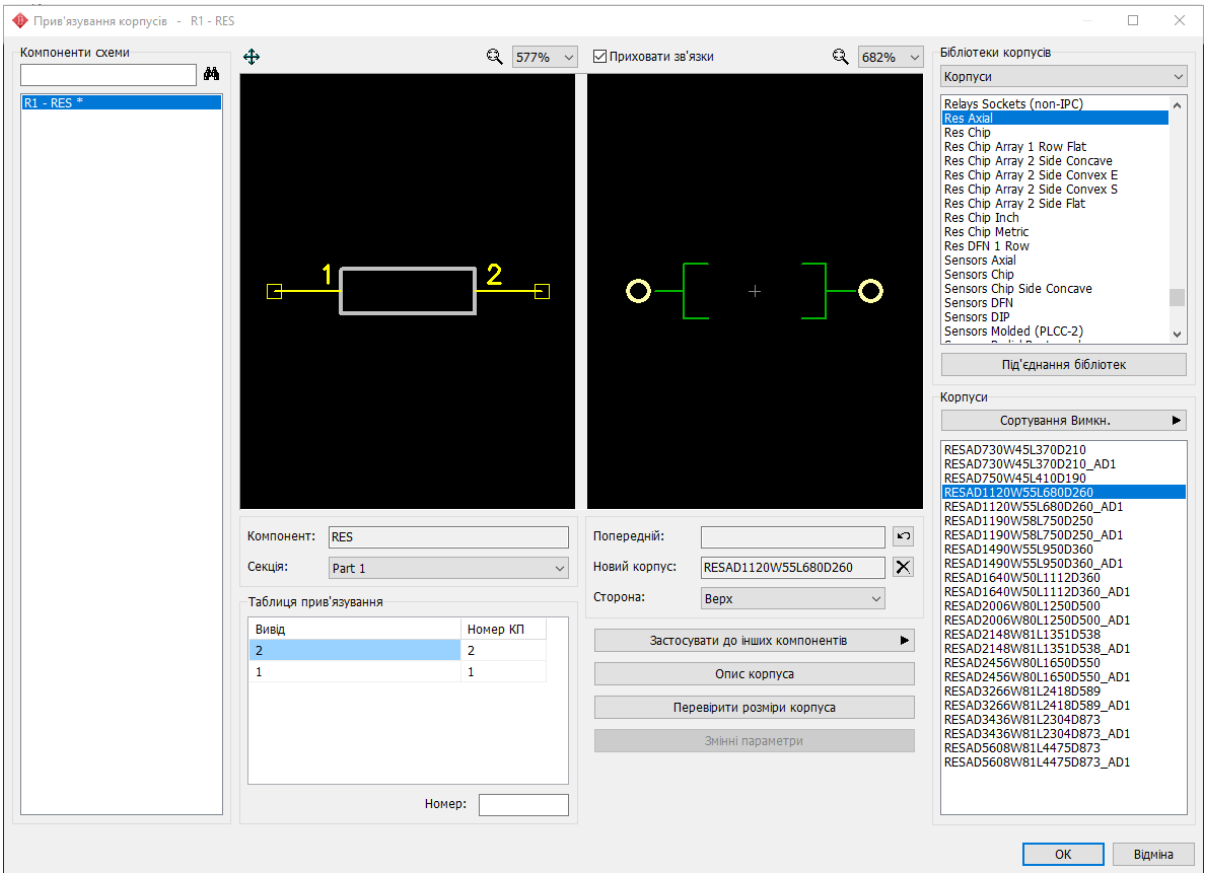


Рисунок 2.2 — Вікно прив’язування корпусів

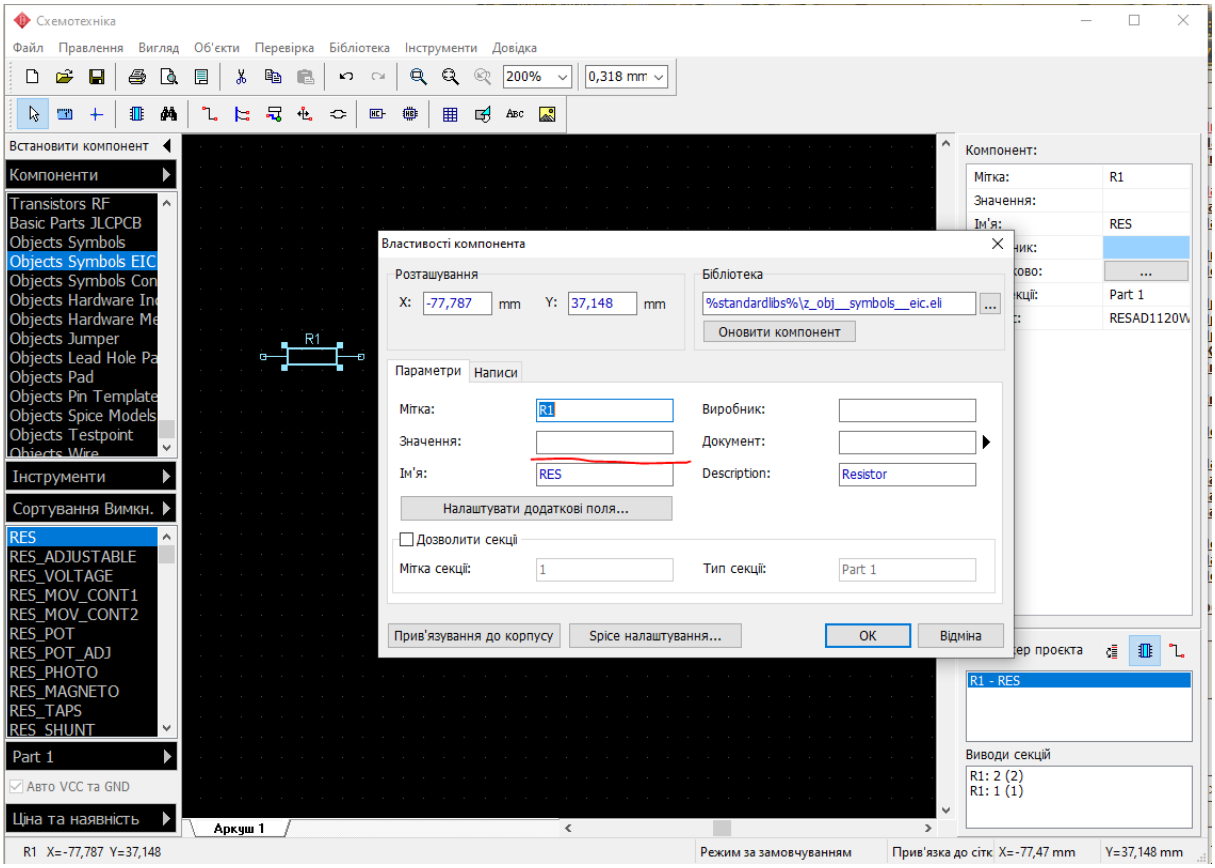


Рисунок 2.3 — Вікно властивостей компонента

Відповідно для фоторезистора *R*1 обрати «*RES_PHOTO*» та корпус «*Res Axial/RESAD730W45L370D210*», також для фоторезистора необхідно змінити «Мітку» з «*PH*» на «*R*». Для резистора *R*2 обрати *RES_POT* та корпус «*Potentiometers Trimmers THT/BOURNS_3306F*» та виконати прив'язування виводів в таблиці прив'язування наступним чином: 2 – 3, 3 – 2, 1 – 1, зміна відбувається просто записуванням необхідного номеру до відповідного поля.

Для транзистора *VT*1 обрати схемне позначення *PNP* з бібліотеки «*Objects Symbols*» та корпус «*TO-92 TO-226 Staggered Leads/TO-92-127P473H752-3 STA*» з наступним прив'язуванням виводів: *B* – 2, *E* – 1, *C* – 3. Для транзистора *VT*2 — «*Objects Symbols/NPN*» та корпус «*TO-92 TO-226 Staggered Leads/TO-92-127P473H752-3 STA*» з наступним прив'язуванням виводів: *B* – 2, *E* – 1, *C* – 3, рис. 2.4. Для транзисторів необхідно змінити мітку з «*Q*» на «*VT*».

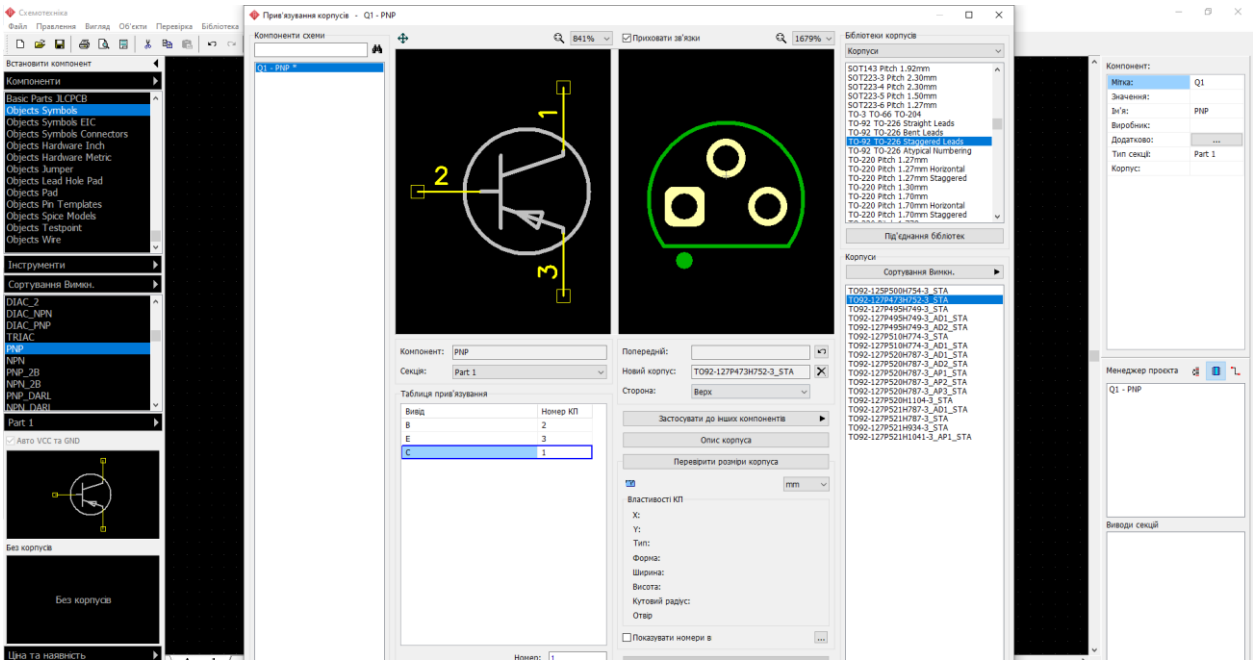


Рисунок 2.4 — Прив'язування транзистора до корпусу

Для конденсаторів *C*1 та *C*3 обрати символ «*Objects Symbols EIC/CAP_POLARIZED*» та корпус «*CAP Electrolytic Radial/CAPPR250W60D680H200*», виводи повинні прив'язатися автоматично, якщо цього не сталося, то відповідно необхідно прив'язати позитивний вивід до квадратної контактної площинки, а негативний — до овального.

Для конденсатора *C*2 обрати символ «*Objects Symbols EIC/CAP*» та корпус «*CAP Radial Model/CAPPRR508W50L730T480H1080*».

Символ батареї живлення «*Objects Symbols EIC/BATTERY*» можна

прив’язати до корпусу *«Con Rect Board In p2.50mm/JST_02DB-6S»*. Перевірити наявність автоматичного прив’язування виводів символу до відповідних контактних майданчиків. Якщо зв’язків немає, то з’єднати позитивний вивід з прямокутним майданчиком, а негативний — з овальним майданчиком.

Світлодіод: символ *«Objects Symbols EIC/DIO_LIGHT»* та корпус *«LED Radial Cylindrical/LED-3mm Cilindrical RED»* та змінити мітку з «*D*» на «*VD*». Перевірити прив’язування виводів, так щоб анод світлодіода був прив’язаний до прямокутного контактного майданчику, а катод — до овального.

Мікросхему знайти готову, за допомогою пошуку компонентів (значок у вигляді бінокля в верхньому меню вікна), область пошуку «Всі бібліотеки», а в поле «Ім’я» введіть *NE*555, при цьому із знайденого списку компонентів слід обрати корпус *DIP*-8, рис. 2.5, змінити мітку на *DA*1.

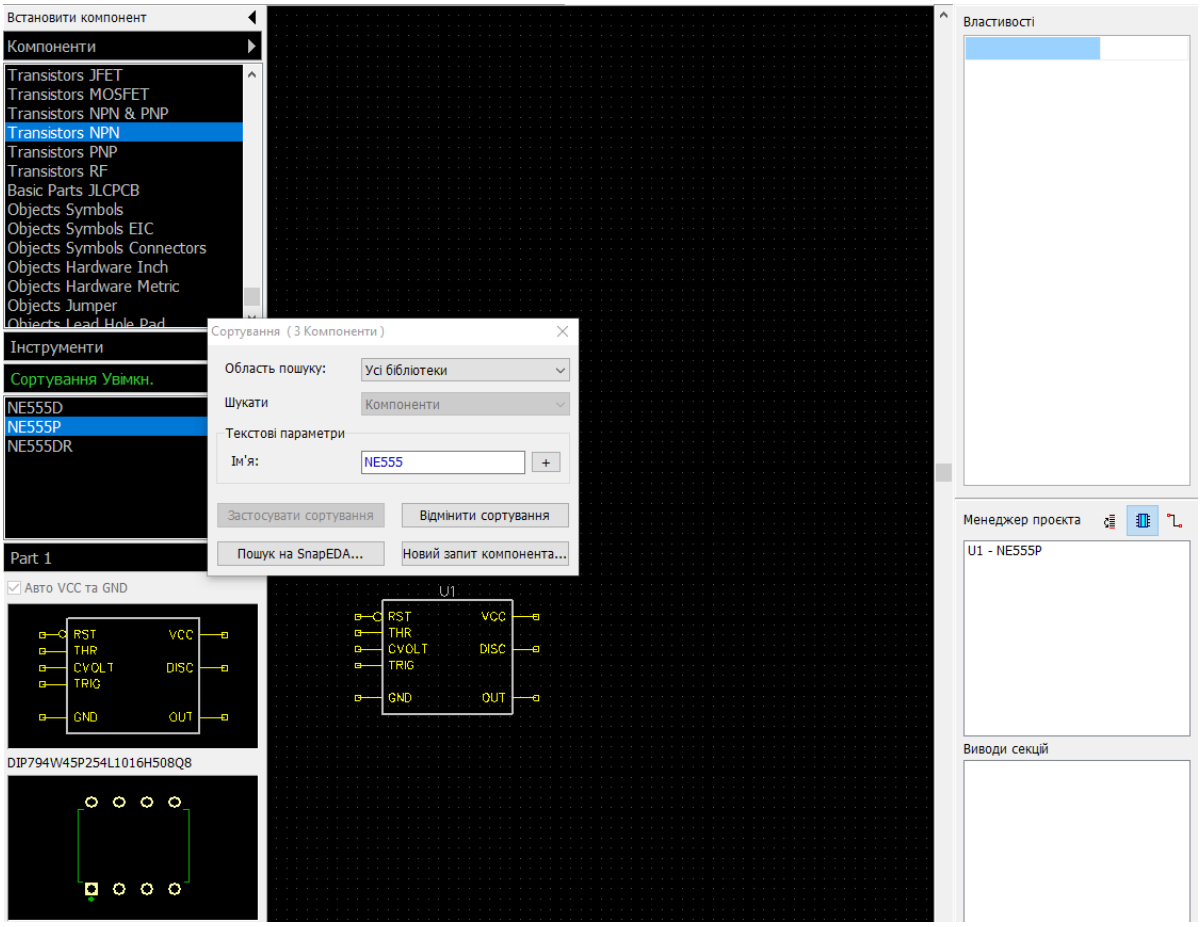


Рисунок 2.5 — Пошук та вибір мікросхеми.

Для символу заземлення обрати *«Objects Symbols/GND»*, корпус для нього відсутній.

Після того, як всі компоненти винесені на екрані — необхідно їх розмістити у відповідності до схеми електричної принципової рис. 1.1 та з'єднати провідниками всі виводи, для цього обрати на панелі інструментів «Встановлення зв'язків» та курсором з'єднати між собою необхідні виводи. Пересування компонентів та зв'язків можливе на будь-якому етапі роботи.

Наступним етапом є перенумерація міток у відповідності до створеної схеми. Це робиться автоматично, виконанням команди «Інструменти/Перенумерація міток». У вікні обрати «Верхній лівий кут» та Напрямок: «Стовпчики».

Також необхідно вимкнути загальне відображення нумерації виводів «Вид/Номера виводів/Приховати». Або ж викликавши контекстне меню компоненту. Для мікросхеми нумерацію виводів потрібно увімкнути окремо. В результаті Ви повинні отримати схему, схожу на зображену на рис. 2.6.

Зверніть увагу! Часто для транзисторів необхідно вимикати нумерацію виводів окремо. Також варто звернути увагу на правильність розміщення написів біля компонентів.

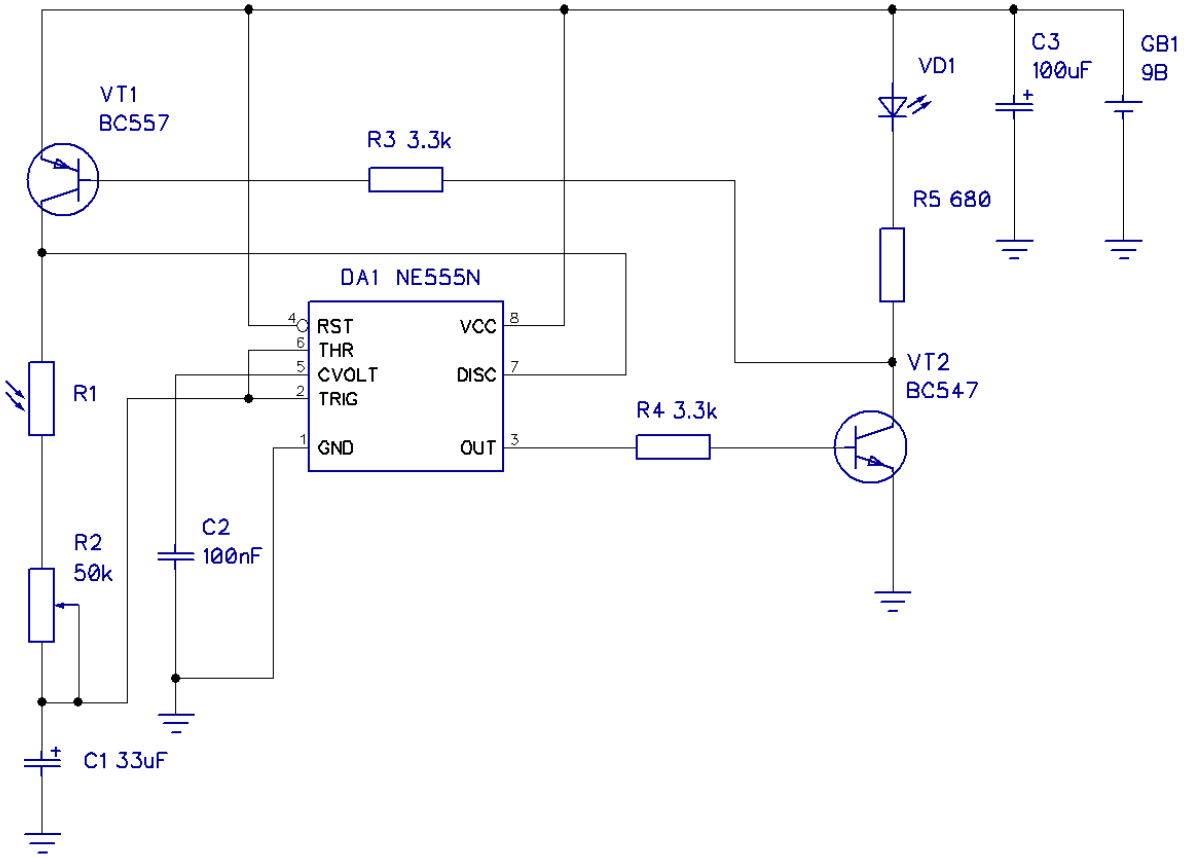


Рисунок 2.6 — Приклад схеми в DipTrace

2. Збережіть розроблену схему на диск *E*: в своїй папці. Також експортуйте свою схему в формат *pdf*, зробити це можна за допомогою віртуального принтеру, для цього тиснемо друк та в полі імені принтеру обираємо той, який в назві містить *pdf*.

Контрольні питання

1. Які модулі містить програма *DipTrace* та для чого вони використовуються?

2. Що таке друкована плата?

3. Назвіть основні етапи типового процесу проектування друкованої плати?

4. Як приєднати корпус до схемного елементу?

5. Як правильно розмістити написи біля ЕК на схемі електричній принциповій?

ЛАБОРАТОРНА РОБОТА № 3
ПРОЕКТУВАННЯ ДРУКОВАНОЇ ПЛАТИ

Мета роботи: Вивчення системи автоматизованого проектування *DipTrace*. Створення друкованої плати в модулі *PCB Layout*

Теоретичні відомості

Важливим етапом проектування друкованих плат є визначення оптимальних габаритів друкованої плати та параметрів друкованого рисунку. До останніх належить: мінімальна ширина провідника, мінімальна відстань між елементами друкованого рисунку, мінімальні діаметри отворів, розміри контактних майданчиків тощо. Проте все це вивчається на старших курсах в межах дисципліни «Проектування друкованих плат».

Для даного заняття розміри плати визначає викладач. Параметри ж друкованого рисунку обираються з погляду технології виготовлення плати в домашніх умовах, а саме: метод хімічного травлення, друк рисунку за допомогою лазерного принтера та перенесення за допомогою термічного пресування. Таким чином мінімальна ширина доріжки повинна бути 0,8 мм, відстань між елементами друкованого рисунку не менше 0,3 мм, відстань до межі плати — не менше 1 мм, рис. 3.1.

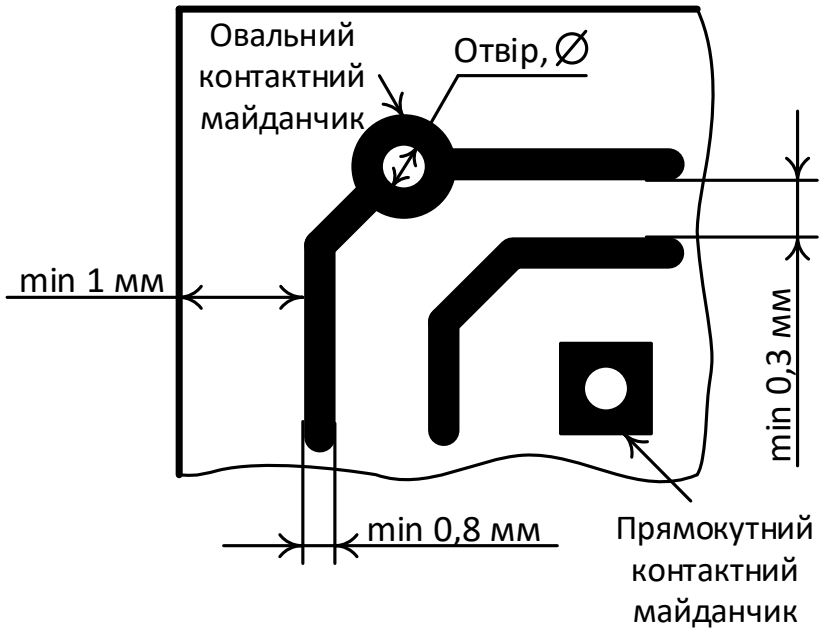


Рисунок 3.1 — Параметри друкованого рисунку

Налаштування діаметрів отворів та розмірів контактних майданчиків провести згідно таблиці, **зверніть увагу**, що деякі ЕК мають як овальні так і прямокутні контактні майданчики, для них у відповідних комірках вказано

по два типи контактних майданчиків.

Таблиці 3.1 — Параметри контактних майданчиків та отворів

Елемент	Контактний майданчик, мм			Отвір, мм
	Ширина	Висота	Форма	
Резистори R1, R3–R5	2	2	Овал	0,8
Резистор R2	2	2	Овал	0,8
	1,9	1,9	Прямокутник	0,8
Конденсатори C1, C3	2	2	Овал	0,8
	1,9	1,9	Прямокутник	0,8
Конденсатор C2	2	2	Овал	0,8
Світлодіод VD1	2	2	Овал	0,8
	1,9	1,9	Прямокутник	0,8
Батарейка GB1	2	2	Овал	0,8
	1,9	1,9	Прямокутник	0,8
Транзистори VT1 та VT2	1,5	1,5	Овал	0,6
	1,3	1,3	Прямокутник	0,6
Мікросхема DA1	2	2	Овал	0,8
	1,9	1,9	Прямокутник	0,8

Порядок виконання роботи

1. Відкрити схему, розроблену на попередньому занятті. Для створення друкованої плати необхідно обрати в основному меню Файл/Перетворити на плату. У вікні, яке з'явиться обрати «Використовувати схемотехнічні правила».

2. У відкритому вікні *PCB Layout* встановити одиниці вимірювання в міліметрах (аналогічно до того, як це було зроблено в *Schematic*). Створити межі плати розміром заданим викладачем, для цього зайти в «Об’єкти/Координати вершин». У вікні встановити відмітку в полі «Створення прямокутної плати» та ввести відповідні значення ширини та висоти плати.

3. В межах плати розмістити всі компоненти так, щоб вони мали мінімальну відстань ліній зв’язку (сині лінії), періодично натискаючи *F*12 для

автоматичного перерозподілу зв’язків. Обертати компоненти можна натисненням клавіши «*R*». Бажано уникати значного перетину ліній зв’язку, адже в іншому випадку автотрасувальнику буде складно провести трасування зв’язків. Приклад розміщення на рис. 3.2.

Зверніть увагу! У Вас розміщення може відрізнятися від наведеного на рисунку!

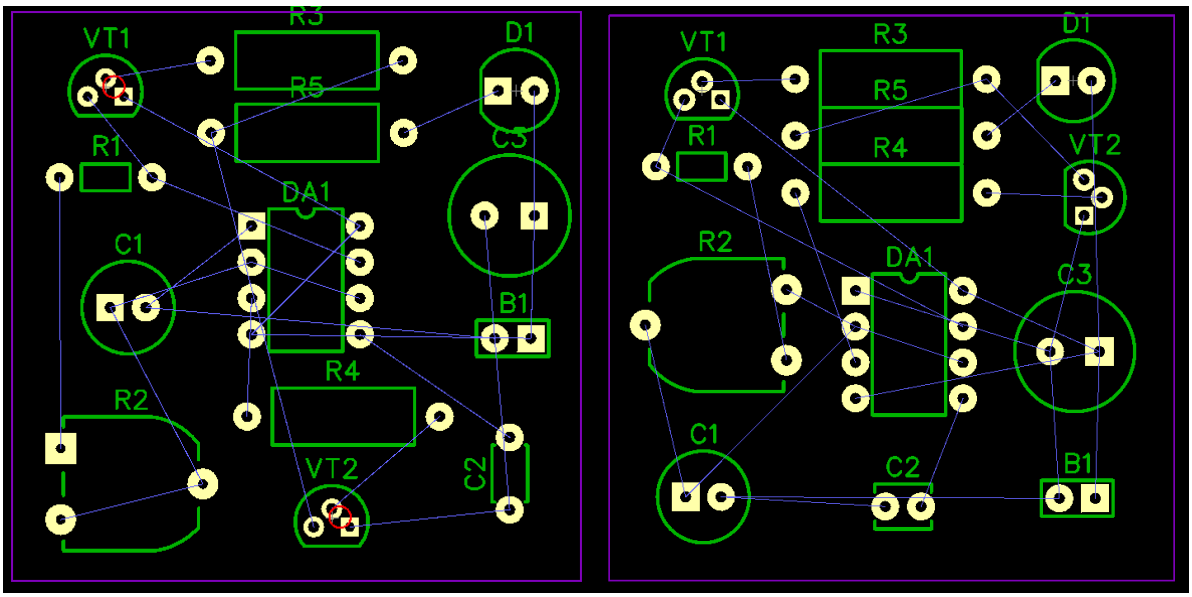


Рисунок 3.2 — Приклад розміщення компонентів на платі.

4. Перед проведенням автотрасування потрібно провести налаштування параметрів елементів друкованого монтажу, а саме змінити мінімальну ширину доріжок в меню «Трасування/Параметри трасування:» ширина трас 0,8 мм, між трасами 0,3 мм, відстань до краю плати — 1 мм. Крім того, потрібно змінити розміри контактних майданчиків, згідно табл. 3.1. Для цього необхідно клікнути правою кнопкою миші на контактному майданчику та обрати «Властивості виводу/Властивості площинок корпусу», ввести необхідні зміни та застосувати для всіх подібних. Однотипні корпуси можна змінювати одночасно, для цього перед внесенням змін необхідно їх виділити.

На останок необхідно налаштувати параметри автотрасувальника так, щоб в результаті отримати односторонню друковану плату, а саме обрати «Трасування/Параметри автотрасування» на закладці налаштування поставити галочку «Викор. пріоритетні напрями у шарах». Обрати верхній шар і змінити «Пріоритет» на «Вимкнений».

5. Провести автотрасування провідників, для цього необхідно натиснути кнопку «Старт автотрасування».

В більшості випадків провести повне трасування (коли відсутні сині зв’язки) з першого разу практично не можливо. Тому, якщо в результаті автотрасування у Вас залишилися сині зв’язки то уважно проаналізуйте наявні траси та подумайте, як змінити розміщення чи орієнтацію компонентів, щоб автотрасувальник зміг провести відсутні траси. Для внесення кардинальних змін в розміщення або орієнтацію компонентів перш за все потрібно детрасувати наявні зв’язки, для цього обрати «Трасування/Детрасувати все». Після цього виконати потрібні зміни в розміщення компонентів та провести повторне трасування. Цей процес може повторюватися декілька разів. В результаті Ви повинні отримати плату виключно з трасами, тобто без синіх зв’язків, рис. 3.3.

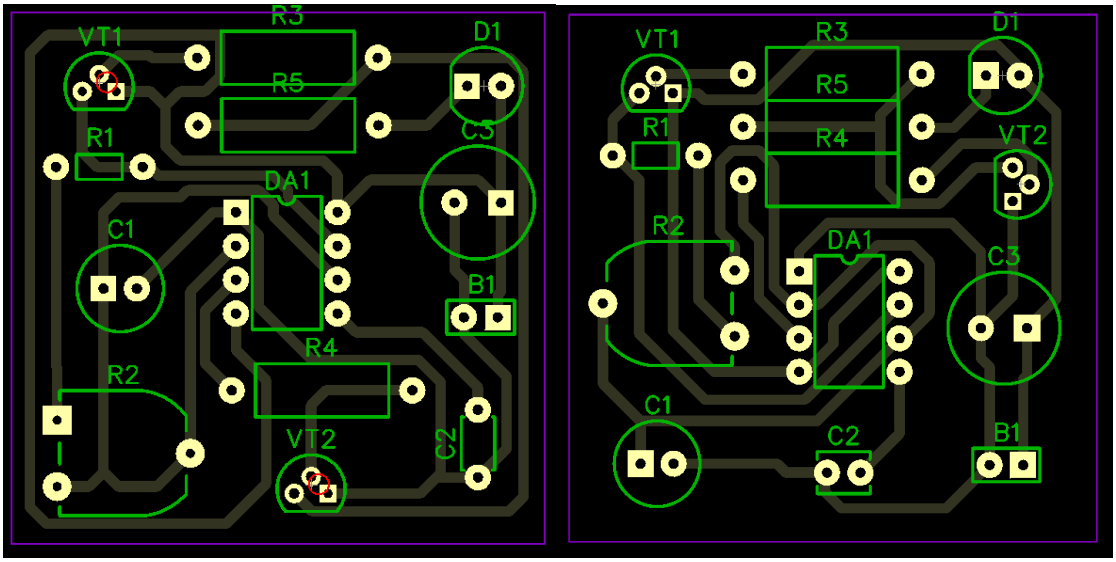


Рисунок 3.3 — Приклад вірно відтрасованої плати.

Часто для покращення естетичного вигляду розміщення компонентів або доріжок (трас) доцільно провести незначне коригування їх положення. Компоненти навіть з готовими трасами можна переміщувати або обертати, проте варто бути уважним, адже при цьому можна створити випадкові непотрібні електричні зв’язки. Для корегування безпосередньо доріжок необхідно перейти в нижній шар, та пересувати проблемні доріжки за допомогою курсору миші. Перевірити чи не було створено непотрібних зв’язків можна за допомогою інструменту перевірки помилок трасування: «Перевірка/Відобразити помилки трасування» або натиснувши F9.

6. За результатом роботи по трасуванню плати Ви повинні створити два PDF файли. Перший повинен містити виключно доріжки та контактні майданчики (рис. 3.4) та називатися «Плата», а другий маркування ЕК (рис. 3.5)

з назвою «Вузол». Виконується це за допомогою роздрукування на віртуальному PDF принтері.

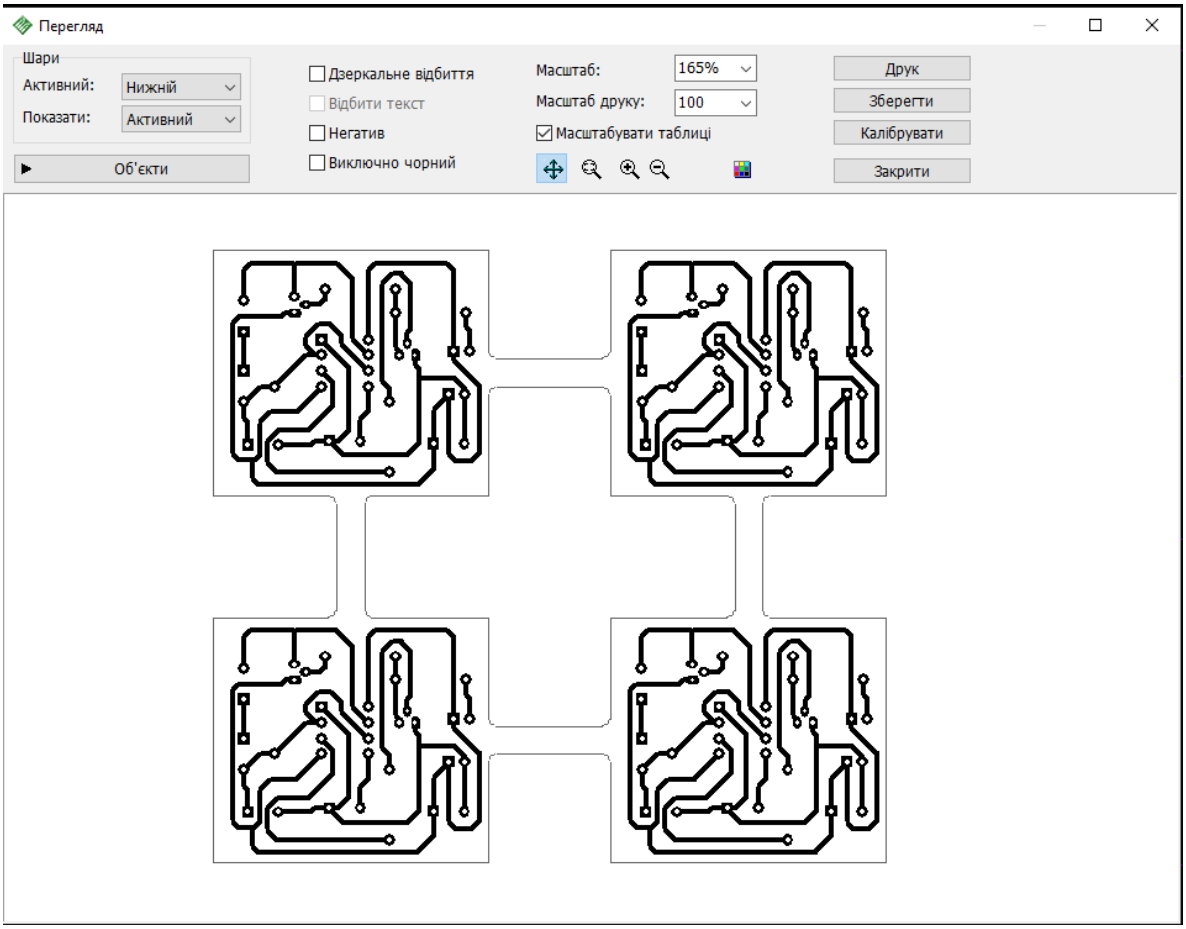


Рисунок 3.4 — Приклад креслення доріжок

Для цього необхідно перейти в попередній перегляд та провести налаштування шарів та масштабу друку, так для «Вузла» масштаб друку 200%, активний верхній шар.

Для «Плата» необхідно провести попередня панелювання плати, для цього в основному меню обрати «Інструменти/Панелювання», вказати 2 стовпчики та рядочки, відстань по X та Y встановити 20 мм, решта налаштувань згідно рис. 3.6. Після цього перейти в попередній перегляд, обрати активним шаром нижній, масштаб друку 100% та розмістити плати по центру. Далі тиснемо кнопку «Друк» та серед принтерів обираємо Adobe PDF.

7. Провести тривимірну візуалізацію розробленої друкованої плати. Для цього необхідно в головному меню програми обрати «Інструменти/3D-перегляд/3D-візуалізація…». Переконатися, що програма знайшла моделі для всіх ЕК крім батарейки (наявність запису в стовбці Файл 3D-моделі) та

натиснути «OK». Переконатися в наявності 3D-моделей для всіх ЕК, при потребі обертаючи плату за допомогою відповідних інструментів «X», «Y» та «Z». Експортуйте створену 3D-модель в формати STEP та VRML, обравши у відповідних вікнах всі пункти.

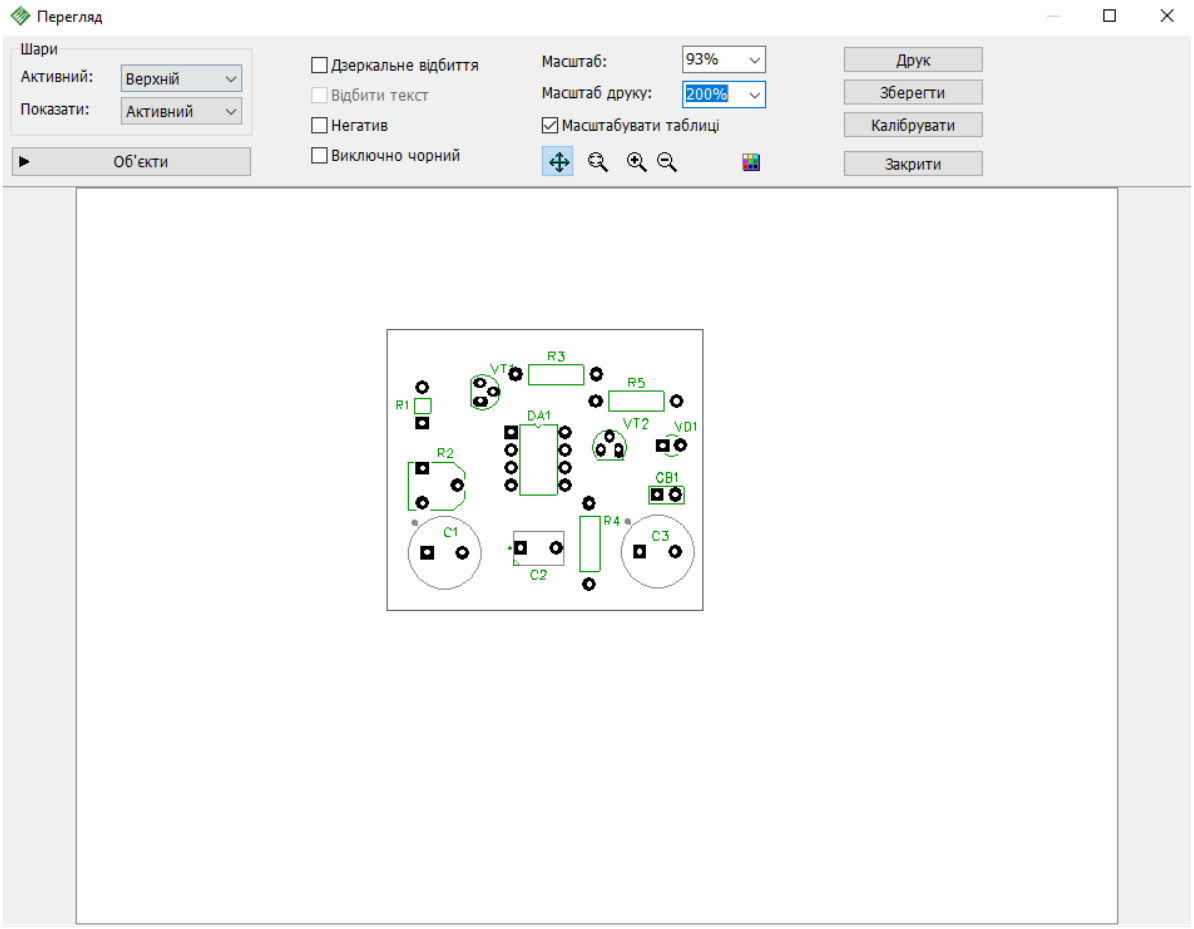


Рисунок 3.5 — Приклад маркування ЕК.

8. Підготувати наступні файли:

- Схема.dhc — Схема в Schematic;
- Схема.pdf — Схема роздрукована в PDF;
- Плата.dip — Плата в PCB Layout;
- Вузол.pdf — Зображення маркування ЕК в PDF;
- Плата.pdf — Креслення доріжок в PDF;
- 3D.STEP — 3D-модель в формати STEP;
- 3D.WRL — 3D-модель в формати VRML.

Архівувати вище перераховані файли в архів з назвою XXXX_П, де XXXX — шифр Вашої групи (наприклад, PI61), П — прізвища студентів, які працювали над платою. В результаті у Вас повинен вийти файл з назвою, наприклад, PI-71_Адаменко.rar

Передати архів викладачу у вказаний спосіб.

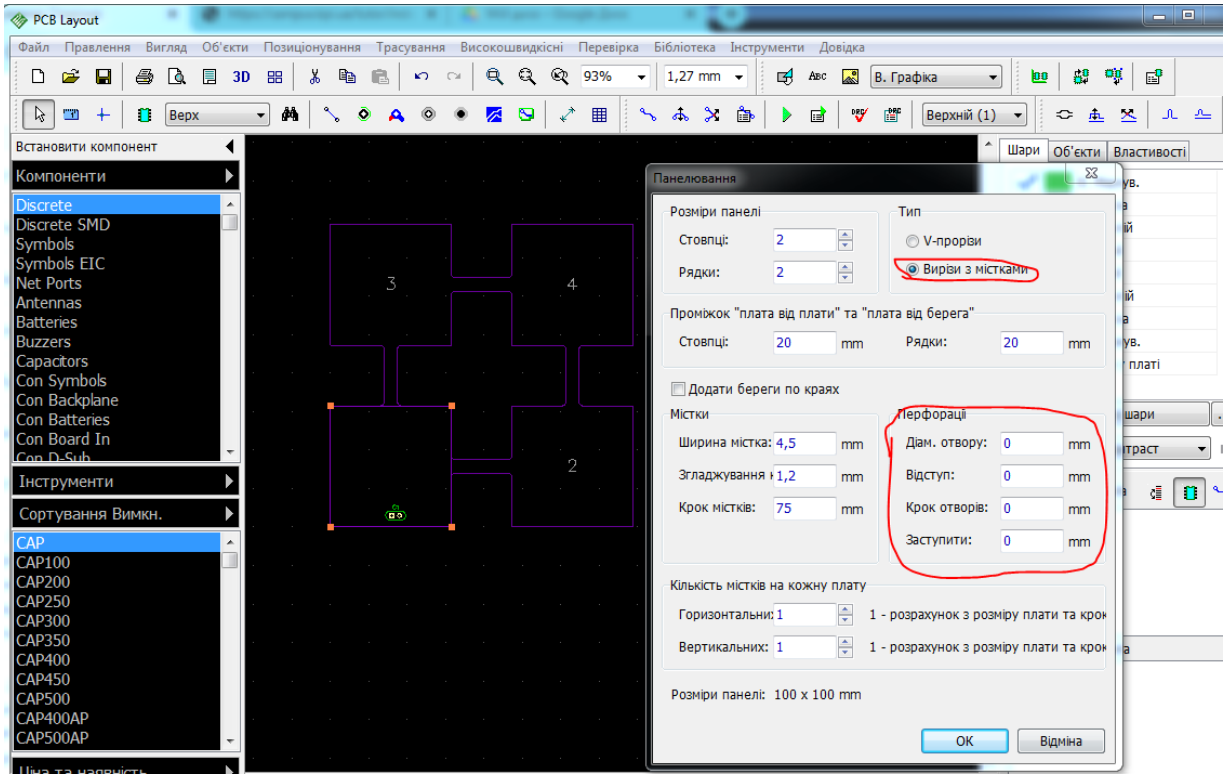


Рисунок 3.6 — Приклад налаштування меню панелювання

Контрольні питання

1. Яким чином задати параметри друкованого рисунку?
2. Як змінити розміри контактних майданчиків та діаметри отворів?
3. Від чого залежить вибір параметрів друкованого рисунку?

ЛАБОРАТОРНА РОБОТА № 4
ВИГОТОВЛЕННЯ ДРУКОВАНОЇ ПЛАТИ

Мета роботи: виготовлення друкованої плати методом хімічного травлення. Отримання уявлення про методи формування провідникового рисунку на друкованій платі.

Теоретичні відомості

Основним завданням в процесі виготовлення друкованої плати є отримання на діелектричній основі провідникового рисунку, який забезпечить електричний контакт між ЕК. Однією з найпростіших технологій виготовлення є метод хімічного травлення, який полягає в тому, що на фольгований діелектрик (зазвичай склотекстоліт з наклеєною мідною фольгою) наноситься захисний шар в місцях, де повинні залишитися провідникові доріжки, а потім за допомогою реактивів відбувається витравлювання міді з незахищених ділянок (рис. 4.1). Після отримання провідникового рисунку проводять висвердлювання отворів в потрібних місцях.

Найпростішим варіантом нанесення захисного шару на фольгований склотекстоліт в домашніх умовах є використання методу термічного пресування, який більш поширений під назвою Лазерно-праскова технологія (Лазерно-утюжная технология, ЛУТ), докладна інформація про цю технологію широко представлена в мережі Інтернет. Для її використання потрібен звичайний лазерний принтер та праска. Крім методу термічного пресування доволі нескладно в домашніх умовах відтворити більш традиційний для виробництва метод з використанням фоторезисту. Проте цей метод потребує наявності самого фоторезисту, прозорої плівки для фотошаблону, лазерного принтеру здатного друкувати на такій плівці, лампи для засвічування фоторезисту та реактивів для його проявлення.

Перенесення провідникового рисунку на склотекстоліт методом термічного пресування можлива завдяки високій адгезійній здатності тонеру лазерного принтера. На спеціальному папері друкується спроектований рисунок друкованої плати (рис. 4.1а). Потім надрукований рисунок прикладається до очищеного склотекстоліту (рис. 4.1б) та проводиться процес прасування. Основна мета якого нагріти плату та притиснути папір з надрукованим рисунком до фольги. Так при повторному нагріванні тонер розм’якшується та під дією тиску прилипає до мідної фольги. Після охолодження тонер знову твердіє і надійно приклеються до поверхні склотекстоліту. Приклеєний таким чином папір можна відмочити у звичайній воді та відділити

від плати. В результаті на платі повинен залишитися чіткий рисунок (рис. 4.1в).

Зверніть увагу! При перенесенні рисунку таким способом ми отримуємо на склотекстоліті його дзеркальну копію!

Після нанесення захисного шару на фольгований діелектрик необхідно провести процес травлення. В домашніх умовах найпростішим є використання двох типів реактивів:

– розчин пероксиду водню, кухонної солі та лимонної кислоти;
– розчин хлориду заліза.

Перший розчин готується за наступним рецептом: в 100 мл пероксиду водню (3%) додається 30г лимонної кислоти та 5 г звичайної солі. Для більш ефективного травлення розчин бажано підігріти.

Для приготування другого розчину необхідно розчинити 50 грам хлориду заліза $FeCl_3$ в 150 мл води. Рекомендовано брати теплу воду, процес травлення буде більш ефективний.

Кожен з цих розчинів має свої переваги та недоліки. Травник на основі пероксиду водню готується з речовин, які дуже просто купити. Крім того, він є більш безпечним. Проте таким чином отримуємо одноразовий розчин, який потрібно змінювати після витравлювання кожної плати (в кращому випадку декількох плат). Для приготування травника на основі хлориду заліза потрібно останній ще десь придбати, що може бути не так і просто. Крім того, розчин є агресивним до багатьох матеріалів, а особливо до металів. При потраплянні на одяг залишає бурі плями, які не піддаються виведенню. Тому працювати з ним потрібно дуже обережно! Переваго ж цього травника є його довший термін експлуатації. Він без проблем витримує витравлювання десятків плат.

Для успішного проведення травлення рекомендується плату класти фольгою до низу на поверхню розчину (правда необхідно переконатися, що на поверхні плати не з’явилися повітряні пухирці), адже в такому положенні процес травлення проходить значно швидше, так як це спрощує відведення осаду, який утворюються в процесі травлення від поверхні міді. Також значно пришвидшує процес травлення перемішування та підігрівання розчинів.

Зверніть увагу! Розчини для травлення готуються в пластиковій або скляній тарі!

Процес травлення потрібно контролювати, так як на час травлення

впливає дуже велика кількість факторів. Типовий час травлення може коливатися в межах від 20 хвилин, до декількох годин. Оптимальним є завершення травлення в момент, коли повністю зникла мідь з незахищених ділянок (рис. 4.1г). Подальше тримання плати в розчині для травлення призведе до значного підтравлювання міді під захисним покриттям в місцях доріжок.

Після завершення травлення плату необхідно промити під проточною водою, висушити та зняти захисне покриття (рис. 4.1д). Розчин для травлення необхідно утилізувати у встановленому порядку.

а б в г д е

Рисунок 4.1 — Ілюстрація етапів виготовлення друкованої плати: а — роздрукований провідниковий рисунок, б — склотекстолітова заготовка, в — перенесений рисунок на плату, г — витравлена плата, д — плата зі знятим тонером, е — залуження провідникових доріжок

Фінальним етапом рекомендовано провести лудіння доріжок, тобто покриття їх тонким шаром припою для захисту від окиснення (рис. 4.1е). Як це зробити розглядається на наступному занятті.

Техніка безпеки

1. Уважно прослухайте додатковий інструктаж по роботі з свердлильним верстатом, паяльником, праскою та реактивами і дотримуйтесь озвучених правил.

2. При роботі з нагрівальними приладами будьте уважні та обережні! Запобігайте контакту нагрітих частин зі шкірою чи одягом.

3. Дотримуйтеся основних правил роботи з електрообладнанням.

4. Під час роботи з реактивами будьте уважні, запобігайте їх потраплянню на шкіру чи одяг.

Порядок виконання роботи

1. Отримати склотекстоліт та роздруковані кресленики у викладача.

2. Уважно ознайомитися з правилами техніки безпеки під час виконання роботи.

3. Очистити склотекстолітову заготовку за допомогою наждачного паперу та звичайної гумки для стирання олівця.

4. Перенести друкований рисунок з паперу на склотекстоліт за допомогою праски. Прибрати папір та переконатися у відповідності та повноті перенесеного рисунку. В разі невдалого процесу перенесення повторити його. Незначні виправлення можна вносити за допомогою перманентного маркера.

5. Провести процес витравлювання, після його завершення промити та висушити друковану плату.

6. Зняти захисний шар та провести висвердлювання отворів в потрібних місцях.

7. Продемонструвати результати виконаної роботи викладачу.

Контрольні питання

1. Коротко опишіть метод хімічного травлення.

2. Якими способами можна перенести рисунок друкованих доріжок на склотекстоліт.?

3. Які травники можна використовувати для витравлювання плати?

ЛАБОРАТОРНА РОБОТА № 5
ПРОВЕДЕННЯ МОНТАЖУ ЕЛЕКТРОННИХ КОМПОНЕНТІВ

Мета роботи: Ознайомитися з принципами монтажу ЕК на друковану плату. Отримати навички роботи з паяльником та проведення монтажу ЕК.

Теоретичні відомості

Монтаж ЕК на друковано плату умовно можна поділити на два етапи: встановлення ЕК на друковану плату, припаювання виводів ЕК до контактних майданчиків друкованої плати. Розглянемо ці два етапи докладніше.

Встановлення ЕК на друковану плату

В даному розділі розглядаються виключно ті варіанти встановлення ЕК, які варто використати при виконанні завдання. Тому нижче наведено інформацію виключно про ЕК, які мають виводи для встановлення їх у отвори друкованої плати.

Так на рис. 5.1 зображено приклад встановлення резисторів (а), звичайних (б) та електролітичних конденсаторів (в). При встановленні напівпровідникових ЕК, таких як транзистори (рис. 5г), світлодіоди (рис. 5д) та фоторезистори варто витримати зазор між ЕК та платою на рівні 5–10 мм, що дозволить уникнути їх перегрівання під час паяння. Для світлодіода та фоторезистора зазор можна залишити і більшим, в даному випадку це не є принциповим.

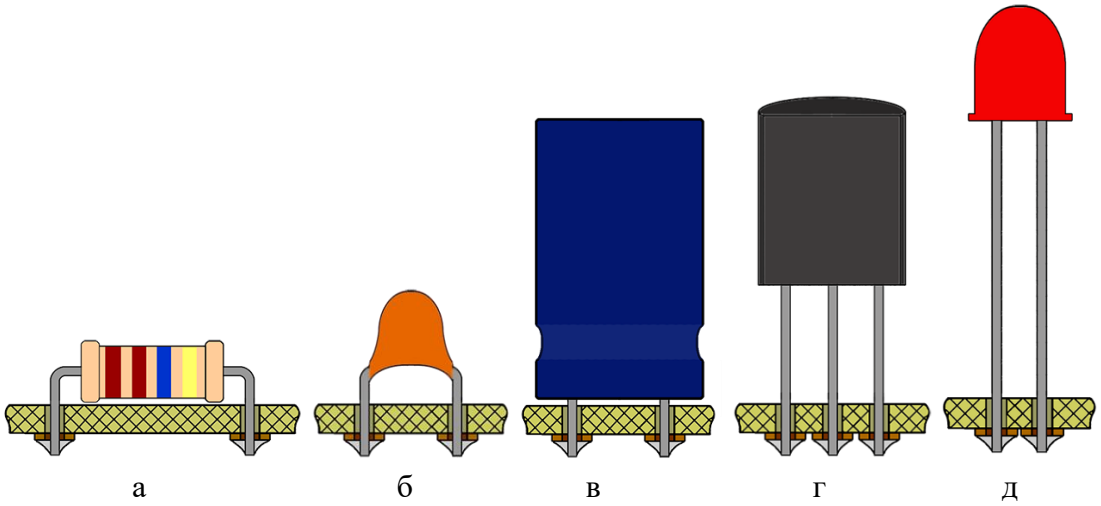


Рисунок 5.1 — Встановлення на плату резисторів (а), керамічного конденсатора (б), електролітичних конденсаторів (в), транзисторів (г) та світлодіода (д)

Висота встановлення на плату змінного резистора та мікросхеми проводиться до їх потовщення на виводах. Власне нижче їх посадити буде проблематично.

Зверніть увагу! При встановленні ЕК на плату потрібно враховувати схему розташування їх виводів, якщо це важливо. А це важливо для наступних ЕК: транзистори, мікросхема, світлодіод, змінний резистор.

Після встановлення ЕК на плату варто провести відкушування зайвих частин виводу за допомогою гострогубців, адже в більшості випадків ЕК випускаються з занадто довгими виводами. Вивід ЕК повинен виходити за межі плати на відстань близько двох міліметрів. Вкорочувати виводи можна як до початку пайки так і після припаювання виводів.

Окремо необхідно підготувати два гнучкі провідники для пайки. Вони будуть виконувати роль «батарейки». Тобто за їх допомогою на плату подаватиметься напруга живлення. Для цього необхідно обережно зняти захисну пластикову ізоляцію з обох кінців кожного провідника. Довжина знятої ізоляції в межах 3–5 мм. Потім потрібно скрутити розпушені жили проводу в одну (варто врахувати напрям їх скрутки по замовчуванню) та залудити їх. Зняти ізоляцію можна за допомогою гострогубців.

Процес пайки

Пайка — процес формування з'єднання різних деталей шляхом введення в зазор між ними розплавленого металу (припою), який має більш низьку температуру плавлення ніж матеріали, які паяються. Після охолодження припою формується паяне з'єднання. В процесі пайки приймає участь два компоненти: припій та флюс.

Припій — суміш двох або більше металів, за допомогою яких власне і відбувається з'єднання. Найпопулярнішим припоєм для монтажу ЕК є ПОС-61 — припій олов'яно-свинцевий з вмістом олова 61%. Іноземним аналогом частіше всього виступає припій Sn63/Pb37, який містить 63% олова (*stannum*) та 37 відсотків свинцю (*plumbum*). Він має температуру плавлення 183 ^{0}C, що, звичайно ж, значно нижче температури плавлення виводів ЕК або мідних доріжок друкованої плати.

Флюс — додаткова речовина, яка вноситься в зону паяння для усунення окислів металів та бруду і в процесі випаровування створює захисну атмосферу, яка запобігає окисненню металів в процесі пайки. Найпоширенішим флюсом для паяння ЕК є каніфоль або її розчин в спиртах. Каніфоль — ре-

човина, яка входить до складу смоли хвойних дерев та містить значну кількість органічних кислот. Температура плавлення каніфолі 100–120 ^{0}C

Сучасна промисловість випускає припої у вигляді тонкої проволоки, яка містить в середині флюс.

Зверніть увагу! Створити гарне паяне з'єднання можливо виключно якщо в зоні пайки є достатня кількість флюсу та припою.

Для проведення ручного монтажу ЕК на друковану плату використовують паяльник (або паяльну станцію), завдання якого нагріти зону пайки до температури вищої за температуру плавлення припою. Якщо в цей момент в зоні пайки буде знаходитися припій та флюс, то відбудеться процес розтікання припою по паяним деталям. Типовий час пайки 2–5 секунд. Типова температура жала паяльника 220–240 ^{0}C. В домашніх умовах проконтролювати температуру жала можна за рахунок каніфолі. Так при зануренні жала паяльника в тверду каніфоль вона повинна швидко плавитися та починати випаровуватися з легким «димком», при цьому при витягуванні жала з каніфолі ще декілька секунд повинен продовжуватися процес випаровування з жала (за цей час і варто проводити паяння). Якщо ж при зануренні жала чути дуже гучний звук кипіння каніфолі та при витягуванні жала воно практично одразу перестає «диміти», то жало паяльника явно перегріте і варто знизити температуру.

Зверніть увагу! Не варто неперервно гріти місце пайки більше 5 секунд, адже це може призвести до перегріву та псуванню ЕК та відшаруванню мідної фольги від діелектрика. Проте менш ніж за 2 секунди надійне паяне з'єднання отримати не можливо, так як необхідно, щоб під час пайки до температури вищої за температуру плавлення припою нагрівся не тільки припій, а ще й деталі, які підлягають спаюванню.

За зміною температури процес пайки можна розділити на декілька стадій:

1. Нагрів зони пайки до температури плавлення флюсу, в результаті чого кислоти, які входять до його складу починають взаємодіяти з оксидами металів, руйнуючи оксидну плівку

2. Температура підвищується далі, флюс починає випаровуватися, витісняючи кисень з зони пайки

3. При досягненні температури плавлення припій за рахунок ефектів змочування та рідиноплинності починає розтікатися по поверхнях з'єднува-

них матеріалів, заповнюючи проміжок між ними. Відбуваються процеси дифузії.

4. Після цього температуру в зоні пайки потрібно понизити, в результаті чого припій затвердіває і утворюється з'єднання матеріалів.

Зрозуміло, що в процесі застигання припою потрібно уникати руху паяних деталей відносно одна одної.

Зверніть увагу! До складу припою входить свинець, який відноситься до важких металів з токсичною дією на організм. Тобто він здатен накопичуватися в організмі та отруювати його. Звичайно ж в процесі пайки випари свинцю доволі мізерні, проте техніка безпеки вимагає проводити паяння в приміщеннях обладнаних місцевою та загальною витяжками. Проте реалізувати це в домашніх умовах складно, та й не доцільно, якщо паяння не проводиться постійно. Проте варто забезпечити гарне провітрювання приміщення під час та після пайки.

Ознакою якісного паяного з'єднання є блискуча поверхня припою, яка рівномірно вкриває вивід ЕК та контактний майданчик, при чому форма припою не повинна бути опуклою.

Лудіння

В радіоелектроніці під лудінням розуміють покривання металевих поверхонь тонким шаром припою (або іншим сплавом) з метою їх захисту від окиснення та покращення подальшого процесу паяння.

Одним з етапів виготовлення друкованої плати в домашніх умовах є лудіння мідних провідників. Найкращим способом лудіння мідної фольги в домашніх умовах є використання рідкого флюсу, проте лудіння з використанням твердої каніфолі провести не так і важко, проте це потребує певних навичок.

Зверніть увагу! Перед початком лудіння мідну фольгу потрібно гарно очистити від бруду та оксидів. Це можна зробити звичайної канцелярської гумки, просто потерши нею мідну поверхню.

На мідну поверхню наноситься невелика кількість флюсу. Потім розігрітим паяльником з нанесеним на жало припоєм потрібно просто доторкнутися до мідної фольги так, щоб вона прогрівалася за рахунок жала паяльника. Як тільки температура міді підніметься до температури плавлення припою, останній почне розтікатися по поверхні міді. Паяльник плавним рухом потрібно переміщати вздовж мідних доріжок.

Зверніть увагу! Основним тут є нагрів міді до температури вищої за температуру плавлення припою, тому не варто неперервно терти поверхню жалом паяльника. Також для лудіння потрібна зовсім невелика кількість припою, адже він повинен вкривати мідь дуже тонким шаром!

Процес лудіння за допомогою твердої каніфолі практично такий же, але каніфоль наноситься безпосередньо на жало паяльника шляхом занурення останнього в каніфоль, а так процес аналогічний до вище написаного.

Техніка безпеки

1. Уважно прослухайте додатковий інструктаж по роботі з паяльником та дотримуйтесь його.

2. При роботі з паяльником будьте уважні та обережні! Запобігайте контакту нагрітих частин зі шкірою чи одягом. Не варто перевіряти температуру паяльника за допомогою рук. Це може призвести до опіку!

3. Очищення жала паяльника проводити виключно за допомогою спеціальної губки.

4. Дотримуйтеся основних правил роботи з електрообладнанням.

5. Зверніть увагу на шкідливість випарів свинцю та каніфолі в процесі паяння.

6. При відкусуванні зайвих частин виводу ЕК за допомогою гострогубців слід спрямовувати потенційний політ відкушеного виводу в стіл чи підлогу!

Порядок виконання роботи

1. Отримати у викладача плату, кресленики друковано плати та вузла і набір ЕК згідно таблиці 5.1.

Таблиця 5.1 — Компоненти, необхідні для виконання роботи

ЕРЕ	**Номінал (тип)**	**Кількість, шт**
Світлодіод	червоний	1
Резистори	680 Ом	2
	220 Ом	1
	3,3 кОм	2
	100 кОм	1
	фоторезистор	1
	50 кОм, підстроювальний	1

Продовження Таблиці 5.1

ЕРЕ	Номінал (тип)	Кількість, шт
Конденсатори	100 нФ	1
	33 мкФ 50В	1
	100 мкФ 16В	1
Мікросхема	NE 555	1
Транзистори	BC547	1
	BC557	1
Гнучкі провідники	–	2

2. Уважно ознайомитися з правилами техніки безпеки та теоретичними відомостями.

3. Розсортувати ЕК згідно отриманої документації та визначити їх положення на друкованій платі.

4. Провести встановлення ЕК у відповідні місця та їх паяння. Рекомендовано робити це по черзі, тобто: встановили перший ЕК, припаяли його і так далі. Особливо уважно провести встановлення мікросхеми, транзисторів та світлодіода. Від правильності їх встановлення залежить працездатність схеми.

5. Після завершення монтажу ще раз переконатися в тому, що всі компоненти знаходяться на своїх місцях, якості паяних з’єднань та відсутність перемичок на платі в не потрібних місцях.

6. У присутності викладача підключити пристрій до джерела живлення (в правильній полярності) та переконатися в його працездатності. Перевірити вплив освітленості фоторезистора та положення повзунка змінного резистора на частоту блимання світлодіода.

7. Якщо пристрій не запрацював — провести його діагностику: перевірити правильність встановлення напівпровідникових ЕК, відсутності випадкових перемичок в непотрібних місцях тощо. Виправити знайдені помилки та переконатися в працездатності пристрою.

8. Прибрати робоче місце від залишків виводів ЕК, скласти інструменти, припій та каніфоль. Вимкнути паяльник.

9. Спаяна плата залишається студенту. В домашніх умовах пристрій можна підключати до будь-якого джерела живлення з напругою від 3 до 16 Вольт: батарейки, акумулятори, зарядні пристрої до телефонів та деяких

ноутбуків тощо. Головне зберігати полярність підключення.

Контрольні питання

1. В чому полягає процес паяння ЕК?
2. Навіщо в процесі паяння використовувати флюс?
3. В чому полягає процес лудіння?
4. Які основні правила встановлення ЕК на плату?

ЛАБОРАТОРНА РОБОТА № 6
ПРОГРАМНО-АПАРАТНА ПЛАТФОРМА ARDUINO

Мета роботи: ознайомитися з апаратною платформою *Arduino* та програмним середовищем *Arduino IDE*. Створити просту програму для керування світлодіодами.

Теоретичні відомості

Arduino — це програмно-апаратна платформа для конструювання різноманітних радіоелектронних систем, вона складається безпосередньо з плати *Arduino* на основі мікроконтролера сімейства *Atmel AVR* з невеликою кількістю додаткових компонентів та програмного середовища *Arduino IDE,* яке допомагає швидко та зручно створювати різні програми. Звичайно не можна не згадати додаткових плат розширення: давачів температури, вологості, загазованості тощо; виконавчих пристроїв, таких як світлодіоди, реле, двигуни тощо; систем керування, індикації та великої кількості інших додаткових плат, які здатні задовольнити запити конструктора-початківця.

Мікроконтролер (англ. *microcontroller*) — це по суті однокристальний комп'ютер з додатковими спеціальними блоками (портами введення/виведення, лічильниками, компараторами, аналого-цифрових перетворювачів тощо), виконаний у вигляді мікросхеми, який здатен виконувати прості завдання, записані за допомогою мови програмування. Основна сфера застосування — керування електронними пристроями. Використання однієї мікросхеми значно знижує розміри, енергоспоживання і вартість пристроїв, побудованих на базі мікроконтролерів.

Апаратна частина

На рис. 6.1 зображено апаратну платформу *Arduino UNO*, яка буде використовуватися під час лабораторних робіт. Вона реалізована на основі мікроконтролера *ATmega 328* [6]. Плата *Arduino* (рис. 6.1) містить роз'єм *USB type B* для підключення до комп'ютера та роз'єм живлення, який використовується у випадку, якщо плата експлуатується окремо від комп'ютера (Напруга, яку можна подавати на цей роз'єм повинна бути в межах від 6 до 12 В). Також плата містить драйвер *USB* порту (Мікросхема *CH*340) два лінійних стабілізатора напруги на 5 та 3,3 В та необхідні ЕК для правильного функціонування драйверу та самого мікроконтролера.

На платі розміщено кнопку, яка використовується для перезавантаження мікроконтролера.

Також на платі розміщено 4 світлодіоди («4» на рис. 6.1), які допомагають проконтролювати роботу платформи: світлодіод з позначкою *«ON»* засвічується, коли на плату подається живлення, світлодіоди *«RX»* та *«TX»* сигналізують передавання даних через послідовний інтерфейс, світлодіод *«L»* підключений через обмежувальний резистор до 13 виводу плати та сигналізує про наявність високого логічного рівня на виводі.

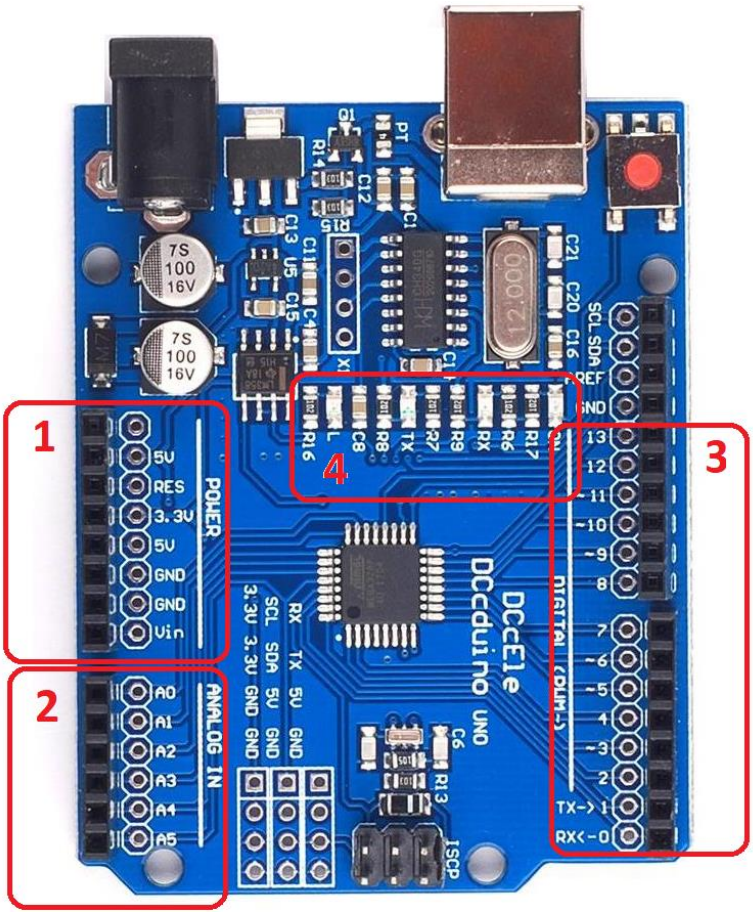


Рисунок 6.1 — Плата Arduino UNO.

Перш за все нас цікавлять два ряди виводів, які дають змогу мікроконтролеру спілкуватися з навколишнім середовищем.

В першому наближенні ці виводи можна поділити на три групи. До першої відноситься група виводів під назвою *«Power»* (Живлення), на рисунку ця група виводів позначена «1». Вона містить (зверху вниз) виводи: 5V — тобто напруга живлення +5 В, *RES* — вивід для перезавантаження плати, 3.3V — напруга живлення +3,3 В, знову вивід 5*V*, два виводи *GND* — спільний провід, земля або ж мінус джерела живлення та вивід *VIN*, який використовується для підключення зовнішнього живлення при відсутності живлення через *USB*.

Друга група виводів («2» на рис. 6.1) — аналогові входи (*Analog In*),

вони дозволяють вимірювати напругу в межах від 0 до 5В (або менше, якщо це задано програмно) та в більшості випадків потрібні для підключення аналогових давачів. Давач (або ж сенсор, від англ. *Sensor*) — це такий пристрій, який перетворює значення одних фізичних величин в інші. Наприклад, терморезистор, можна використовувати у якості давача температури, адже опір такого резистора змінюється в залежності від температури навколишнього середовища.

Третя група найбільша («3» на рис. 6.1), вона містить так звані цифрові порти введення-виведення (*I/O — Input/Output*), на платі вони підписані, як *«Digital PWM(~)»*. Тобто виводи, які здатні як виводити інформацію так і зчитувати її ззовні. Правда є одна особливість — ці виводи працюють тільки з двома значеннями: 0 (або ж логічним нулем, низьким рівнем, *LOW*, мінусом) та 1 (або ж логічною одиницею, високим рівнем, *HIGH*, напругою живлення +5В). Максимальний струм, який може віддати такий вивід — 40 мА (рекомендований струм — 20 мА), проте максимальний струм, який може віддати мікроконтролер не повинен перевищувати 200 мА. Виводи, які мають додаткову позначку «~» також можуть працювати, як виходи для широтно-імпульсної модуляції (ШІМ — англ. *pulse-width modulation, PWM*), більш докладно це питання буде розглянуто пізніше.

Звичайно все трохи складніше, ніж написано вище, адже більшість виводів можуть виконувати декілька функцій, але це тема не цього заняття. З повним переліком виводів та режимів їх роботи можна ознайомитися в Додатку Б або ж безпосередньо на кольоровій наліпці на макетних платах.

В загальному випадку завдання мікроконтролера покроково виконати команди, які записані до нього під час програмування. При цьому він здатен виконувати різні математичні та логічні операції, змінювати значення напруги на своїх виходах та зчитувати її з входів.

Проте більш докладно принцип роботи мікроконтролерів та окремих його частин буде розглянуто на старших курсах.

Програмна частина

Програмування мікроконтролера, який встановлений на платі *Arduino* відбувається за допомого спеціально розробленого для цього програмного середовища розробки *Arduino IDE*. Власне абревіатура *IDE* розшифровується як *Integrated Development Environment*, тобто інтегроване середовище розробки. Вікно середовища програмування *Arduino IDE* (рис. 6.2) нічим

принципово не відрізняється від типових вікон подібних програм.

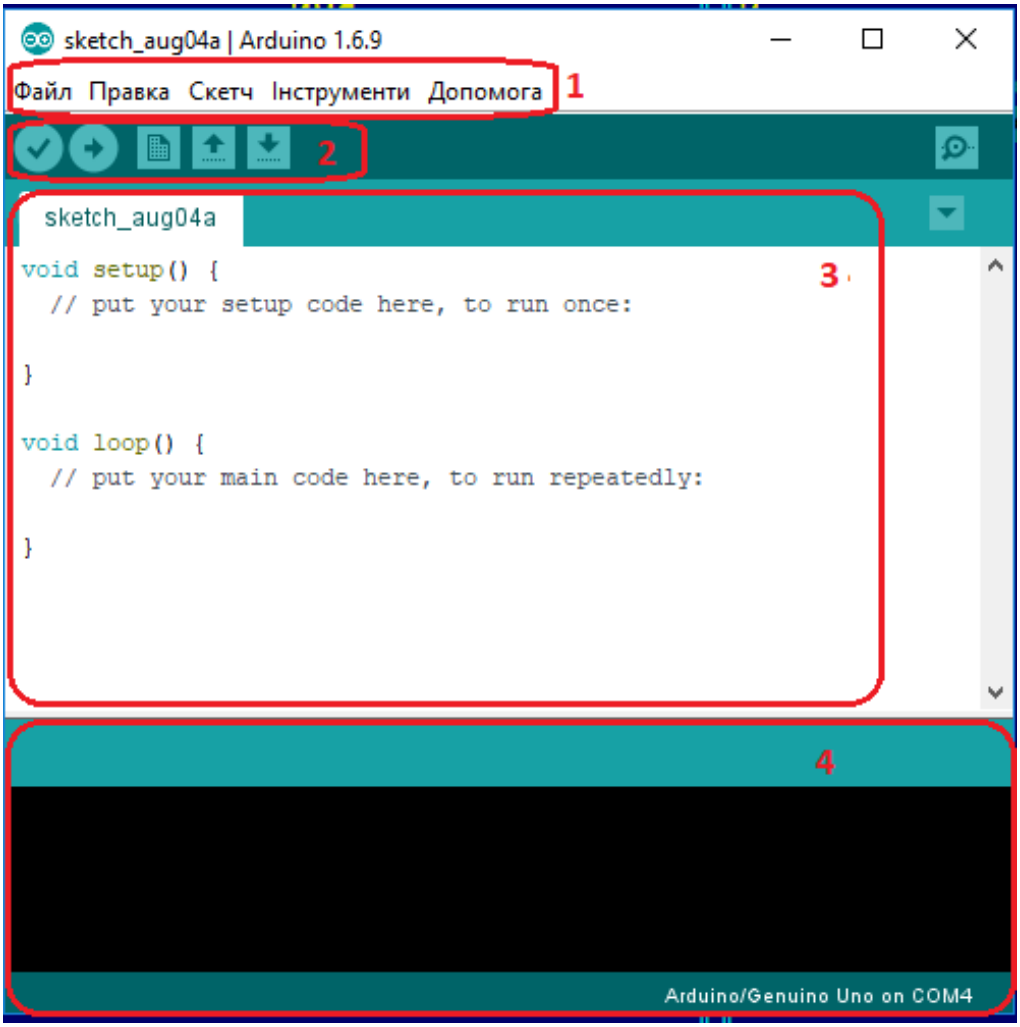


Рисунок 6.2 — Вікно програмного середовища *Arduino IDE*

Програма містить головне меню «1» (обведено червоним), меню швидкого доступу «2», вікно для написання коду програми «3» та вікно статусу виконання «4».

За допомогою меню швидкого доступу можна вивантажити програму до апаратної платформи *Arduino*. Перед вивантаженням варто переконатися у вірності налаштування з'єднання. Для цього зайти в меню «Інструменти» переконатися що обрано, а при потребі обрати: плата: *«Arduino/Cenuino UNO»*; програматор *«AVRISP mkII»*; порт — *COM*2–6 в залежності від того, який з них буде у списку (*COM*1 — зазвичай є по замовчуванню, але це не плата Arduino).

Традиційно програми для платформи *Arduino* називають Скетчами від англійського слова *Sketch* (ескіз).

Зверніть увагу! Мова програмування *Arduino* базується на *C/C++* та

спеціальній бібліотеці *avr-libc*, яка розроблена для програмування мікроконтролерів. Мову *Arduino* можна розділити на чотири розділи: оператори, дані, функції та бібліотеки [7]. Повний опис мови програмування можна переглянути на офіційному сайті *Arduino*: https://www.arduino.cc/reference/en/. Частково опис мови програмування наведено в Додатку В навчального посібника.

Для створення нового скетча в *Arduino IDE* необхідно натиснути відповідну кнопку меню швидкого доступу або ж обрати в головному меню «Файл/Створити».

Новостворений скетч вже містить дві функції: *void setup ()* та *void loop ()*. Перша виконується один раз, друга — повторюється весь час, який плата *Arduino* підключена до живлення. Саме у цьому вікні потрібно писати ті команди, які Ви хочете, щоб виконувала плата *Arduino* постійно.

Зверніть увагу! Функції *void setup ()* та *void loop ()* повинні буди в скетчі обов'язково, навіть якщо вони пусті.

В даній лабораторній роботі використовуються наступні нові функції та синтаксис:

«;» — крапка з комою ставиться в кінці оператора/функції.

«{}» — фігурні дужки виокремлюють тіло певно функції, застосовуються в функціях, циклах та операторах умови. **Зверніть увагу!** Кожній відкритій дужці повинна бути відповідна закриваюча дужка!

«//» — коментар. Текст, записаний після подвійного слешу ігнорується компілятором.

«/*» та «*/» — відкриваючий та закриваючий символ багаторядкового коментаря.

#define *constantName value*

Це директива, яка створює константу з ім'ям заданим в полі ***constantName*** та присвоює їй значення ***value.*** ЇЇ доречно використовувати в тих випадках, коли потрібно вказати значення, які можуть змінюватися в процесі налагодження Вашого пристрою. Наприклад, номера виводів контролера, кількість світлодіодів тощо. **Зверніть увагу!** Значення директиви не може змінитися в процесі виконання програми! Адже в процесі компіляції всі ***constantName*** просто замінюються на ***value***.

pinMode(*pin, mode*)

Функція конфігурує вивід з номером, який записаний на місці *pin* мікроконтролера на виконання певної функції. Є три варіанти значення для *mode*:

– *INPUT* (або ставлять 0) — зчитування інформації з виводу, власне даний режим встановлюється по замовчуванню, тому викликати дану функцію з цим значенням в більшості випадків немає потреби;

– *OUTPUT* (або «1») — виведення інформації;

– *INPUT_PULLUP* — увімкнення внутрішнього підтягувального резистора та зчитування інформації з виводу. При такій конфігурації з виводу по замовчуванню буде зчитуватися значення логічної одиниці, так як підтягувальний резистор підключається до +5 В.

delay(*ms*)

Функція зупиняє виконання програми на час ***ms*** (значення в мілісекундах).

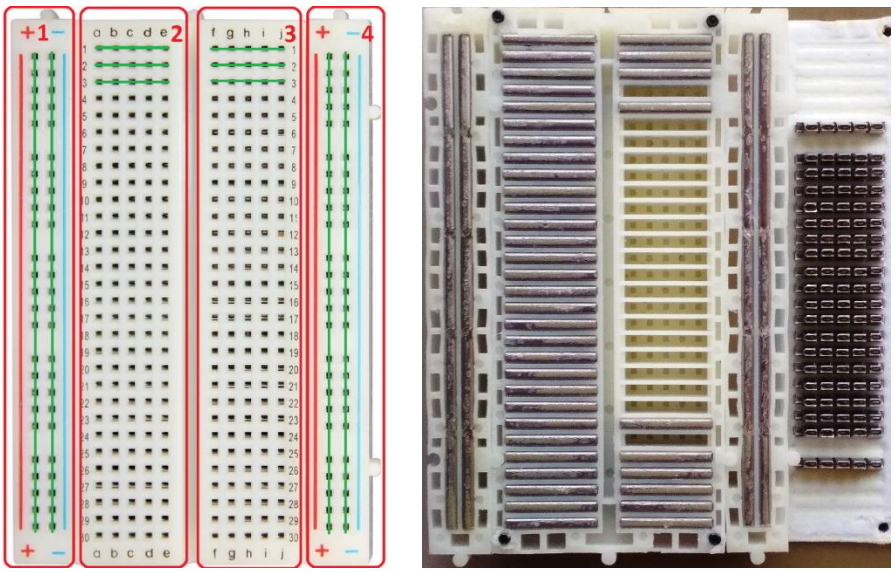
digitalWrite(*pin, value*)

Функція виводить цифрове значення ***value*** на вивід з номером ***pin***. Може виводити два значення: логічний нуль «0» або LOW та логічну одиницю «1» або HIGH (Наприклад, записи digitalWrite*(led_pin_2, LOW)* та digitalWrite*(led_pin_2, 0)* ідентичні та виконують одне й теж, а саме: виведення на вивід *led_pin_2* логічного нуля).

Зверніть увагу! При виконанні функції digitalWrite(*2, HIGH*) на другому виводі мікроконтролера буде встановлено логічну одиницю, що відповідає подачі на цей вивід напруги живлення +5 В. Якщо ж виконана функція digitalWrite(*2, LOW*), то на виводі буде напруга 0 В, що еквівалентно підключенню виводу до GND.

***Макетна плата та з'єднувачі**

Для створення простих електричних схем без використання паяних з'єднань доцільно використовувати макетну плату зі спеціальними гніздами, яка більш відома як макетна плата типу *Breadboard* (Рис. 6.3).



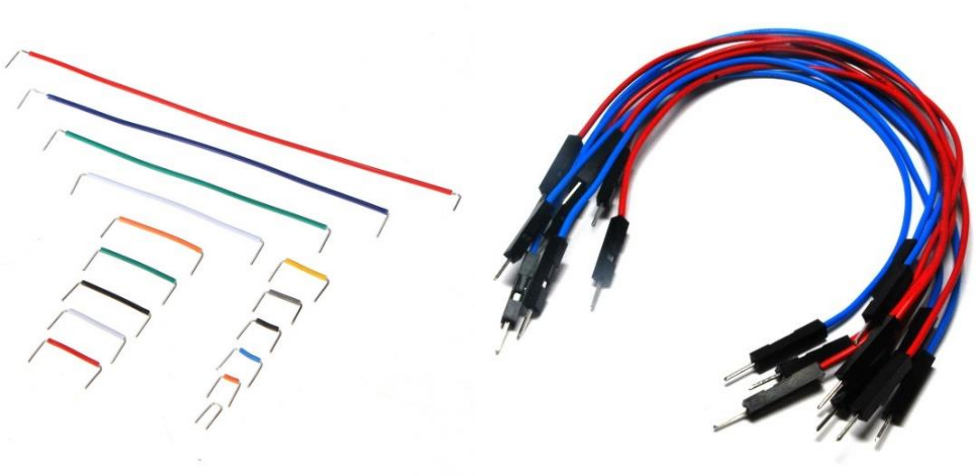
а б

Рисунок 6.3 — Макетна плата (а) та її будова (б)

Як видно з рисунку з’єднання між контактами відбувається за допомогою спеціальних металевих зажимів (рис. 6.3б).

Умовно плату можна поділити на 4 частини (обведені червоним): 1 та 4 шини живлення, слугують в основному для підключення напруги живлення (+ та −), гнізда шин з’єднані між собою вздовж зелених ліній. Дві ж частини 2 та 3 — набір гнізд у вигляді матриці, яка містить 5 стовпчиків (a, b,…, e) та 30 рядків. Причому в цих матрицях елементи одного ряду з’єднані між собою (див. зелені лінії на рис. 6.3а).

Зверніть увагу! Вивід ЕК або перемички в комірку макетної плати повинен входити без значних зусиль.



а б

Рисунок 6.4 — Жорсткі (а) та гнучкі (б) перемички

На макетній платі розміщуються електронні компоненти, так щоб забезпечити потрібні електричні зв’язки за допомогою внутрішніх з’єднань макетної плати та за допомогою додаткових перемичок, які можуть бути як в жорсткому виконанні (рис. 6.4а), так і у гнучкому (рис. 6.4б).

Жорсткі перемички рекомендовано використовувати для забезпечення з’єднань на порівняно коротких відстанях. Гнучкі використовуються в решті випадків, включаючи з’єднання з зовнішніми пристроями. Гнучкі провідники на кінцях можуть мати роз’єми двох типів: «гніздо» (*Female*) (рис. 6.5а) або «штекер» (*Male*) (рис. 6.5б).

Зустрічаються три типи гнучких провідників в залежності від роз’ємів на кінцях: *male-female*, *male-male* та *female-female*. Кожен з них знаходить своє застосування у різних випадках з’єднання макетної плати, зовнішніх модулів та плати *Arduino*.

а б

Рисунок 6.5 — Роз’єми гнучких провідників типу гніздо (а) та штекер (б)

Техніка безпеки

1. Перед підключенням плати *Arduino* до комп’ютера ще раз перевірте правильність схеми та покажіть її викладачу.

2. Будь-які зміни в схемі проводьте виключно при вимкненому живленні (від’єднаному *USB*).

3. Уникайте потрапляння зайвих предметів на монтажну плату та безпосередньо на плату *Arduino*.

4. Забороняється підключати будь-які виконавчі та індикаторні пристрої (світлодіоди, зумери, семисегментні індикатори тощо) безпосередньо до цифрових виводів *Arduino*. Наявність струмообмежувального резистора є обов'язковою умовою!

Порядок виконання роботи

1. Отримати у викладача макетну плату.

2. Підключити *Arduino* за допомогою кабелю до комп'ютера. Переконатися, що засвітився світлодіод живлення.

3. Запустити *Arduino IDE* та перевірити налаштування програми згідно теоретичних відомостей.

4. Скопіювати нижченаведений код до основного вікна програми:

```
/* константа з іменем LED_PIN та значенням 13, в даному випадку 13 — номер виводу Arduino до якого підключено вбудований світлодіод
*/
# define LED_PIN 13
// Ця функція виконується один раз
void setup() {
  // ініціалізація цифрового піну LED_PIN, як виходу
  pinMode(LED_PIN, OUTPUT);
}

// ця функція постійно повторюється
void loop() {
/* вмикаємо світлодіод, подаючи на вивід LED_PIN високий логічний рівень, що відповідає напрузі +5 В.
*/
  digitalWrite(LED_PIN, HIGH);
  delay(500);  // чекаємо протягом 500 мілісекунд
/* вимикаємо світлодіод, подаючи на вивід LED_PIN низький логічний рівень, що відповідає напрузі 0 В
*/
  digitalWrite(LED_PIN, LOW);
  delay(500);   // знову чекаємо протягом 500 мілісекунд
}
```

5. Вивантажити програму до *Arduino*, натиснувши кнопку «Вивантажити» на панелі швидкого доступу або ж обравши в основному меню «Скетч/Вивантажити».

6. Переконатися в успішному завершенні вивантаження в вікні статусу виконання. Та переконатися, що встановлений на платі світлодіод «*L*» блимає з частотою 1 раз за секунду.

Якщо в процесі вивантаження на плату виникла помилка:

«avrdude: ser_open(): can't open device "\\.\COM3": The system cannot find the file specified.

Проблема вивантаження в плату. Зверніться до http://www.arduino.cc/en/Guide/Troubleshooting#upload для пошуку рішення.»

То це означає, що не правильно обрано *COM*-порт до якого підключено *Arduino* (ну або *Arduino* просто не підключено до комп'ютера). Потрібно перевірити налаштування згідно теоретичних відомостей.

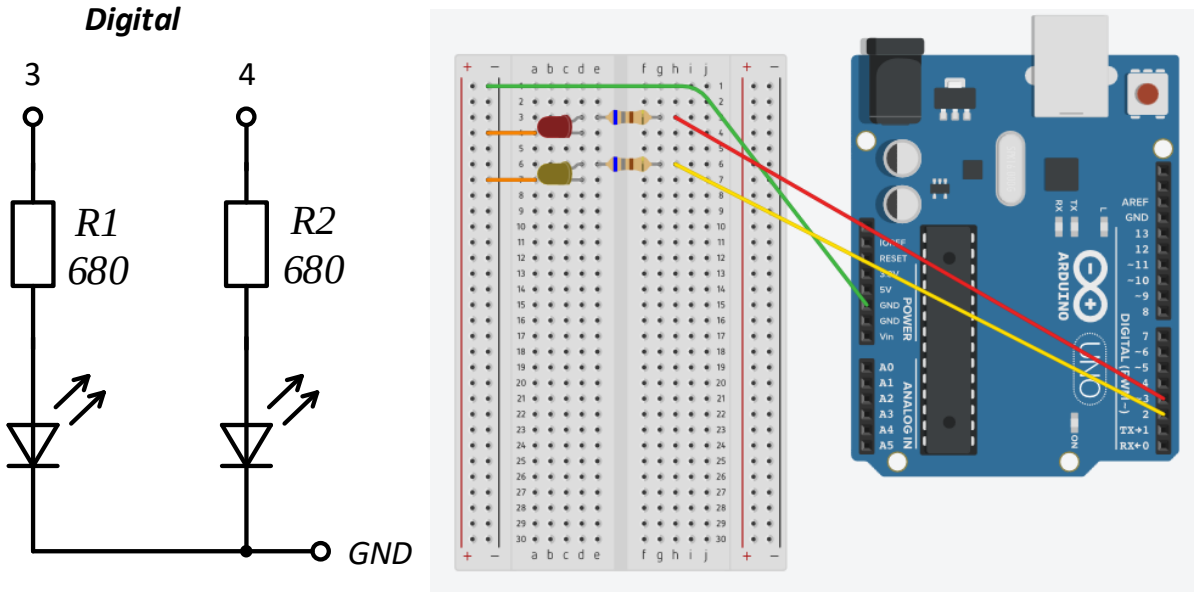
7. Змінити час паузи в першому та другому випадку та заново вивантажити скетч до *Arduino*. При внесенні змін в програму можна не від'єднувати Arduino від комп'ютера.

*8. Отримати у викладача набір ЕК згідно табл. 6.1.

Таблиця 6.1 — Необхідні ЕК для виконання роботи

ЕРЕ	**Номінал (тип)**	**Кількість, шт**
Світлодіоди	різнокольорові	2
Резистори	680 Ом	2
Гнучкі провідники	*male-male*	8
Жорсткі перемички	–	набір

*9. Зібрати на макетній платі схему згідно з рис. 6.6



а б

Рисунок 6.6 — Схема підключення (а) та приклад підключення (б) світлодіодів до Arduino

*10. Переробити вищенаведену програму так, щоб підключені світлодіоди засвічувалися та згасали по черзі. Продемонструвати програму викладачу.

11. Записати до *Arduino* порожній скетч (для цього виконати «Файл/Створити», який містить виключно пусті функції *void setup()* та *void loop()*, вивантажити до *Arduino*), переконатися, що світлодіоди перестали блимати та розібрати зібрану схему. Повернути макет та ЕК викладачу.

Контрольні питання

1. Яке призначення виводів апаратної платформи *Arduino*?
2. Що розміщено на платі *Arduino*?
3. Які основні правила безпеки при роботі з платформою *Arduino*?
4. Як працює Ваша програма?
5. Яка напруга встановлюється на виводі Arduino, якщо він працює як вихід і туди записано рівень логічної одиниці?

ЛАБОРАТОРНА РОБОТА № 7
ВВЕДЕННЯ ТА ВИВЕДЕННЯ ЦИФРОВИХ ДАНИХ. ЧАСТИНА 1

Мета роботи: навчитися працювати з цифровими *I/O* – виводами *Arduino*. Практичне застосування операторів умовного переходу. Написання програми керування режимами роботи світлодіодів.

Теоретичні відомості

Цифровими даними вважаються дані, які кодуються за допомогою двійкової системи числення. Як Вам відомо в двійковій системі числення всі дані записуються за допомогою двох логічних значень: 0 (нуль) або *false* (хиба) та 1 (одиниця) або *true* (істина). Двійкова система числення нерозривно пов'язана з поняттям біту (*bit*) інформації — тобто мінімальна одиниця кількості інформації, яка може приймати значення логічного нуля або одиниці.

За введення та виведення інформації в *Arduino* (як і в інших мікроконтролерних системах) відповідають порти введення/виведення або ж *I/O*–порти від англійських слів *input/output*. Схемотехнічна реалізація даних портів та особливості їх роботи більш докладно вивчається в курсі «Програмування мікроконтролерів та мікропроцесорів».

Виведення цифрових даних

Під виведенням цифрових даних розуміємо встановлення на виводі мікроконтролера напруги, яка відповідає біту інформації, який ми виводимо на нього.

Для виведення цифрових даних використовується вже знайома функція **digitalWrite(*pin, value*)**, при виконанні якої на виводі за номером ***pin*** мікроконтролера встановлюється напруга +5 В, якщо ***value*** відповідає логічній одиниці, та 0 В, якщо логічному нулю (див. заняття №6).

Виведення цифрових даних використовується для індикації та керування зовнішніми пристроями. Наприклад, можна виводити інформацію на семисегментний індикатор чи звичайний світлодіод. Для керування більш потужними навантаженнями використовуються спеціальні підсилювальні схеми на транзисторах чи мікросхемах (так звані драйвери). Також виведення цифрових даних використовується для передавання даних до інших мікроконтролерів чи цифрових мікросхем.

Обмежувальний резистор

Як вже було зазначено вище згідно з технічними характеристиками плати *Arduino* рекомендований струм, який може протікати через ланку виводу I/O складає 20 мА, тому для підключення світлодіода до *Arduino* необхідно використовувати обмежувальний резистор, причому він буде одночасно захищати як мікроконтролер так і світлодіод від перенавантаження. Це зумовлено особливостями вольт-амперної характеристики напівпровідників, до яких і належить світлодіод, більш докладно ця тема буде розглянута на старших курсах в дисципліні «Схемотехніка. Електронні компоненти». В залежності від типу світлодіода та необхідної яскравості його світіння значення обмежувального резистора може коливатися в межах від сотень Ом до декількох кОм (Для практичного застосування часто достатньо використовувати резистор номіналом від 500 Ом до 1 кОм). Для розрахунку обмежувального резистора використовуються закони Ома та Кіргофа, які більш докладно вивчаються в курсі «Основи теорії кіл». Під час виконання роботи можна використовувати наступну формулу:

$$R = \frac{U_{ж} - U_{LED}}{I_{LED}},$$

де $U_{ж}$ — напруга живлення (в даному випадку максимальна напруга, яка присутня на виводі I/O плати Arduino, коли на ньому присутній рівень логічної одиниці, а саме 5 В); U_{LED} — напруга, яку потрібно отримати на світлодіоді в Вольтах; I_{LED} — струм, який потрібно отримати на світлодіоді, в Амперах.

Після розрахунку опору (в Омах) обираємо резистор з ряду стандартних номіналів, який найближче до розрахованого значення. Орієнтовні параметри струму та робочі напруги світлодіодів, які використовуються на цьому занятті, наведено в табл. 7.1.

Таблиця 7.1 — Параметри світлодіодів різних кольорів

Колір LED	**Висока яскравість**		**Помірна яскравість**	
	Напруга, В	**Струм, мА**	**Напруга, В**	**Струм, мА**
Червоний	1,9–2	10–20	1,9	4,5
Жовтий	1,9–2	10–20	1,9	4,5
Зелений	1,9–2	10–20	1,9	14
Синій	2,8–3	7–15	2,7	2,2
Білий	2,8–3	7–15	2,6	0,7

В режимах індикації доцільно експлуатувати світлодіоди з помірною яскравістю, особливо це стосується синього та білого світлодіода. Тому при розрахунках використовувати дані для помірної яскравості світлодіодів.

Пояснення до програми

Новими функціями та конструкціями для Вас може бути:

int

Визначає змінну, яка може містити тільки цілі числа з діапазону -32768 до 32767, на відміну від константи, створеної за допомогою **define** значення змінної можна змінювати в процесі виконання програми.

for (*initialization*; *condition*; *increment*) {функції, які виконуються в циклі}

Ця конструкція дозволяє створити цикл (тобто повторювати вказані функції певну кількість разів, функції, які повторюються повинні розміщуватися в фігурних дужках {}). ***Initialization*** виконується один раз і використовується для встановлення початкового значення змінної. ***Condition*** — тут вказується умова не виходу з циклу. ***Increment*** — вказується операція над змінною з поля ***Initialization*** при кожній ітерації, наприклад її збільшення чи зменшення. Наприклад:

```
for (int i = 2; i<=6; i++){
   pinMode(i, OUTPUT);
 }
```

Словами це можна описати наступним чином: змінна *«i»* дорівнює 2, якщо змінна *«i»* менше або рівня 6, то виконується функція *pinMode(i, OUTPUT)*, після цього до *«i»* додається 1 (саме це означає запис *i*++), знову порівнюємо і з 6, і якщо умова все ще виконується, то знову повторюємо функцію *pinMode(i, OUTPUT)*. Тобто в нашому випадку мікроконтролер в результаті виконає наступні дії:

```
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
pinMode(6, OUTPUT);
```

Тобто 5 разів викличе функцію *pinMode* з різним значенням *«i»*

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 7.2.

Таблиця 7.2 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	Номінал (тип)	Кількість, шт
Світлодіоди	різнокольорові	5
Резистори	680 Ом	2
	220 Ом	1
	1 кОм	1
	3,3 кОм	1
Гнучкі провідники	*male-male*	8

2. Розрахувати номінали резисторів *R*1–*R*5 для підключення світлодіодів відповідних кольорів згідно теоретичних відомостей (для помірної яскравості) та зібрати на макетній платі схему згідно з рис. 6.4 (Додаткова інформація про макетну плату наведена в теоретичних відомостях до заняття №6). Таблицю кольорового маркування резисторів наведено в Додатку А. Номера цифрових виводів можна використовувати довільні (при внесенні відповідних змін до програми).

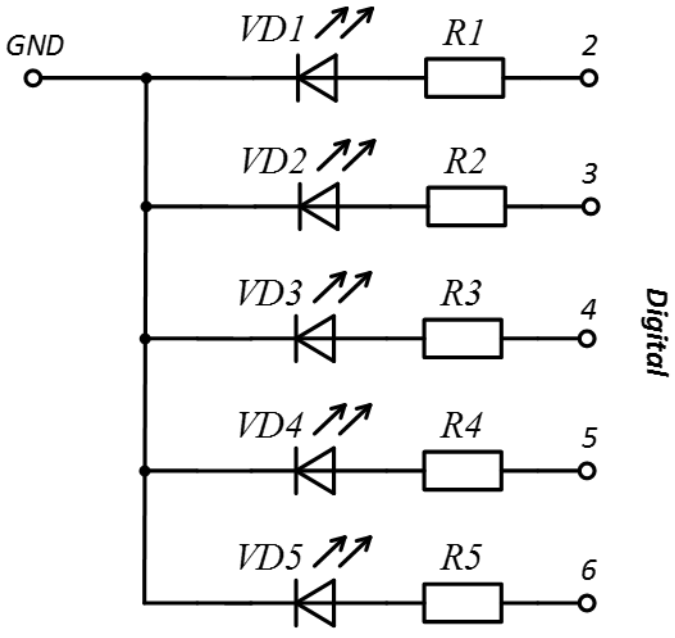


Рисунок 7.1— Схема підключення світлодіодів до *Arduino*

Приклад збирання схеми для двох світлодіодів наведено у попередньому занятті!

3. Продемонструвати зібрану схему викладачу. Підключити *Arduino* до комп'ютера.

4. Запустити *Arduino IDE*. У вікно коду програми вставити наведений нижче код:

```
 // Визначення виводів, до яких підключено світлодіоди та кнопки
#define led_pin_0 2
#define led_pin_1 3
#define led_pin_2 4
#define led_pin_3 5
#define led_pin_4 6

void setup() {
 // put your setup code here, to run once:
 // Ініціалізація виводів, до яких підключено світлодіоди, як виходи
 for (int i = led_pin_0; i<=led_pin_4; i++){
  pinMode(i, OUTPUT);
 }
}

void loop() {
 // put your main code here, to run repeatedly:

 // Реалізація програми миготіння світлодіодами
  for (int i = led_pin_0; i<=led_pin_4; i++){
    digitalWrite(i, HIGH);
    delay(200);
    digitalWrite(i, LOW);
    delay(100);
}
}
```

5. Вивантажити програму до платформи Arduino та переконатися в її працездатності. В разі необхідності провести потрібне налаштування. Розібратися як працює дана програма.

6. Додати ще один світловий ефект у вигляді блимання всіх світлодіодів одночасно (не менше 10 блимань). Для цього після завершення циклу, вставити паузу на 2 секунди та додати новий цикл в тілі якого записувати у відповідні цифрові виходи логічну одиницю (світлодіод засвітиться), через певну паузу записати логічний нуль (світлодіод згасне), знову зробити паузу.

7. Продемонструвати остаточний варіант програми викладачу. Дати відповіді на контрольні питання.

8. Записати до *Arduino* порожню програму (для цього виконати «Файл/Створити» та створений скетч, який містить виключно пусті функції *void setup()* та *void loop()*, вивантажити до *Arduino*) та розібрати зібрану схему. Повернути макет та ЕРЕ викладачу.

Контрольні питання

1. Які основні правила безпеки при роботі з платформою *Arduino*?

2. Як працює Ваша програма?

3. Навіщо потрібен обмежувальний резистор?

4. Як розрахувати опір обмежувального резистора?

5. Яка напруга встановлюється на виводі *Arduino*, якщо туди записали логічну одиницю?

6. Яка напруга встановлюється на виводі *Arduino*, якщо туди записали логічний нуль?

ЛАБОРАТОРНА РОБОТА № 8
ВВЕДЕННЯ ТА ВИВЕДЕННЯ ЦИФРОВИХ ДАНИХ. ЧАСТИНА 2

Мета роботи: навчитися працювати з цифровими *I/O* – виводами *Arduino*. Практичне застосування операторів умови та циклу. Створення схеми десяткового лічильника.

Теоретичні відомості

Введення цифрових даних

Під введенням цифрових даних розуміємо аналіз сигналу, поданого на один з цифрових входів *Arduino* на відповідність його рівням логічного нуля (*false*) чи одиниці (*true*). При чому прийнято, що рівень напруги від 3 до 5 В сприймається платформою *Arduino* як логічна одиниця, якщо ж напруга менше 3 В, то як логічний нуль.

Таким чином, при викликанні функції:

***val* = digitalRead(*pin*)**

Присвоює змінній ***val*** значення логічної одиниці (*true*), якщо до виводу за номером ***pin*** мікроконтролера прикладено позитивну напругу від 3 до 5 В, та значення логічного нуля (*false*), якщо прикладено напругу від 0 до 1,5 В.

Введення цифрових даних використовується для керування роботою мікроконтролера або приймання даних від інших мікроконтролерів чи цифрових мікросхем.

Підключення кнопки

Керувати роботою *Arduino* можна за допомогою звичайних кнопок (кнопка на схемі позначається як *SB*), які підключаються до цифрових *I/O*–виводів. Існує два основних способи підключення кнопки до виводу Arduino, рис. 8.1. В першому по замовчуванню на виводі встановлена напруга 5 В (рис. 8.1а), і відповідно мікроконтролер постійно буде зчитувати рівень логічної одиниці з виводу і тільки після натиснення кнопки — рівень логічного нуля. В другому — навпаки, по замовчуванню подається нульовий потенціал, а при натисненні кнопки — +5В (рис. 8.1б), відповідно постійно зчитується логічний нуль, а при натисненні — логічна одиниця.

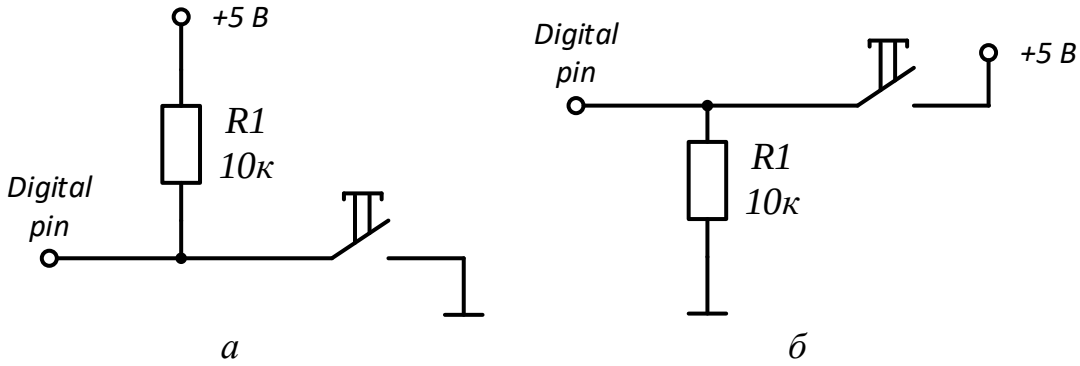


Рисунок 8.1 — Підключення кнопки

Резистор в наведеній схемі називається підтягувальним (*pull-up resistor*), і потрібен для того, щоб встановити значення напруги по замовчуванню, адже без даного резистора стан виводу до натискання кнопки буде важко спрогнозувати, адже на нього будуть впливати різні електромагнітні завади. Номінал даного резистора розраховується з врахуванням необхідної швидкодії, проте в межах даного курсу швидкодія порту нас не цікавить і тому резистор береться будь-який з діапазону 1–10 кОм.

Зовнішній вигляд кнопки та її схема наведена на рис. 8.2

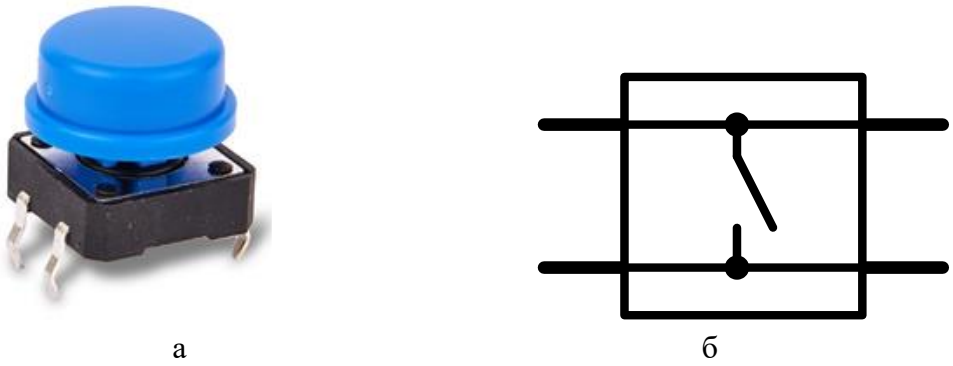


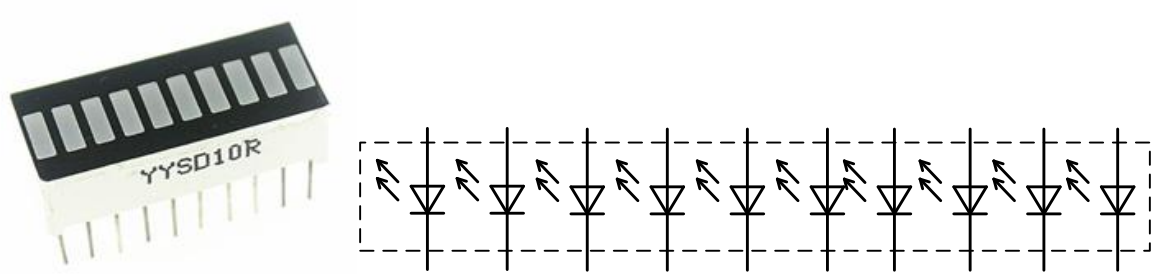
Рисунок 8.2 — Зовнішній вигляд кнопки (а) та її внутрішня схема (б)

Кнопка є механічним контактом, тому їй притаманна шкідлива риса, яка називається «вібрація контактів» (англ. *contact bounce*) — небажане тремтіння контактів в момент комутації, що призводить до їх швидкого замикання та розмикання. Цей процес може тривати до декількох мілісекунд і якщо мікроконтролер буде зчитувати дані з кнопки в цей час, то може фіксуватися багатократне натиснення чи помилковий результат. Для усунення впливу негативного явища використовують апаратні (за допомогою тригерів чи RC–ланок) та програмні засоби. Під час виконання лабораторних робіт будемо використовувати програмний засіб, причому найпростіший його варіант, а саме встановлення паузи після першого спрацювання кнопки, цей

спосіб має свої недоліки, але вони не критичні в процесі роботи даних схем.

Підключення десятисегментного індикатора

Десятисегментний світлодіодний індикатор складається з десяти окремих світлодіодів з прямокутними лінзами зібраними в одному корпусі і призначений для індикації рівнів чи значень величин (рис. 8.3).



а б

Рисунок 8.3— Десятисегментний світлодіодний індикатор (а) та його схема (б)

Зверніть увагу! Аноди таких світлодіодів розміщені зі сторони напису на корпусі індикатору. Зрозуміло, що як і звичайний світлодіод десятисегментний індикатор потребує обмежувального резистора, причому окремого на кожен сегмент. Номінал резисторів можна обрати в діапазоні від 500 Ом до 1 кОм.

Пояснення до програми

Новою конструкцією для вас є оператор умовного переходу:

if (умова)

{

// функції, які необхідно виконати при виконанні умови, якщо умова не виконується, то ці функції пропускаються

}

Для задавання умови можна використовувати звичайні оператори порівняння: == — рівно, != — не рівно, < — менше, > — більше тощо. Також можна використовувати логічні оператори: && — логічна функція «і», || — логічна функція «або», ! — логічна функція «ні». Тобто все як в мові C/C++, і точно так же можна використовувати конструкції **if...else**, синтаксис можна знайти на вже відомому Вам сайті: https://www.arduino.cc/reference/en/ або в Додатку В.

На цьому занятті умова потрібна для прийняття рішення натиснута кнопка чи ні. Для цього розглянемо оператор:

```
  if (!digitalRead(pin)) {
// Функції, які потрібно виконати при натисненні кнопки
  }
```

Яка ж тут умова? Все просто **digitalRead(*pin*)** буквально означає «зчитати цифрове значення з виводу з номером *pin*». Для того, щоб зрозуміти що власне буде зчитано з виводу *pin* необхідно проаналізувати схему рис. 8.1а.

Якщо кнопка не натиснута, то на 8 вивід через резистор потрапляє напруга живлення +5В, що відповідає рівню логічної одиниці, якщо ж кнопка натиснута, то на вивід подається «земля», тобто рівень логічного нуля. Тобто в нашому випадку якщо кнопка не натиснута, то **digitalRead(*pin*)** повертає значення «1», якщо ж в момент виклику цієї функції кнопка буде натиснута, то «0». Оператор **if** трактує логічну «1», як істину (***true***), тобто виконання умови, а «0», як хибу (***false***) або не виконання умови.

Відповідно, для такої схеми включення кнопки, умова буде виконуватися постійно, але ж нам потрібно, щоб умова виконувалася лише при натисненні кнопки. На допомогу приходить логічна операція «ні», або ж інверсія (щоб виконати «інверсію» перед умовою встановлено знак оклику «!»). І тепер зчитана «1» перетвориться на «0» і навпаки.

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 8.1.

Таблиця 8.1 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	Номінал (тип)	Кількість, шт
Світлодіодний індикатор	10 сегментний	1
Кнопки	–	2
Резистори	680 Ом чи 1 кОм	10
	10 кОм	2
Гнучкі провідники	male-male	14
Жорсткі перемички	–	набір

2. Зібрати на макетній платі схему згідно з рис. 8.4 та продемонструвати зібрану схему викладачу.

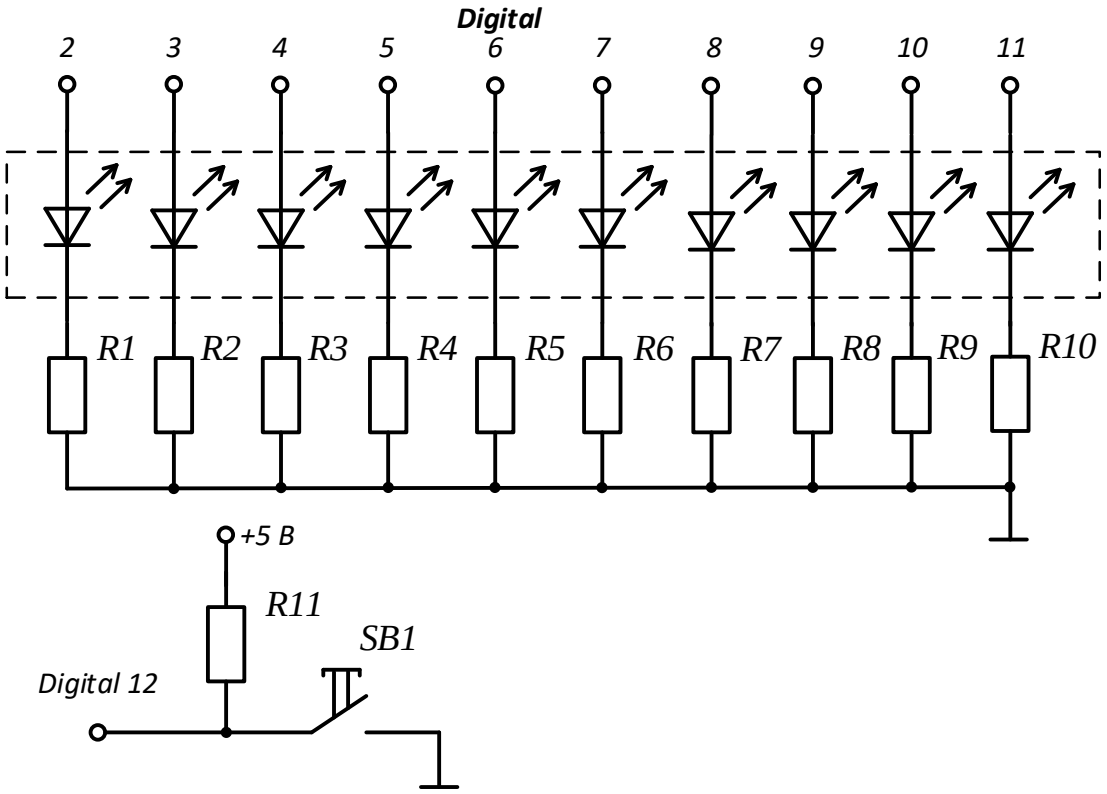


Рисунок 8.4 — Схема десяткового лічильника

3. Запустити *Arduino IDE* та скопіювати до нього наведену нижче програму:

```
#define led_pin_1 2
#define led_pin_10 11
#define up 12

int number = 0;
int out = led_pin_1;

void setup() {
  // ініціалізація виводів плати, як виходів
  for (int i = led_pin_1; i <= led_pin_10; i++) {
    pinMode(i, OUTPUT);
  }

// ініціалізація виводу як входу
  pinMode(up, INPUT);
}

void loop() {
  // опитування виводу, до якого підключено кнопку:
```

```
 if (!digitalRead(up)) {
  number++;
  delay (200); // Затримка для усування ефекту вібрації контакту
 }
 for (int i = led_pin_1; i <= led_pin_10; i++) {
  if(i <= number+1) digitalWrite(i, HIGH);
  else digitalWrite(i, LOW);
 }
}
```

4. Вивантажити програму до *Arduino* та перевірити її працездатність, а саме: кожне натискання кнопки повинно збільшувати на одиницю кількість засвічених світлодіодів. Розібратися як працює дана програма.

5. Відключивши *Arduino* від комп'ютера, доповнити схему ще однією кнопкою з підтягувальним резистором (при чому за варіантом рис. 8.1б), підключити її до 13 виводу мікроконтролера. Показати змінену схему викладачу.

6. Доповнити програму так, що при натисненні на нову кнопку кількість засвічених світлодіодів зменшувалася на одиницю. Для цього необхідно ініціалізувати 13 вивід як вхід та вставити нову умову та результат виконання умови для опитування виводу, до якого підключено кнопку.

7. Вивантажити програму на *Arduino* та переконатися в її працездатності. При потребі провести коригування програми. Продемонструвати фінальний варіант програми викладачу.

8. Доробити програму так, щоб значення змінної *number* не виходило за межі 0…10 при будь-якій кількості натиснення кнопок.

*9. Розробити програму, яка буде виводити на світлодіоди число натискань на кнопку у двійковому коді.

10. Завантажити до *Arduino* порожній скетч, розібрати схему та повернути макет та ЕК викладачу.

Контрольні питання

1. Як працює Ваша програма?

2. Що таке вібрація контактів та як можна з нею боротися?

3. Навіщо використовується підтягувальний резистор?

4. Що за замовчуванням буде повертати функція *digitalRead(pin)*, якщо до виводу *pin* буде підключено кнопку за варіантом рис. 8.1а (рис.8.1б) і що буде повертатися при натисненні кнопки?

ЛАБОРАТОРНА РОБОТА № 9
ЗЧИТУВАННЯ АНАЛОГОВИХ СИГНАЛІВ. ПЕРЕДАВАННЯ ДАНИХ НА КОМП'ЮТЕР

Мета роботи: навчитися працювати з аналого-цифровим перетворювачем для зчитування неперервних сигналів, які подаються з давачів. Ознайомитися з методами передавання даних на комп'ютер з платформи *Arduino*. Проведення вимірювання параметрів електричних кіл за допомогою *Arduino*.

Теоретичні відомості

Аналоговий сигнал — сигнал (напруга, струм тощо), неперервний на всьому проміжку часу. Звичайно цифрові схеми не можуть працювати з аналоговими сигналами, так як оперують виключно двійковим кодом. Тому для зчитування даних з аналогових давачів використовують аналого-цифровий перетворювач (АЦП). Arduino містить вбудований АЦП, який працює з виводами A0–A5.

Аналого-цифровий перетворювач

Аналого-цифровий перетворювач, АЦП (англ. *Analog-to-digital converter, ADC*) — пристрій, що перетворює вхідний аналоговий сигнал в цифровий код, який кількісно характеризує амплітуду вхідного сигналу [8].

Одним з основних параметрів АЦП є розрядність, яка характеризує кількість дискретних значень, які АЦП може видати на виході та вимірюється в бітах. Якщо розрядність АЦП складає $n = 10$, то максимальна кількість дискретних значень може бути $2^n = 2^{10} = 1024$. З даної розрядності випливає і мінімальне значення напруги, яке може бути розрізнене при перетворенні (роздільна здатність за напругою або крок квантування): $h = U_{оп} / 2^n$, де $U_{оп}$ — опорна напруга.

Процес перетворення аналогового сигналу в цифровий полягає в наступному: діапазон від 0 В до величини опорної напруги $U_{оп}$ розбивається на проміжки, кількість яких залежить від розрядності АЦП, кожному проміжку присвоюється цифровий код, значення вхідної напруги через інтервали τ порівнюється з встановленими рівнями і найближчий рівень приймається як поточне значення вхідного сигналу.

На приклад на рис. 9.1 проілюстровано процес аналого-цифрового перетворення для трирозрядного АЦП. Опорна напруга на рівні 1 В ділиться

на $2^3 = 8$ рівнів, і кожному з них присвоюється відповідний двійковий код: 0 В — 000; 0,125 В — 001; 0,25 В — 010 … 0,875 — 111. На вхід АЦП подається неперервна напруга $U(t)$, відліки беруться з інтервалом τ, тобто в точках $t_n = n \cdot \tau$, де n — 1, 2, 3 …. Таким чином, для наведено на рисунку неперервного сигналу отримаємо наступну послідовність двійкового коду: 001, 010, 011, 100, 101, 101, 110, 111, 111, 111.

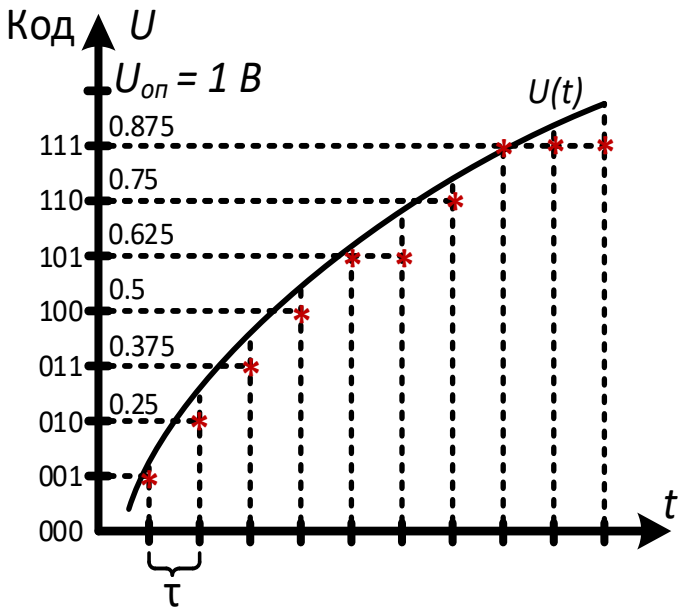


Рисунок 9.1 — Ілюстрація роботи аналого-цифрового перетворювача

Для визначення яка ж власне напруга була подана на вхід АЦП в певний момент, достатньо перемножити отримане значення АЦП з величиною h, наприклад, якщо з АЦП отримано код 101 (що відповідає десятковому числу 5), то на вході була присутня напруга $0{,}125 \cdot 5 = 0{,}625\,B$.

Більш докладно про принцип роботи АЦП та його параметри Ви ознайомитеся на старших курсах в дисципліні «Інформатика 2» та «Дизайн цифрових та аналогових схем».

Апаратна платформа *Arduino UNO* має 6 аналогових входів, які позначаються A0 – A5, сигнали з них подаються на 10-розрядний АЦП. Опорна напруга АЦП може змінюватися, проте за замовчуванням використовується $U_{оп} = 5\,B$.

Для зчитування даних з аналогового входу використовується функція **analogRead(*pin*)**, де ***pin*** — номер виводу, з якого необхідно зчитати значення напруги. Можна використати позначення 0 – 5 або ж A0 – A5. Функція

повертає ціле число в діапазоні від 0 до 1023, яке прямо пропорційна прикладеній до аналогового входу напруги.

Передавання даних

Для зв'язку апаратної платформи з комп'ютером використовується універсальний послідовний порт *UART* (англ. *universal asynchronous receiver/transmitter* — універсальний асинхронний приймач/передавач). Він пов'язаний з виводами 0 (*RX*) та 1 (*TX*) та забезпечує зв'язок *Arduino* з комп'ютером через *USB*. Власне тому, якщо Ви плануєте використовувати передавання даних до комп'ютера, то цифрові виводи 0 та 1 повинні бути вільними.

Зверніть увагу! При обміні даними між комп'ютером та *Arduino* використовується *USB*–кабель та, відповідно, *USB*–шина. Проте платформа *Arduino*, як і багато інших апаратних платформ, для операційної системи представляється в якості пристрою з інтерфейсом *RS*-232, більш відомим як *COM*-порт (послідовний порт). Власне і процес передавання даних є емуляцією передавання даних за допомогою *COM*-порту.

Клас Serial

В об'єктно-орієнтованому програмуванні клас — це спеціальна конструкція, яка використовується для групування пов'язаних змінних та функцій.

Платформа *Arduino* містить декілька класів, які допомагають в програмуванні складних задач. Одним з таких класів є клас *Serial*, основним завданням якого є забезпечення зв'язку плати *Arduino* з комп'ютером чи іншими пристроями. Даний клас містить низку функцій, з якими можна ознайомитися в Додатку В, всі вони призначені для передавання та аналізу даних від Arduino до комп'ютера та навпаки. Розглянемо більш докладно функції, які будемо використовувати в цій лабораторній роботі.

begin() — функція, яка задає швидкість передавання даних по послідовному інтерфейсу в бітах за секунду. Синтаксис **Serial.begin(*speed*)**, де ***speed*** традиційно задається з ряду 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 або 115200. Дана функція є обов'язковою, якщо заплановане передавання даних через послідовний інтерфейс. Записується один раз в функції **void setup ()**.

print() та **println()** — виводить через послідовний порт інформацію у вигляді, зрозумілому для людини. Функція **println()** додатково переводить

курсор на початок наступного рядка. Наприклад, функція **Serial.print(**"Hello world."**)** виведе напис «Hello world.».

Звичайно ж за допомогою функції **Serial.print** можна виводити дані різного типу: текстові, числові, значення змінних тощо. На відміну від мови C/C++ не обов'язково вказувати формат виведення даних, проте, якщо Вам потрібно отримати специфічний формат, то вказати його можна. Крім того, функція **print()** та **println()** повертають кількість виведених байт.

Для перегляду виведеної через *COM*-порт інформації на комп'ютері необхідно в *Arduino IDE* викликати вікно монітору послідовного порту в меню «Інструменти» та правильно встановити швидкість передачі даних (рис. 9.2).

Часто передавання даних до монітору послідовного порту використовується для налагодження програми, щоб з'ясувати стан змінних в певні моменти часу.

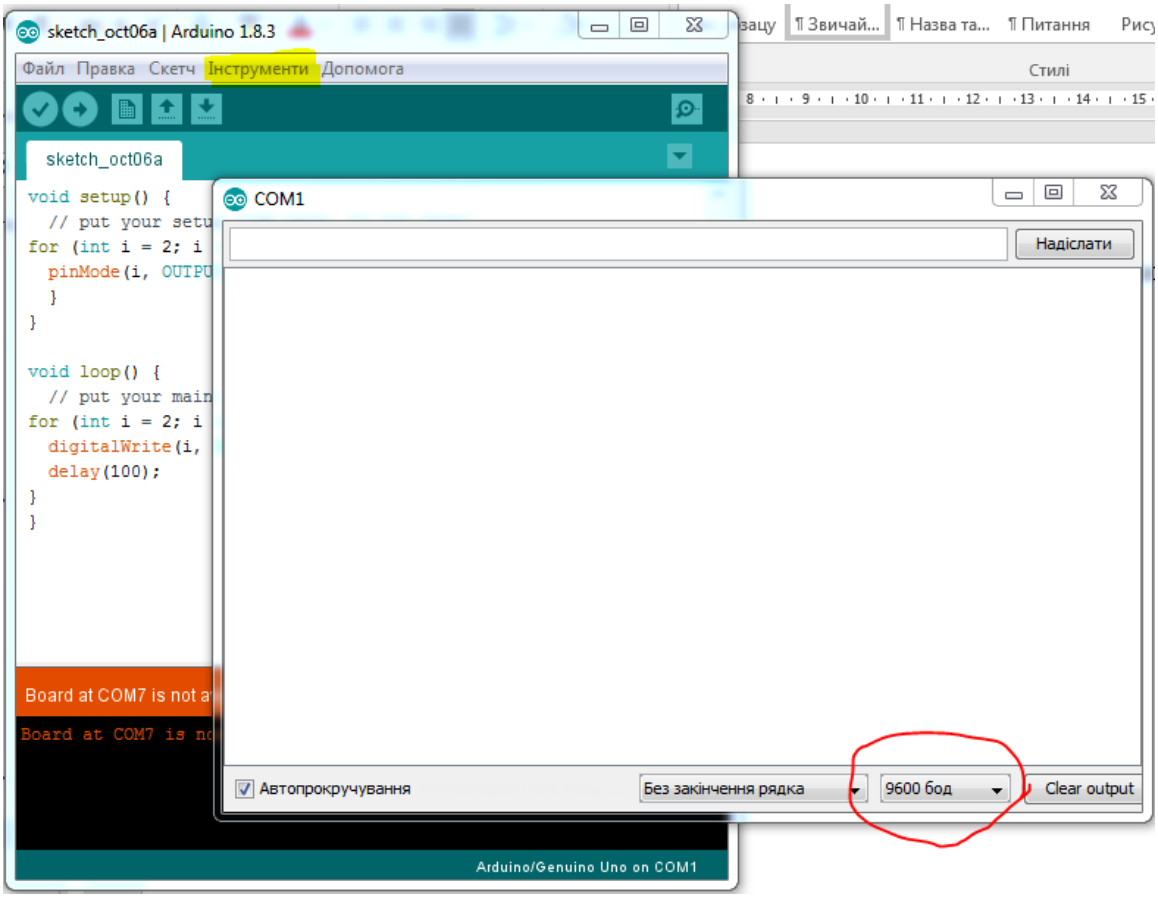


Рисунок 9.2 — Монітор послідовного порту в Arduino IDE

Також для перегляду інформації можна скористатися модулем для побудови графіку послідовного порту, який можна викликати в меню «Інструменти» обравши *Serial Plotter*.

Подільник напруги

Подільник напруги — лінійна електронна схема, напруга на виході якої (V_{out}) складає частину напруги на вході (V_{in}). Найпростіший подільник напруги складається з двох послідовно увімкнених резисторів (рис. 9.3).

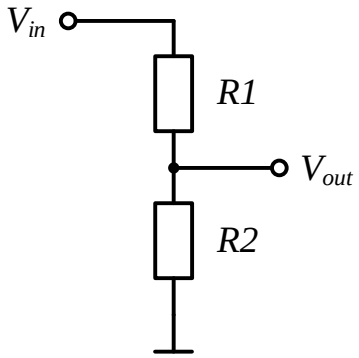


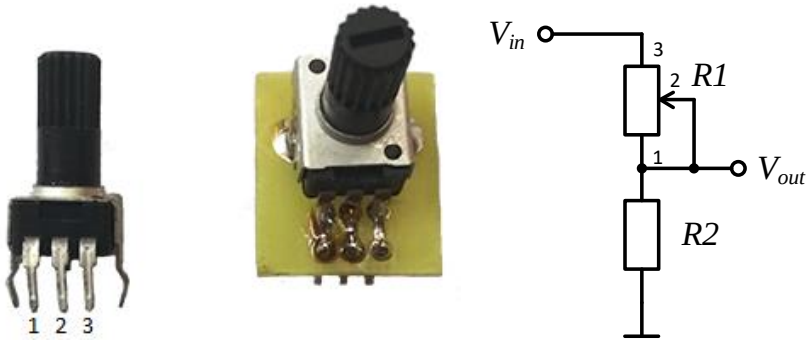
Рисунок 9.3 — Схема подільника напруги

Значення V_{out} розраховується за формулою:

$$V_{out} = V_{in} \cdot \frac{R2}{R1 + R2}. \quad (9.1)$$

Подільники напруги часто використовуються для підключення аналогових резистивних давачів до мікроконтролера. Адже резистивні давачі змінюють значення свого опору в залежності від певних фізичних впливів, наприклад, терморезистор змінює свій опір в залежності від температури навколишнього середовища, а фоторезистор — від освітленості. Тому якщо такий резистивний давач підключити замість резистора *R1* у дільник напруги, то зміна його опору буде змінювати значення напруги V_{out}, яку вже можна без проблем зчитати за допомогою АЦП.

Змінний резистор



а б в

Рисунок 9.4 — Змінний резистор (а), модуль на його основі (б) та приклад схеми його включення (в).

Для регулювання електричних параметрів кіл використовують змінні резистори (рис. 9.4а), опір між рухомим контактом 2 та нерухомими контактами 1 та 3 змінюється в залежності від положення рухомого контакту.

На рис. 9.4в показано один зі способів включення змінного резистора, тобто коли один з бічних виводів резистора з'єднаний з виводом центрального рухомого контакту. При такому включення опір між виводом 3 та з'єднаними між собою 1 та 2 змінюється в залежності від положення рухомого контакту від мінімального до максимального (максимальний опір змінного резистора вказується на його корпусі).

Зверніть увагу! Опір кодується трьома цифрами: перші дві — значення опору, третя — степінь десяткового множника. Наприклад, напис 102 означає, що опір резистора складає $10 \cdot 10^2 = 1000$ Ом, або ж 1 кОм.

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 9.1.

Таблиця 8.1 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	Номінал (тип)	Кількість, шт
Змінний резистор	1к, 5к, 10к	1
Резистори	680 Ом	2
	1 кОм	2
	3,3 кОм	2
	10 кОм	2
Мультиметр	–	1
Гнучкі провідники	*male-male*	10
Жорсткі перемички	–	набір

2. Зібрати подільник напруги згідно з рис. 9.3, використовуючи будь-яку пару однакових резисторів, під'єднавши вихід V_{out} до будь-якого аналогового входу, а вхід V_{in} до +5 В.

3. Скопіювати нижченаведену програму до *Arduino IDE*, провести необхідні зміни та вивантажити скетч у плату.

```
#define analog_in A0 // порт, з якого плануємо зчитувати дані

void setup() {
  // put your setup code here, to run once:
Serial.begin (9600); //Встановлюємо швидкість передачі даних
```

```
}

void loop() {
  // put your main code here, to run repeatedly:
int in = analogRead(analog_in); // зчитуємо дані з аналогового входу
Serial.print ("ADC = ");
Serial.println (in);
delay(200);
}
```

4. Відкрити монітор послідовного порту та переконатися, що на комп'ютер виводиться значення зчитане з аналогового входу.

5. Використовуючи формулу розрахунку напруги за даними АЦП дописати програму так, щоб вона розраховувала напругу на виході дільника та відправляла її на комп'ютер у вигляді: ADC = 532 U = 2.60V.

6. За допомогою мультиметра виміряти напругу на виході дільника і порівняти її з даними, які отримані за допомогою Arduino та розрахунковими даними (розрахувати відносну похибку).

7. Змінюючи резистори в плечах дільника напруги відслідкувати як зміна значень їх опорів впливає на значення напруги на виході дільника (див. контрольні питання).

8. Визначити номінал змінного резистора, який Вам видали, та підібравши найбільш схожий серед наявних резисторів зібрати схему дільника згідно з рис. 9.4в.

9. Використовуючи формулу (9.1) переробити програму так, щоб вона виводила значення поточного опору змінного резистора R1, наприклад так: ADC = 532 U = 2.60V R1 = 4804 Om

10. Визначити яке мінімальне та максимальна значення напруги можна отримати на виході дільника напруги змінюючи опір змінного резистора та пояснити результат.

*11. Підключити вивід рухомого контакту змінного резистора до аналогового входу Arduino, а нерухомі до +5 В та GND. Переробити програму так, щоб вона визначала опір обох плечей змінного резистора та виводила його на комп'ютер. Дослідити як змінюється значення на виході дільника напруги при обертанні ротору змінного резистора. Контролюючи покази з Arduino встановити опір одного з плечей на рівні 500 Ом. Відключити рези-

стор та виміряти опір за допомогою мультиметра. Пояснити різницю у показах.

*12. Використовуючи мультиметр спробувати налагодити програму так, щоб значення опору виміряне за допомогою Arduino максимально співпадало зі значенням опору виміряного мультиметром (в програмі необхідно корегувати максимальне значення опору резистора та вхідну напругу).

13. Завантажити до *Arduino* порожній скетч. Розібрати схему та повернути макет та ЕК викладачу.

Контрольні питання

1. Як працює Ваша програма?

2. Що таке АЦП та як він працює?

3. Що таке подільник напруги? Навіщо їх використовують?

4. Яка напруга буде на виході подільника напруги, якщо $R1 = R2$, $R1 > R2$, $R1 < R2$?

ЛАБОРАТОРНА РОБОТА № 10
ОБРОБЛЕННЯ ДАНИХ З АНАЛОГОВИХ ДАВАЧІВ

Мета роботи: закріпити знання зі зчитування та оброблення аналогових сигналів. Написати програму для вимірювання температури в приміщенні за допомогою терморезистора.

Теоретичні відомості

Терморезистор

Для вимірювання температури часто використовують терморезистори (рис. 10.1), тобто резистори, які змінюють свій опір в залежності від температури навколишнього середовища.

Рисунок 10.1 — Зовнішній вигляд та схемне позначення терморезисторів

Залежність опору терморезистора від температури нелінійна, тому для визначення температури за опором терморезистора необхідно використовувати рівняння Стейнхарта–Харта [9]. Проте воно описує більш загальний випадок поведінки p-n переходу в залежності від температури. У випадку з терморезистором доцільно скористатися спрощеною формулою β–параметру:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \cdot \ln(\frac{R}{R_0}) \qquad (10.1)$$

де T — розрахована температура в Кельвінах;

T_0 — нормальна температура, 25 °C або ж 298.15 K (для розрахунку використовувати температуру в Кельвінах);

B — коефіцієнт β, обирається в залежності від терморезистора та наводиться в Datasheet;

R — значення опору терморезистора в залежності від температури навколишнього середовища

R_0 — значення опору терморезистора при нормальній температурі (25 °C), залежить від типу терморезистора.

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 10.1.

Таблиця 9.1 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	**Номінал (тип)**	**Кількість, шт**
Терморезистор	MF 52 AT 103 3950	1
Резистор	10 кОм	1
Гнучкі провідники	*male-male*	4
Жорсткі перемички	–	набір

2. Зібрати схему дільника напруги, який складається з терморезистора та звичайного резистора. Підключити вхід дільника до напруги 5 В, а вихід дільника до аналогового входу *Arduino*. Продемонструвати зібрану схему викладачу.

3. Написати програму, яка буде зчитувати дані з аналогового входу та розраховувати за ними значення температури навколишнього середовища, скориставшись формулою (10.1). Опір терморезистора розраховувати за формулою, яку вивести самостійно з формули, що описує дільник напруги (9.1). Натуральний логарифм розраховується за допомогою функції **log()**.

4. За допомогою функції **Serial.println()** передавати значення температури навколишнього середовища в градусах Цельсія на комп'ютер кожні 50 мс. Прослідкувати динаміку зміни температури за допомогою модуля Serial Plotter. Для зміни значення температури можна використовувати температуру тіла або дихання. Продемонструвати результат викладачу.

5. Доробити програму так, щоб вона виводила в монітор послідовного

порту інформацію у вигляді: «Опір терморезистора R = xx кОм; Температура T = xx,xx C». Для цього потрібно передавати почергово в порт текстову та цифрову інформацію за допомогою функцій **Serial.print()** та **Serial.println()**. Продемонструвати фінальний варіант програми викладачу.

6. Записати порожній Скетч до *Arduino*, розібрати схему та повернути макет та ЕК викладачу.

Контрольні питання

1. Як працює Ваша програма?

2. Що таке терморезистор?

3. Яким чином перевести температуру в Кельвінах в температуру в градусах Цельсія?

ЛАБОРАТОРНА РОБОТА № 11
ВИВЕДЕННЯ АНАЛОГОВИХ СИГНАЛІВ

Мета роботи: вивчити метод виведення аналогових сигналів з платформи *Arduino* за допомогою широтно-імпульсної модуляції. Розробити програму зміни яскравості світлодіода в залежності від навколишнього освітлення.

Теоретичні відомості

Як ви вже знаєте платформа Arduino, як і будь-який інший пристрій побудований на мікроконтролері, відноситься до цифрових пристроїв і, відповідно, оперує виключно рівнями логічного нуля та одиниці. Проте дуже часто виникає потреба керувати зміною напруги за допомогою мікроконтролера. Так, наприклад, зміною напруги можна регулювати частоту обертання двигуна, яскравість світіння світлодіода чи звичайної лампочки тощо. Для перетворення цифрового коду на пропорційну до нього напругу використовують цифро-аналогові перетворювачі (ЦАП, англ. *DAC — Digital-to-Analog Convertor*). Методів побудови ЦАП доволі багато тож докладне вивчення їх проводиться в курсі «Інформатика 2.» та «Дизайн цифрових та аналогових схем». На справді *Arduino* не містить повноцінного ЦАП, адже на аналогових виходах *Arduino* не формується постійна напруга, а генерується послідовність прямокутних імпульсів різної тривалості.

Широтно-імпульсна модуляція

Програмно-апаратна платформа *Arduino* містить широтно-імпульсний модулятор, за допомогою якого є можливість керувати середнім значенням напруги на аналоговому виході.

Широтно-імпульсна модуляція, ШІМ (англ. *pulse-width modulation, PWM*) — процес керування тривалістю високочастотних імпульсів за законом, який задає низькочастотний сигнал. Найчастіше ШІМ використовується для керування живленням різних пристроїв, адже середнє значення напруги імпульсної послідовності залежить від її коефіцієнту заповнення D (англ. *Duty cycle*), тобто відношення тривалості імпульсу τ до періоду їх слідування T, рис. 10.1:

$$D = \frac{\tau}{T}, \tag{10.1}$$

Коефіцієнт заповнення безрозмірна величина та зазвичай подається у відсотках, тобто відношення тривалості імпульсу до періоду множиться на

100%. Також використовується поняття шпаруватість Q — це відношення періоду до тривалості імпульсу, тобто величина обернена до коефіцієнту заповнення. Так, наприклад, для зображеної на рис. 10.1 прямокутної послідовності $T = 2$ мс, $\tau = 1{,}2$ мс, відповідно $D = 1{,}2 / 2 = 0{,}6$ або ж 60%. Відповідно середня значення напруги $U_{сер}$ складає 60 % від амплітудного, в даному випадку 5 В:

$$U_{сер} = 5 \cdot 0{,}6 = 3\text{В}$$

Платформа *Arduino* містить восьми розрядний ШІМ, тобто кодування напруги проводиться в межах від 0 до 2^8–1. Опорна напруга, як і в АЦП, складає 5 В. Частота слідування прямокутних імпульсів близько 500 Гц.

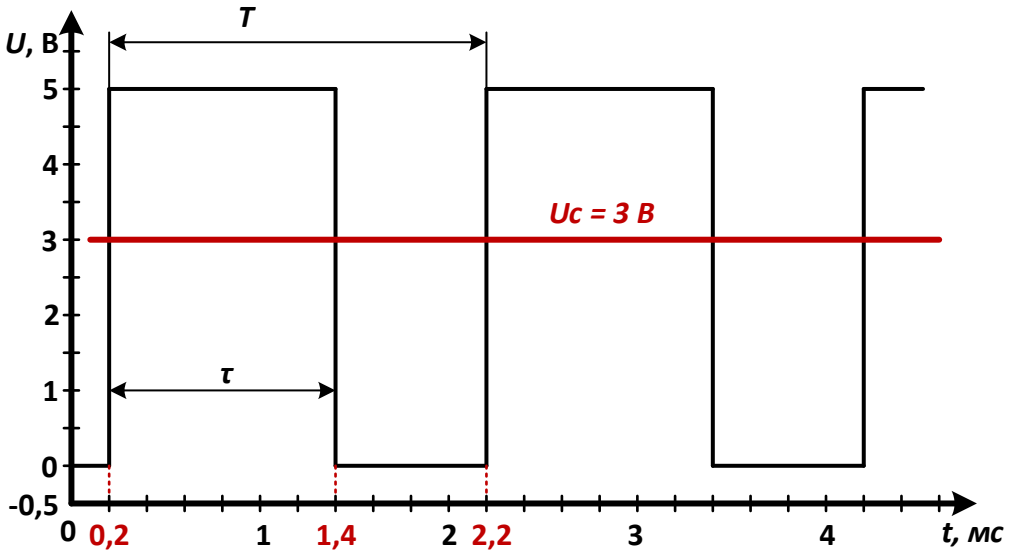


Рисунок 10.1 — Послідовність прямокутних імпульсів

ШІМ в *Arduino* може використовуватися тільки з деякими виводами, на платі вони позначені символом «~». Для *Arduino UNO* це виводи 3, 5, 6, 9, 10 та 11.

Для встановлення певного значення на відповідному виводі використовується функція **analogWrite(*pin, value*)**, де ***pin*** — номер виводу, на якому потрібно встановити значення напруги, ***value*** — число від 0 до 255, яке пропорційне значенню напруги. Наприклад, для встановлення на 11 виводі напруги 3 В, потрібно розрахувати значення ***value***: $3 \cdot 255 / 5 = 153$ та викликати функцію analogWrite(11, 153). На 11 виводі буде постійно (до наступного звернення до цього виводу) генерувати послідовність прямокутних імпульсів з частотою 500 Гц та коефіцієнтом заповнення 0,6.

Зверніть увагу! Після виконання функції analogWrite на відповідному

виводі буде послідовність прямокутних імпульсів, а не постійна напруга! Просто середнє значення при цьому буде відповідати встановленому. Для отримання постійної напруги цей сигнал необхідно пропустити через RC–ланку, тобто найпростіший фільтр низьких частот.

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 11.1.

Таблиця 11.1 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	**Номінал (тип)**	**Кількість, шт**
Фоторезистор	–	1
Світлодіод	–	1
Резистори	10 кОм	1
	680 Ом	1
Гнучкі провідники	male-male	5

2. Підключити світлодіод через обмежувальних резистор (680 Ом) до одного з аналогових виходів *Arduino*.

3. Використовуючи функцію **analogWrite** з різним значенням ***value*** переконатися, що Ви можете змінювати яскравість світіння світлодіода.

4. Написати програму, яка буде плавно збільшувати яскравість світіння світлодіода від мінімальної до максимальної, а потім від максимальної до мінімальної.

5. Зібрати дільник напруги з фоторезистором в одному із плеч та підключити його вихід до аналогового входу *Arduino*. Номінал другого резистора 10 кОм. Показати зібрану схему викладачу.

6. Написати програму, яка буде зчитувати дані з аналогового входу та передавати їх на комп'ютер. Простежити як змінюються дані в залежності від того, скільки світла падає на фоторезистор. Запам'ятати значення L_{max} та L_{min} — код для максимальної та мінімальної освітленості фоторезистора.

7. Доробити програму таким чином, щоб зміна освітленості фоторезистора впливала на яскравість світіння світлодіода. Тобто при повній освітленості фоторезистора світлодіод не світився, а при затемненні починав плавно збільшувати свою яскравість, пропорційно до степені затемнення. Для цього можна скористатися формулою:

$$LED = (L_{\max} - L) * 255.0 / (L_{\max} - L_{\min})$$

де L — зчитані дані з фоторезистора. L_{max} та L_{min} — максимальне та мінімальне зчитане значення при зміні освітленості фоторезистора.

8. Продемонструвати працюючу програму викладачу.

*9. Встановити на світлодіоді за допомогою функції **analogWrite** напругу на рівні 0,1 В. Пояснити, чому світиться світлодіод, хоча мінімальна напруга для його роботи повинна бути близько 2 В.

*10. Підключити аналоговий вхід *Arduino* до його аналогового виходу. Доповнити програму зчитуванням даних з аналогового входу та виведенням їх на комп'ютер. Пояснити результат.

11. Завантажити до *Arduino* порожній скетч, розібрати схему та повернути макет та ЕК викладачу.

Контрольні питання

1. Як працює Ваша програма?

2. Що таке ЦАП та ШІМ?

3. Як визначити середнє значення напруги, сформованої послідовністю прямокутних імпульсів?

4. Яка частота слідування прямокутних імпульсів ШІМ *Arduino*?

5. Яка розрядність ШІМ *Arduino*?

ЛАБОРАТОРНА РОБОТА № 12 КЕРУВАННЯ РОБОТОЮ RGB СВІТЛОДІОДА

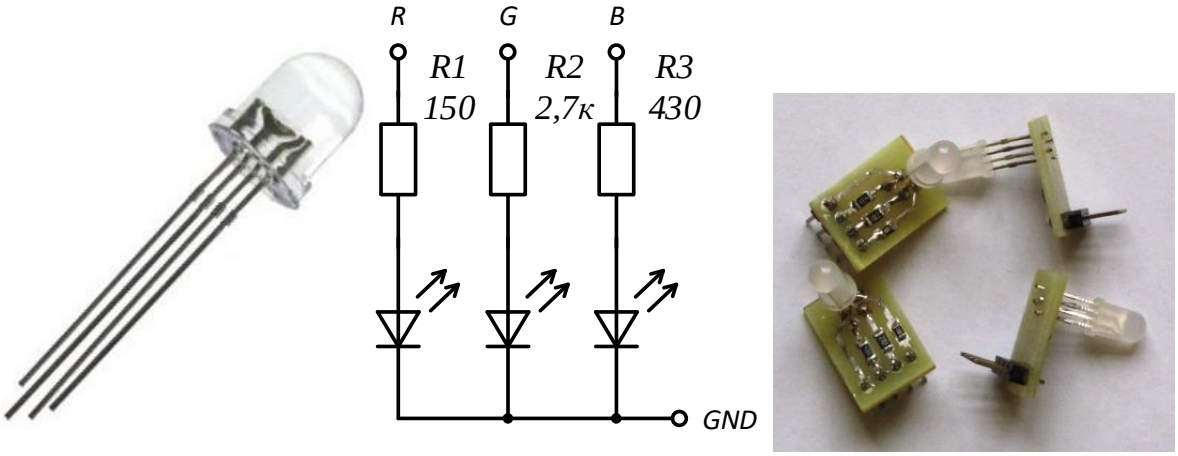
Мета роботи: закріпити знання з виведення аналогових сигналів за допомогою Arduino. Написати програму керування RGB світлодіодом.

Теоретичні відомості

RGB світлодіод

Для формування кольорового зображення на моніторах використовують так звану *RGB* колірну модель, яка полягає в тому, що змішуючи в певній пропорції основні кольори: червоний, зелений та синій, можна сформувати будь-який проміжний колір.

Для дослідження даного ефекту використовують *RGB*–світлодіоди, це такі світлодіоди, в корпусі якого знаходиться одразу три світлодіоди різних кольорів: червоний, зелений та синій (рис. 10.2а). Для зручності використання *RGB* світлодіодів часто їх встановлюють на окремих платах разом з обмежувальними резисторами. Зовнішній вигляд модулю та його схема показана на рис. 10.2.



а б в

Рисунок 10.2 — *RGB*–світлодіод (а), схема (б) та зовнішній вигляд модулю на його основі (в)

Для отримання основних кольорів (червоного, зеленого та синього) достатньо подати напругу +5 В на відповідний вивід, за умови, що спільний вивід підключено до *GND*. Одночасно подаючи напругу +5 В на декілька виводів можна отримати ще 3 додаткові кольори та білий колір, якщо одночасно подати напругу на всі три світлодіоди.

В цілому, *RGB* колірна модель передбачає зміну впливу яскравості кожного з основних кольорів на сумарний колір в діапазоні від 0 до 255 (тобто кожен колір містить 256 градацій яскравості), що в певному наближенні відповідає значенням, які можуть передаватися на аналоговий вихід *Arduino,* відповідно одночасно регулюючи середнє значення напруги на кожному світлодіоді теоретично можна отримати $256^3 = 16777216$ кольорів. Реальність, звичайно, вносить свої корективи, адже по перше середньостатистичне людське око здатне розрізнити тільки близько 10 млн. кольорів, а по друге — подібні світлодіоди не достатньо якісно змішують складові між собою. Крім того, яскравість світіння основних кольорів різна і, відповідно, отримати потрібний відтінок дуже складно.

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 10.1.

Таблиця 10.1 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	**Номінал (тип)**	**Кількість, шт**
Світлодіодний *RGB* модуль зі спільним катодом	-	1
Гнучкі провідники	*male-male*	5

2. Визначити, який вивід модулю відповідає за який з основних кольорів. Для цього підключити спільний вивід (він йде безпосередньо до світлодіода) до мінусу, а на решту по черзі подати +5 В. Отримати додаткові кольори, подаючи одночасно на декілька виводів напругу +5 В.

3. Створити програму, яка буде перебирати по черзі всі кольори веселки. Для цього потрібно по черзі змінювати напругу на потрібному виводі світлодіода у порядку, наведеному у таблиці 10.2.

Таблиця 10.2 — Формування кольорів веселки

Колір	**R**	**G**	**B**
Червоний	255	0	0
Помаранчевий	255	125	0
Жовтий	255	255	0
Зелений	0	255	0
Блакитний	0	255	255
Синій	0	0	255
Фіолетовий	255	0	255

4. Використовуючи декілька циклів, вдосконалити програму так, щоб зміна кольорів веселки була плавною. Тобто для отримання червоного кольору потрібно поступово змінювати значення, яке функція analogWrite подає на вивід, до якого підключено червоний світлодіод від 0 до 255, потім не змінюючи значення на червоному світлодіоді, збільшувати значення на зеленому від 0 до 125 і так далі. Продемонструвати робочу програму викладачу.

*5. Отримати у викладача три змінні резистори. Підключити їх до *Arduino*. Написати програму, яка буде змінювати яскравість кольорів *RGB* світлодіода в залежності від положення ротора змінного резистора (один резистор відповідає за один колір). Продемонструвати програму викладачу.

6. Записати до *Arduino* порожній скетч, розібрати схему та повернути макетну плату з ЕК викладачу.

Контрольні питання

1. Що таке *RGB* світлодіод?

2. Як працює Ваша програма?

3. Як формується кольорове зображення за допомогою *RGB* світлодіодів?

ЛАБОРАТОРНА РОБОТА № 13
ГЕНЕРУВАННЯ СИГНАЛІВ РІЗНОЇ ЧАСТОТИ

Мета роботи: вивчити методи генерування сигналів різної частоти за допомогою платформи *Arduino*. Дослідити електромагнітний випромінювач звуку. Ознайомитися з методами підвищення вихідної потужності за допомогою транзистора. Створення програми для керування електромагнітним випромінювачем звуку.

Теоретичні відомості

Для генерування сигналу у вигляді прямокутних імпульсів різної частоти (з коефіцієнтом заповнення 50%) використовуються внутрішній таймер/лічильник. Для виведення сигналу використовується функція **tone()**.

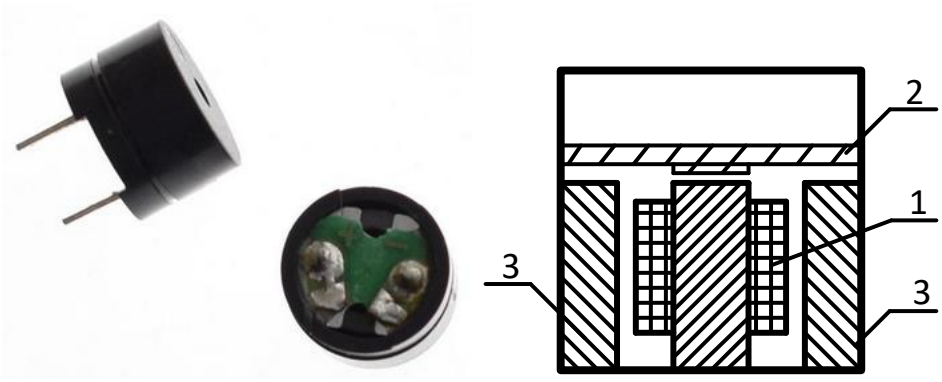
Використовується два варіанти синтаксису: **tone(*pin, frequency*)** та **tone(*pin, frequency, duration*)**, де ***pin*** — номер виводу, на який необхідно подати сигнал, ***frequency*** — частота сигналу в Герцах, ***duration*** — тривалість сигналу в мілісекундах. Якщо тривалість звучання не вказана, то генерування сигналу буде відбуватися до тих пір, доки не буде вимкнене функцією **noTone(*pin*)** або змінене функцією **tone()**.

Зверніть увагу! За допомогою функції **tone()** не можна одночасно подавати сигнали на різні виводи. Для виведення сигналу на інший вивід потрібно перш за все вимкнути сигнал на поточному виводі за допомогою функції **noTone(*pin*).**

Електромагнітний випромінювач звуку

Для відтворення не складних малопотужних звукових сигналів використовують електромагнітні випромінювачі звуку (рис. 13.1). Працює він дуже просто: прикладення змінної напруги до котушки 1 викликає зміну електромагнітного поля навколо, що в свою чергу призводить до коливання мембрани 2, також в конструкції обов’язково присутній постійний магніт 3.

Параметри даного звукового випромінювача наступні: внутрішній опір — 15 Ом, максимальний струм 50 мА, напруга — 1–2 В. Також кожен випромінювач звуку має свою амплітудно-частотну характеристику, тобто різні частоти випромінюються з різною потужністю. Невеликі за розміром випромінювачі мають яскраво виражену резонансну частоту, тобто таку частоту, на якій звук буде найгучніший.



а б

Рисунок 13.1 — Електромагнітний випромінювач: зовнішній вигляд (а), будова (б), де 1 — котушка індуктивності, 2 — мемебрана, 3 — магніт.

На пряму випромінювач звуку до Arduino підключати не можна, потрібен резистор мінімум на 200 Ом, але при цьому гучність звуку буде порівняно низькою. Тому доцільно використовувати транзистор, який може підсилювати струм (напругу) і в результаті збільшувати потужність, яку можна передати до навантаження. Типова схема ввімкнення транзистора наведена на рис. 13.2

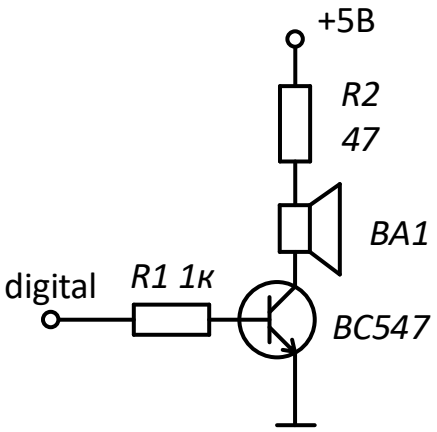


Рисунок 13.2 — Схема підключення звукового випромінювача до мікроконтролера через біполярний транзистор.

Схема працює наступним чином: якщо на цифровому виході присутня напруга +5 В, тобто рівень логічної одиниці, то струм протікає через резистор *R1* та перехід база-емітер транзистора, це призводить до відкривання транзистора, та протікання струму через ланку емітер-колектор транзистора, звуковий випромінювач *BA1*, резистор *R2*. При чому максимальний струм ланки залежить від опору звукового випромінювача *BA1* та резистора *R2*, проте цей струм не повинен перевищувати максимального струму колектора транзистора (для *BC*547 це 100 мА).

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 11.1.

Таблиця 11.1 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	**Номінал (тип)**	**Кількість, шт**
Випромінювач звуку	–	1
Світлодіод	Червоний	1
Кнопка	–	2
Транзистор	BC547	1
Резистори	47 Ом	1
	1 кОм	1
	680 Ом	1
	10 кОм	2
Гнучкі провідники	*male-male*	8
Жорсткі перемички	–	набір

2. Визначити резонансну частоту електромагнітного випромінювача звуку. Для цього зібрати схему згідно рис. 13.2 та підключити дві кнопки до цифрових виводів *Arduino* (не забуваємо про підтягувальні резистори!). Продемонструвати зібрану схему викладачу.

3. Написати програму, яка буде збільшувати частоту звучання при натисненні однієї кнопки або зменшувати її при натисненні другої. При цьому поточне значення частоти повинно виводитися в *COM*-порт.

4. Визначити приблизне значення резонансної частоти звукового випромінювача та частоту, на якій вже не чути випромінення звуку. Продемонструвати програму та визначені частоти викладачу.

5. Розробити програму, яка буде по натисненню однієї кнопки неперервно передавати сигнал *SOS* за допомогою азбуки Морзе до тих пір, доки не відбудеться вимкнення за допомогою іншої кнопки. Сигнал *SOS* складається з трьох крапок, трьох тире та трьох крапок. В якості крапки обрати сигнал тривалістю 100 мс, в якості тире — 200 мс. Пауза між сигналами — 100 мс, пауза між окремими сигналами *SOS* — 2 секунди. В якості частоти звучання обрати визначену раніше резонансну частоту.

***6. Додаткове завдання!** Скопіювати код, наведений в додатку Г, ро-

зібратися в ньому. Доробити програму так, щоб натисканням на кнопки можна було вибирати варіант мелодії для відтворення, а також зупиняти та запускати виконання мелодії.

8. Записати до *Arduino* порожній скетч, розібрати схему та повернути макет і ЕК викладачу.

Контрольні питання

1. Як працює Ваша програма?

2. Чому не можна підключати випромінювач звуку безпосередньо до виводу мікроконтролера?

3. Що робить схема, зображена на рис. 13.2?

ЛАБОРАТОРНА РОБОТА № 14
РОБОТА ІЗ ЗОВНІШНІМИ МОДУЛЯМИ. СЕМИСЕГМЕНТНИЙ ІНДИКАТОР

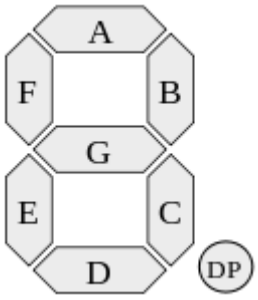
Мета роботи: навчитися працювати із зовнішніми модулями та сторонніми бібліотеками. Розробити програму для керування чотирьохрозрядним семисегментним індикатором.

Теоретичні відомості

Для платформи *Arduino* промисловість випускає велику кількість різноманітних модулів, які дозволяють значно розширити її можливості. Зазвичай такі модулі підключаються до *Arduino* за допомогою шини даних, що дозволяє суттєво зменшити кількість задіяних для керування модулем виводів мікроконтролера.

Семисегментний індикатор

Семисегментний індикатор — індикатор (рис. 14.1а), для відображення значення в якому використовується 7 окремих сегментів (часто 8 сегментів, хоч назва і зберігається — 7 сегментів відповідають за відображення цифри, а восьмий сегмент — за відображення десяткової крапки).



Сегмент	DP	G	F	E	D	C	B	A
Біт	7	6	5	4	3	2	1	0

а б

Рисунок 14.1 — Приклад семисегментного індикатора (а) та способу кодування (б)

В якості сегментів можуть виступати звичайні лампочки, розміщені за спеціальними розсіювачами світла, світловідбиваючі сегменти, які перевертаються механічним способом тощо. Проте зараз найчастіше в якості сегментів використовуються світлодіоди, при чому катоди (або аноди) всіх світлодіодів з’єднані між собою. Таким чином для керування однорозрядним семисегментним індикатором потрібно сім або вісім виводів мікроконтро-

лера (і така ж кількість обмежувальних резисторів). Зрозуміло що за допомогою однорозрядного індикатора можна відобразити максимум 10 десяткових цифр або ж 16 шістнадцяткових. Також використовуючи бітову послідовність можна закодувати будь-який символ (рис. 14.1б). Для цього в 8 бітному числі ставлять логічну 1 для сегменту, який повинен світитися. Це відповідає подаванню напруги на відповідний сегмент.

Семисегментні індикатори бувають різної розрядності, тобто кількість цифр може бути різною, при чому, наприклад, двохрозрядний семисегментний індикатор може мати лише на один вивід більше, ніж однорозрядний. Це досягається за рахунок того, що однакові сегменти з'єднані паралельно в різних розрядах (рис. 14.2).

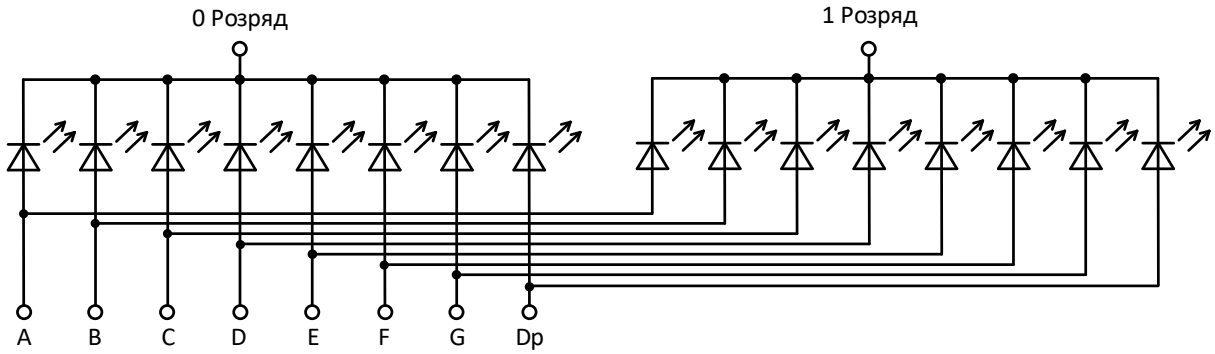


Рисунок 14.2 — Схема дворозрядного семисегментного індикатора.

Для керування такими індикаторами потрібно застосовувати динамічну індикацію або ж використовувати спеціальні драйвери. З динамічною індикацією Ви познайомитеся на старших курсах, а в даному курсі буде використано мікросхему-драйвер.

Одним з таких драйверів є мікросхема *TM*1637, яка дозволяє керувати чотирьохрозрядним семисегментним індикатором всього за допомогою двох проводів: *CLK* (*Clock*) — використовується для синхронізації та *DIO* (*Data Input/Output*) — використовується, власне для передавання даних. Звичайно ж потрібні ще два додаткові проводи для під'єднання +5 В (*Vcc*) та *GND*.

Підключення сторонніх бібліотек

Для підвищення зручності роботи програміста використовують додаткові бібліотеки, тобто набори функцій та класів, які розроблені іншими програмістами та виконують задані ними функції.

Arduino IDE теж містить функціонал для роботи з бібліотеками. Переглянути всі бібліотеки можна в меню Скетч/Додати Бібліотеку там є список вбудованих та рекомендованих бібліотек. Для зручності варто скористатися менеджером бібліотек, який знаходиться за тим же шляхом.

Для роботи з мікросхемою-драйвером *TM*1637 є декілька сторонніх бібліотек. Перед початком роботи ці бібліотеки потрібно підключити до *Arduino IDE*, зробити це можна за допомогою меню Скетч/Додати бібліотеку/Додати *ZIP* бібліотеку, якщо необхідно встановити бібліотеку, відсутню в списку встановлених чи рекомендованих бібліотек.

Для цього перш за все необхідно перевірити потребу у встановленні бібліотеки, запустивши менеджер бібліотек та в рядку пошуку ввести *TM*1637 (рис. 14.3). Якщо в результат пошуку негативний, необхідно скачати з FTP бібліотеку ***TM1637_PRE.zip*** та викликавши команду Скетч/Додати бібліотеку/Додати *ZIP* бібліотеку обрати скачаний файл.

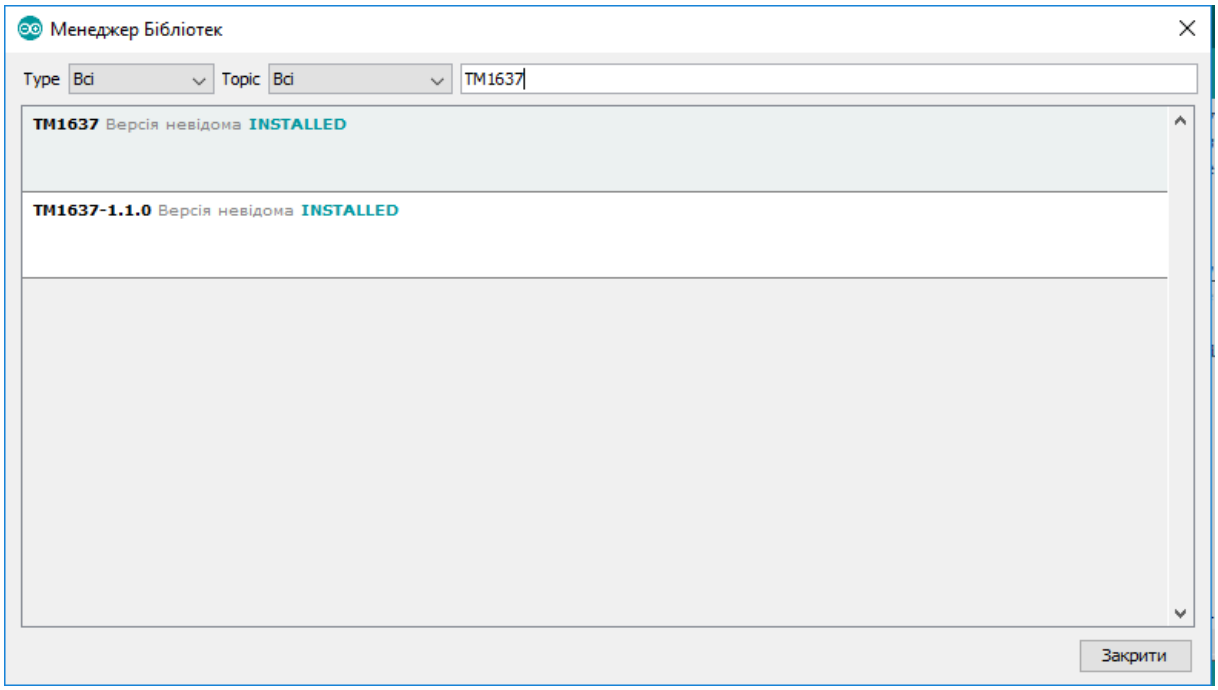


Рисунок 14.3 — Менеджер бібліотек *Arduino IDE*

Після встановлення бібліотеку необхідно підключати до тих скетчів, де заплановано використання її функцій за допомогою директиви #include.

Бібліотека TM1637

Для підключення бібліотеки до скетчу необхідно на його початку записати директиву #include "TM1637.h"

Бібліотека містить близько десяти функцій. В даній роботі будуть використовуватися наступні:

.init() — ініціалізація підключеної бібліотеки;

.set(*bright*) — встановлення яскравості світіння індикатора. Змінна ***bright*** може приймати значення від 0 до 7, типовими значеннями (вбудованими константами) є *BRIGHT_DARKEST*, яка рівна 0, *BRIGHT_TYPICAL* — 2 та *BRIGHTEST* — 7. **Зверніть увагу!** Яскравість змінюється після наступного запису даних на індикатор.

.point(*value*) — вмикає або вимикає двокрапки між розрядами, може приймати значення 0 або 1. Типові константи: *POINT_ON* та *POINT_OFF*. **Зверніть увагу!** Оновлення відбувається тільки після звернення до 2 розряду індикатора.

.display(*BitAddr, DispData*) — виводить дані ***DispData*** на один з чотирьох розрядів індикатора, задається змінною ***BitAddr***. ***DispDat*** може приймати значення від 0 до 15, а ***BitAddr*** — значення 0, 1, 2 або 3.

.display(*Decimal*) — виведення десяткового числа, значення ***Decimal*** можуть бути в діапазоні від –999 до 9999.

.displayByte(*BitAddr, DispData*) — виводить байт даних ***DispData*** на один з чотирьох розрядів індикатора, задається змінною ***BitAddr***. ***DispDat*** може приймати значення від 0 до 255, символ для відображення формується за принципом: 0bDpGFEDCBA — тобто послідовності з 8 бітів у двійковому представленні числа. Потрібно в двійковій послідовності поставити одиницю на місце сегмента, який повинен світитися та нуль на місце того сегмента, який повинен бути вимкнений (рис. 14.1б) та отримане число в будь якому форматі підставити замість ***DispData***.

.clearDisplay() — вимикає всі сегменти семисегментного індикатора.

Для успішного використання даної бібліотеки необхідно ввести до програми наступний код:

```
#include <TM1637.h> // Підключення бібліотеки

// Задавання пінів Arduino до яких підключено відповідні виводи модуля
#define CLK 3
#define DIO 2

TM1637 tm1637(CLK, DIO);
```

```
/*створення екземпляру об'єкту типу TM1637 і заносимо туди номери пінів, до яких підключені виводи модулю. Ім'я tm1637 може бути довільним.*/
void setup()
{
tm1637.init();// ініціалізація бібліотеки «TM1637.h»
tm1637.set(BRIGHT_TYPICAL);//встановлення типового значення яскравості
}
```

Як приклад використання описаних функцій та тестування працездатності модулю можна додати наступний код:

```
void loop() {
tm1637.point(POINT_ON);
tm1637.display(0,0);
tm1637.display(1,1);
tm1637.display(2,2);
tm1637.display(3,3);
delay(1000);
tm1637.point(POINT_OFF);
tm1637.display(1324);
delay(2000);
}
```

Також кожна бібліотека зазвичай містить приклади для полегшення роботи з ними. Приклади можна знайти в меню *Arduino IDE*: Файл/Приклади/Назва_бібліотеки.

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 14.1.

Таблиця 14.1 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	**Номінал (тип)**	**Кількість, шт**
Модуль *Catalex*	4–*Digit Display*	1
Резистори	10 кОм	2
Кнопки	–	2
Гнучкі провідники	*male-male*	4
	male-female	4
Жорсткі перемички	–	набір

2. Перевірити наявність підключеної бібліотеки *TM*1637 та при необхідності провести її підключення згідно інструкції.

3. Підключити до Arduino модуль з семисегментним індикатором: 4 провідники типу male-female з'єднати з відповідними виводами на Arduino. Показати зібрану схему викладачу.

4. Перевірити працездатність модулю виконавши програму, наведену в теоретичних відомостях.

5. Підключити до *Arduino* кнопку (не забувати про підтягувальний резистор), якщо Ви все ще сумніваєтесь, що зможете правильно підключити кнопку до *Arduino* то продемонструйте зібрану схему викладачу.

6. Розробити програму, яка буде змінювати яскравість світіння семисегментного індикатора від 0 до 7, збільшуючи значення яскравості на одиницю при натисненні на кнопку. При цьому на семисегментний індикатор повинна виводитися цифра 8 в третій десятковий розряд. Продемонструвати програму викладачу.

7. Підключити до *Arduino* ще одну кнопку. Переробити програму так, щоб по натисненню на одну кнопку змінювався розряд відображення цифри, а при натисненні на іншу цифра збільшувалася на одиницю в діапазоні від 0 до 15. Продемонструвати робочу програму викладачу

*8. Розробити програму, яка при натисненні кнопки буде по черзі вмикати в одному розряди сегменти від A до G, а в іншому виводити відповідну цьому сегменту літеру.

9. Записати до *Arduino* порожній скетч, розібрати схему та повернути макет та ЕК викладачу.

Контрольні питання

1. Як працює Ваша програма?

2. Що таке семисегментний індикатор?

3. Навіщо підключати бібліотеки до проектів та як це зробити?

ЛАБОРАТОРНА РОБОТА № 15
РОБОТА ІЗ ЗОВНІШНІМИ МОДУЛЯМИ. ДАВАЧ ТЕМПЕРАТУРИ ТА ВОЛОГОСТІ

Мета роботи: закріпити навички роботи з семисегментним індикатором. Ознайомлення з модулем температури та вологості *DHT*–11. Створення програми вимірювання вологості та температури з виведенням даних на семисегментний індикатор.

Теоретичні відомості

На ринку існує велика кількість різноманітних давачів вологості та тиску різного цінового діапазону та точності. В якості побутового давача для приміщень часто використовується давач *DHT*–11 (рис. 15.1), який має наступні параметри:

- діапазон вимірюваної температури: 0–50 °С, точність ±2 °С;
- діапазон вимірюваної вологості: 20–90% *RH*, точність 5%;
- напруга живлення: 3,5–5,5 В;
- струм: 0,3 мА під час вимірювання, 60 мкА в режимі очікування

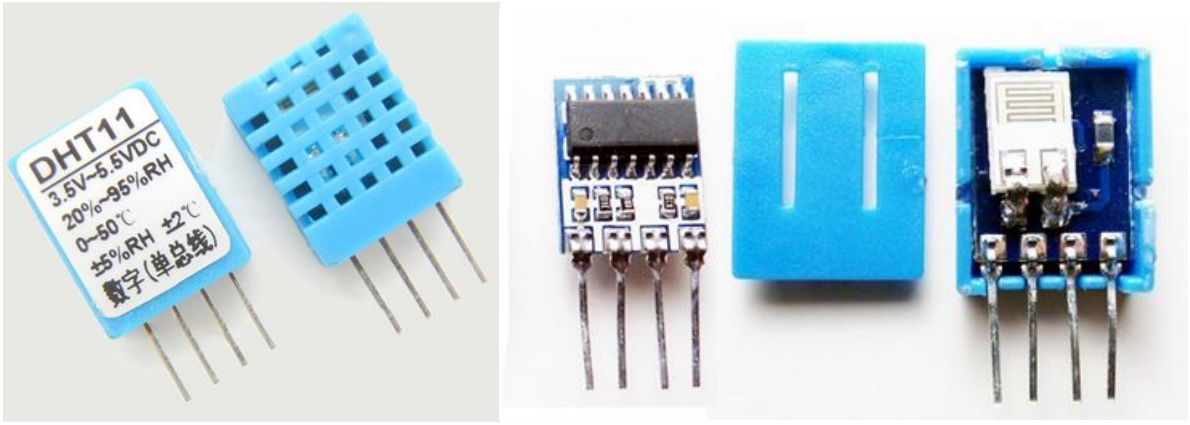


Рисунок 15.1 — Зовнішній вигляд та внутрішня будова давача *DHT*-11

Нумерація виводів йде зліва на право, якщо дивитися з боку «решітки». Вони мають наступне призначення: 1 — +5 В, 2 — *Data*, 3 — не підключений, 4 — *GND*.

При підключенні давач потребує зовнішнього підтягувального резистора на 10 кОм від +5 В до виводу *Data*.

Зверніть увагу! Давач не можна опитувати частіше одного разу в дві секунди!

Для використання давача потрібно підключити відповідну бібліотеку.

Arduino IDE містить вбудовану бібліотеку для роботи з даним давачем. Інсталювати її можна за допомогою менеджеру бібліотек, ввівши в рядку пошуку «*DHT*».

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 15.1.

Таблиця 15.1 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	**Номінал (тип)**	**Кількість, шт**
Модуль *Catalex*	4–*Digit Display*	1
Резистори	10 кОм	2
Кнопка	–	1
DHT-11	–	1
Гнучкі провідники	*male-male*	8
	male-female	4
Жорсткі перемички	–	набір

2. Перевірити за допомогою Менеджеру бібліотек чи підключена необхідна бібліотека і якщо ні, то встановити її. Відкрити демонстраційний приклад роботи з давачем: меню «Файл/Приклади», шукаємо рядок *DHT Sensor Library* та запускаємо програму *DHTtester*.

2. Уважно прочитати коментарі програми та підключити давач *DHT*-11 до *Arduino*. Продемонструвати зібрану схему викладачу.

3. Розібратися в роботі програми та перевірити її працездатність. Якщо в процесі компіляції виникає помилка, то необхідно підключити додаткову бібліотеку: *Adafruit Unified Sensor*.

4. Підключити до *Arduino* модуль семисегментного індикатора та кнопку. Продемонструвати схему викладачу.

5. Написати програму, яка буде виконувати наступний алгоритм дій: при натисненні на кнопку проводиться вимірювання температури та виведення її значення на семисегментний індикатор у вигляді: дві цифри температури, вимкнений сегмент, літера *C* на останньому сегменті. Через дві секунди проводиться вимірювання вологості та виводиться на семисегментний індикатор аналогічно до температури, але замість літери *C* літера *H* (Для цього потрібно використати функцію **.displayByte**). Ще через дві секунди дисплей вимикається. Продемонструвати готову програму викладачу.

*6. Переробити програму так, щоб команду на початок вимірювання *Arduino* зчитувало з *COM*-порту, і виведення інформації дублювало туди ж. Для цього Вам потрібні нові функції, які можна знайти в Додатку В.

7. Записати до *Arduino* порожній скетч, розібрати схему, повернути макет та ЕК викладачу.

Контрольні питання

1. Які функції виконує програма *DHTtester*?
2. Як працює Ваша програма та програма *DHTtester*?
3. Які параметри має давач *DHT*–11?
4. Як правильно підключити *DHT*–11 до *Arduino*?

ЛАБОРАТОРНА РОБОТА № 16
ПЕРЕДАВАННЯ ДАНИХ ЗА ДОПОМОГОЮ РАДІОМОДУЛІВ

Мета роботи: навчитися працювати з радіомодулем nRF24L01. Вивчити функції двостороннього передавання даних через COM-порт. Написати програму, яка реалізовує чат через радіоканал.

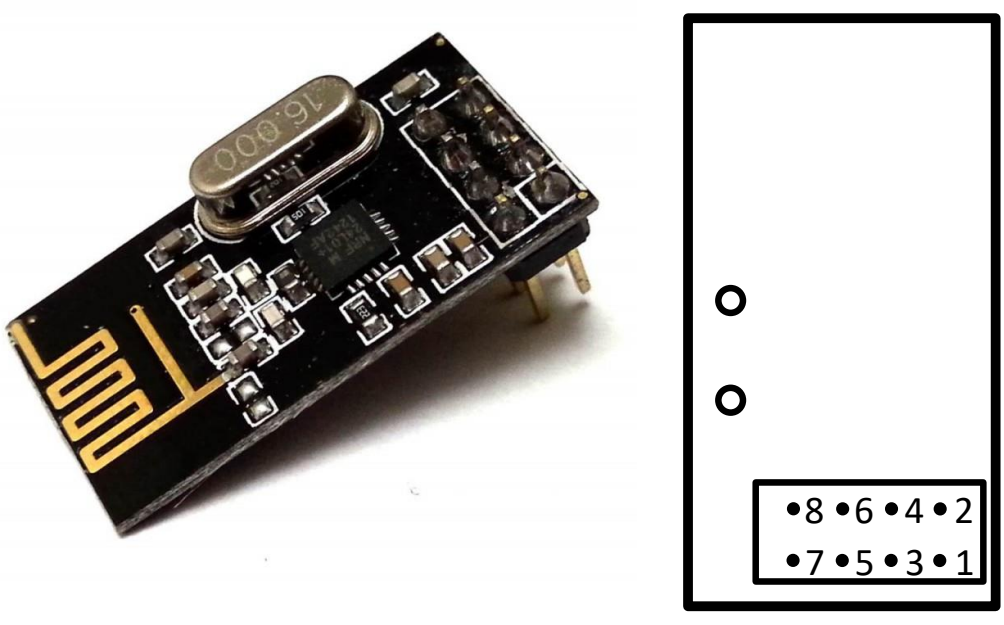
Теоретичні відомості

Бездротовий зв'язок забезпечується за допомогою спеціальних приймально-передавальних пристроїв (радіомодулів), які працюють в певному діапазоні частот. Конструкція та методи роботи приймально/передавальних пристроїв вивчаються на старших курсах. В межах даної лабораторної роботи будуть розглядатися тільки деякі характеристики радіомодулів.

Зазвичай радіомодулі використовуються для забезпечення передавання даних з різноманітних давачів чи між різними пристроями, дистанційного керування тощо.

Радіомодуль nRF24L01

Одним з популярних радіомодулів є nRF24L01+, який працює на частоті 2.4 ГГц (рис. 16.1). Вони забезпечують швидкість передавання даних до 2 Мбіт за секунду. Орієнтовна дальність передавання — до 100 метрів.



а б

Рисунок 16.1 — Радіомодуль nRF24L01+ (а) та нумерація виводів (б)

Призначення виводів та їх підключення до *Arduino* наведено в

табл. 16.1. **Зверніть увагу! Напруга живлення +3,3 В!**

Таблиця 16.1 — Призначення виводів радіомодуля

nRF24L01+		**Виводи Arduino UNO**
Номер виводу	**Призначення**	
1	*GND*	*GND*
2	*Vcc*	**+3.3 *V***
3	*CE*	9
4	*CSN*	10
5	*SCK*	13
6	*MOSI*	11
7	*MISO*	12
8	*IRQ*	–

Радіомодуль, особливо при роботі на великих потужностях, дуже чутливий до стабільності напруги живлення, тому необхідно забезпечити стабілізацію та фільтрування напруги, яка йде від *Arduino* до радіомодулю. Для цього необхідно підключити два конденсатора до живлення згідно з рис. 16.2, при чому відстань між конденсаторами та провідниками живлення до модулю та до *Arduino* повинна бути мінімальною.

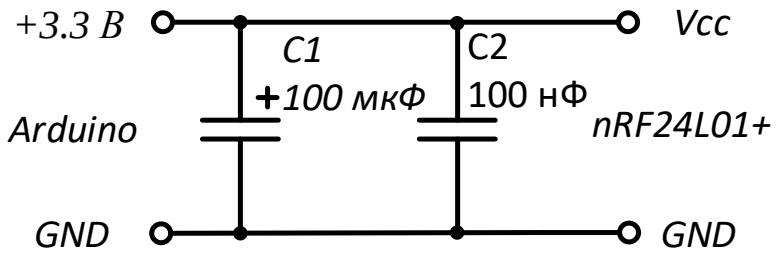


Рисунок 16.2 — Схема ввімкнення живлення радіомодулю

Функції для роботи з радіомодулем

Для роботи з радіомодулем необхідно підключити відповідну бібліотеку RF24 (рис. 16.3).

Розглянемо функції, які необхідні для роботи з радіомодулем.

.begin() — ініціалізація об’єкту;

.setDataRate(*value*) — встановлення швидкості передавання даних. Типові значення ***value***: RF24_250KBPS, RF24_1MBPS, RF24_2MBPS;

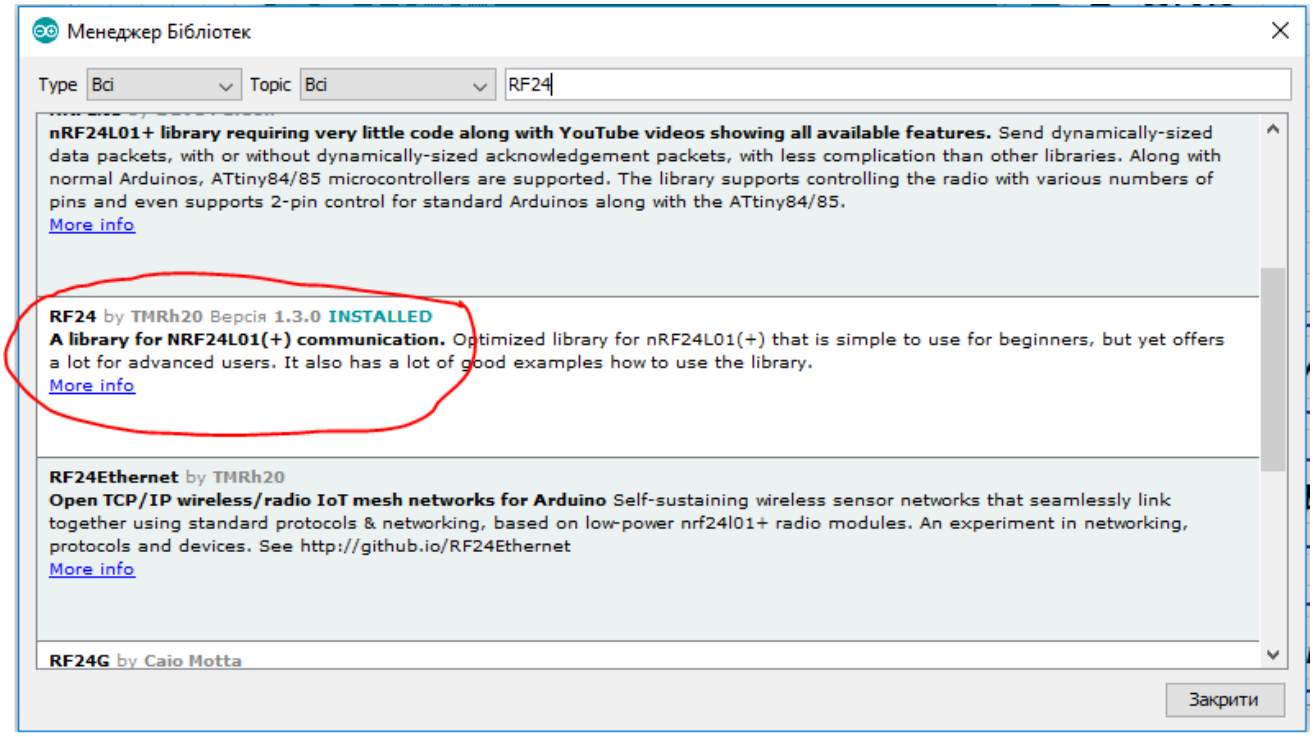


Рисунок 16.3 — Підключення необхідної бібліотеки

.setPALevel(*value*) — встановлення потужності передавача, типові значення ***value***: RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH RF24_PA_MAX, що відповідає потужності: –18dBm, –12dBm, –6dBm, 0dBm;

.setChannel(*value*) — призначення номеру каналу, ***value*** може приймати значення від 0 до 127;

.startListening() та **.stopListening()** — початок та завершення прослуховування каналу. **Забороняється під час прослуховування каналу намагатися передавати дані!**

.openWritingPipe(*CHANNEL_PIPE*) — відкриття каналу з унікальним іменем ***CHANNEL_PIPE*** на передавання даних;

.openReadingPipe(1, CHANNEL_PIPE) — відкриття каналу з унікальним іменем ***CHANNEL_PIPE*** на приймання даних;

.available() — перевірка наявності даних для зчитування;

.write(*DATA_BUFFER, len*) — передавання даних до радіоканалу з буферу ***DATA_BUFFER*** довжиною ***len***;

.read(*DATA_BUFFER, len_max*) — приймання даних з радіоканалу та

записування їх до буферу ***DATA_BUFFER*** максимальною довжиною ***len_max***;

.radio.getPayloadSize() — визначає, скільки було зчитано байт з радіоканалу.

Зверніть увагу! Для встановлення зв'язку між двома різними радіомодулями необхідно, щоб в них співпадав номер та унікальне ім'я каналу!

Робота з послідовним портом

Також для виконання лабораторної роботи знадобляться функції, для передавання та приймання даних через послідовний порт (com-порт) з класу **Serial**. Новими для Вас функціями будуть:

.write(*buf, len*) — запис до послідовного порту даних з буферу ***buf***, та довжиною ***len***;

.readBytesUntil(*character, buffer, length*) — зчитування даних з послідовного порту до тих пір, доки не зустрінеться символ ***character*** або не буде досягнуто максимальної кількості ***length***, зчитані символи записуються до буферу ***buffer***, також функція повертає кількість реально зчитаних символів.

Приклад програми

Зверніть увагу! Нижче наведено частини коду, які потрібно правильно використати при написання чату.

Більш докладно розглянемо принцип роботи з вищеперерахованих функцій на прикладі:

```
// ініціалізація необхідних бібліотек для роботи з радіомодулем та послідовною шиною SPI
#include "SPI.h"
#include "RF24.h"

#define len_max 250 // Задавання максимального розміру буферу для даних

RF24 radio(9, 10); // Створення об'єкту радіо

//Встановлюємо унікальне ім'я каналу приймання/передавання
const uint64_t CHANNEL_PIPE = 0xF0F0F0F0E1LL;

//Створюємо буфер для даних заданої довжини
byte DATA_BUFFER[len_max];

void setup()
```

```
{
  Serial.begin(115200); // Задаємо швидкість передавання даних
  radio.begin(); // Ініціалізуємо об'єкт radio
// Встановлюємо швидкість передавання
  radio.setDataRate(RF24_250KBPS);
// Встановлюємо потужність передавача
  radio.setPALevel(RF24_PA_LOW);
  // Призначаємо номер каналу
  radio.setChannel(100);
  //Починаємо слухати канал
  radio.startListening();
}
```

Вищенаведений код є обов'язковим для успішної роботи програми!

Нижче наведено фрагменти коду, який необхідно правильно розмістити в функції void loop()

```
/*** Приклад коду для передавання даних! ***/
  // Вимкнути прослуховування каналу (вимкнути приймач)
  radio.stopListening();
  // Відкрити канал з унікальним іменем CHANNEL_PIPE для передавання
  radio.openWritingPipe(CHANNEL_PIPE);
  // Сюди необхідно вставити код для зчитування даних з послідовного порту комп'ютера
  // Передати дані до радіоканалу
  radio.write(DATA_BUFFER, len);
```

```
/*** Приклад коду для приймання даних ***/
  // Відкрити канал з унікальним іменем CHANNEL_PIPE для зчитування з нього даних
  radio.openReadingPipe(1, CHANNEL_PIPE);
  // Починаємо прослуховувати канал
  radio.startListening();
  // Перевіряємо, чи вже почалося передавання даних, функція radio.available() повертає true до тих пір, доки в каналі є дані для приймання
    while (radio.available())
    {
      // Зчитати дані та записати їх до буферу DATA_BUFFER
      radio.read(DATA_BUFFER, len_max);
      int len_read = radio.getPayloadSize(); // Визначити, скільки було зчитано реальних даних
```

```
}
```

/ Приклад коду зчитування даних введених в Монітор послідовного порту на комп'ютері ***/***

```
while (Serial.available()) Serial.read(); // Очищення черги на зчитування даних з послідовного порту
while (len == 0) {
// Зчитування даних введених в послідовний порт до тих пір, доки не зустрінеться символ переведення каретки 0x0D ( Enter ) але не більше 250 символів.
len = Serial.readBytesUntil(0x0D, DATA_BUFFER, 250);
}
// Виведення введених даних з послідовного порту назад до Монітору ПП для створення ефекту чату.
Serial.print("I say: ");
Serial.write(DATA_BUFFER, len);
Serial.println();
```

/ Приклад коду для виведення в Монітор ПП даних, отриманих з радіоканалу ***/***

```
Serial.print("Answer: ");
Serial.write(DATA_BUFFER, len_read);
Serial.println();
```

Порядок виконання роботи

1. Отримати у викладача макетну плату та набір ЕК згідно табл. 16.2.

Таблиця 16.2 — Необхідні ЕК для виконання лабораторної роботи

ЕРЕ	Номінал (тип)	Кількість, шт
Модуль nRF24L01+	–	1
Конденсатори	100 мкФ	1
	100 нФ	1
Кнопка	–	1
Резистори	10 кОм	1
	680 Ом	1
Світлодіод	–	1
Гнучкі провідники	*male-male*	4
	male-female	7
Жорсткі перемички	–	набір

2. Уважно зібрати схему згідно з рис. 16.2 та табл. 16.1. Кнопку та світлодіод підключати не потрібно! Продемонструвати схему викладачу.

3. Розібратися в наведених в теоретичних відомостях частинах коду. **Номер каналу** для передавання даних обрати за розрахунком: номер комп'ютера помножена на 10. **Унікальне ім'я** каналу сформувати як: 0xF0F0F0F0**XX**LL, де **XX** номер каналу в шістнадцятковій формі!

4. Написати програму для реалізації чату між двома комп'ютерами через радіоканал. Для цього необхідно, щоб програма забезпечувала наступну послідовність дій:

1) Проведення підготовки для передавання даних у визначений канал;

2) Зчитування введених символів з Монітору ПП, при цьому програма не повинна виконуватися далі, доки не буде отримано повідомлення з комп'ютера;

3) Передавання зчитаних даних в радіоканал;

4) Переведення режиму роботи модуля в режим приймання даних;

5) Очікування відповіді від іншого модулю;

6) Зчитування даних, які приходять на приймач;

7) Виведення отриманих даних до Монітору ПП;

8) Повернення до пункту 1.

Для зручності Ви можете тестувати програму з іншою бригадою. При цьому для успішного початку діалогу один з приймачів повинен бути в режимі передавача, а інший в режимі приймача!

5. Отримати від викладача дані його каналу та провести з ним чат: Надіслати прізвища студентів бригади (Використовувати латинські літери!), отримати у відповідь математичний приклад, надіслати у відповідь результат.

*6. Переробити схему та програму так, щоб перемикання режимів приймання/передавання здійснювалося за допомогою натискання кнопки. При цьому світлодіод засвічувався в режимі передавача!

7. Записати до *Arduino* порожній скетч, розібрати схему та повернути макет і ЕК викладачу.

Контрольні питання

1. Які функції виконує Ваша програма?

2. На якій частоті працює приймач та передавач nRF24L01+?

3. Як Ви думаєте, від чого залежить максимальна дальність передавання даних через радіоканал і якими функціями Ви можете регулювати цю дальність в даному радіомодулі?

ПЕРЕЛІК ПОСИЛАНЬ

1. Ряди номіналів радіоелементів — Режим доступу: https://uk.wikipedia.org/wiki/Ряди_номіналів_радіоелементів — Назва з екрану.

2. Даташіт — Режим доступу: https://uk.wikipedia.org/wiki/Даташіт — Назва з екрану.

3. Друкована плата — Режим доступу:

https://uk.wikipedia.org/wiki/Друкована_плата — Назва з екрану.

4. Система автоматизованого проектування і розрахунку — Режим доступу: https://uk.wikipedia.org/wiki/Система_автоматизованого_проектування_і_розрахунку — Назва з екрану.

5. Посібник користувача DipTrace — Режим доступу: https://diptrace.com/books/tutorial_ua.pdf — Назва з екрану.

6. ATMEL 8-Bit Microcontroller with 4/8/16/32Kbytes in-system programmable flash. Datasheet — Режим доступу: http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf.

7. Language Reference — Режим доступу:

https://www.arduino.cc/reference/en/ — Назва з екрану.

8. Аналого-цифровий перетворювач — Режим доступу: https://uk.wikipedia.org/wiki/Аналого-цифровий_перетворювач — Назва з екрану.

9. Рівняння Стейнхарта — Харта — Режим доступу: https://uk.wikipedia.org/wiki/Рівняння_Стейнхарта_—_Харта — Назва з екрану.

10. Arduino Uno Pinout Diagram — Режим доступу: https://makezine.com/2013/02/06/arduino-uno-pinout-diagram/ — Назва з екрану.

ДОДАТОК А.
КОЛЬОРОВЕ МАРКУВАННЯ РЕЗИСТОРІВ

Кольорове маркування резисторів наноситься у вигляді 3, 4, 5 або 6 кольорових кілець. Кільця повинні бути зсунуті до одного з країв або ширина кільця першого знаку повинна бути в два рази більше інших, що на практиці витримується не завжди. Значення кольорів наведено в табл. Б.1. Під час виконання лабораторних робіт будуть використовуватися резистори з 4 та 5 кільцями. В першому випадку два перших кільця — значення, третє кільце — множник, останнє — допуск. В другому випадку три перші кільця — значення, четверте кільце — множник, останнє — допуск.

Таблиця Б.1 – Таблиця кольорового кодування опору резисторів

Колір	Значення		Множник	Допустиме відхилення ± %	Темп.коеф.опору $\pm 10^{-6}/K$
	1 кільце	2 кільце	3 кільце	4 кільце	Останнє кільце
відсутнє				20	
сріблястий			0,01 Ω	10	
золотистий			0,1 Ω	5	
чорний	0	0	x 1 Ω	20	200
коричневий	1	1	x 10 Ω	1	100
червоний	2	2	x 100 Ω	2	50
оранжевий	3	3	x 1 kΩ	3	15
жовтий	4	4	x 10 kΩ	0,1	25
зелений	5	5	x 100 kΩ	0,5	
голубий	6	6	x 1 MΩ	0,25	10
фіолетовий	7	7	x 10 MΩ	0,1	5
сірий	8	8		0,05	1
білий	9	9			

Для декодування опору резисторів можна використовувати онлайн-додатки, наприклад,

http://radionavt.com.ua/uk/library/resistors-color-code-calculator

ДОДАТОК Б. ХАРАКТЕРИСТИКИ ARDUINO UNO

Технічні характеристики

Мікроконтролер	ATmega328
Робоча напруга	5 В
Вхідна напруга	7–12 В
Вхідна напруга (гранична)	6–20 В
Цифрові входи/виходи	14 (6 з них можна використати для ШІМ)
Аналогові входи	6
Струм через входи/виходи	40 мА
Струм для виводу 3,3В	50 мА
Flash пам’ять	32 кБ
Оперативна пам’ять	2 кБ
EEPROM	1 кБ
Тактова частота	16 МГц

Призначення виводів

На рис. А.1 умовно показано призначення виводів апаратної платформи Arduino UNO та безпосередньо мікроконтролера ATmega328. Зверніть увагу, що більшість виводів мікроконтролера (і, відповідно, платформи Arduino) можуть виконувати декілька функцій в залежності від початкового їх конфігурування чи використання відповідних функцій. Для уникнення проблем, пов’язаних з особливостями програмування контролера та електричної схеми платформи Arduino, не рекомендується використовувати: вивід Reset у якості виводу введення\виведення; виводів 0 та 1 платформи, якщо програма передбачає передавання даних через COM-порт (також останні варто підключати до низькоомного навантаження тільки після запису програми до мікроконтролера); вивід 13 підключено до світлодіода на платі платформи, тому використовувати його з вбудованим підтягувальним резистором не можна.

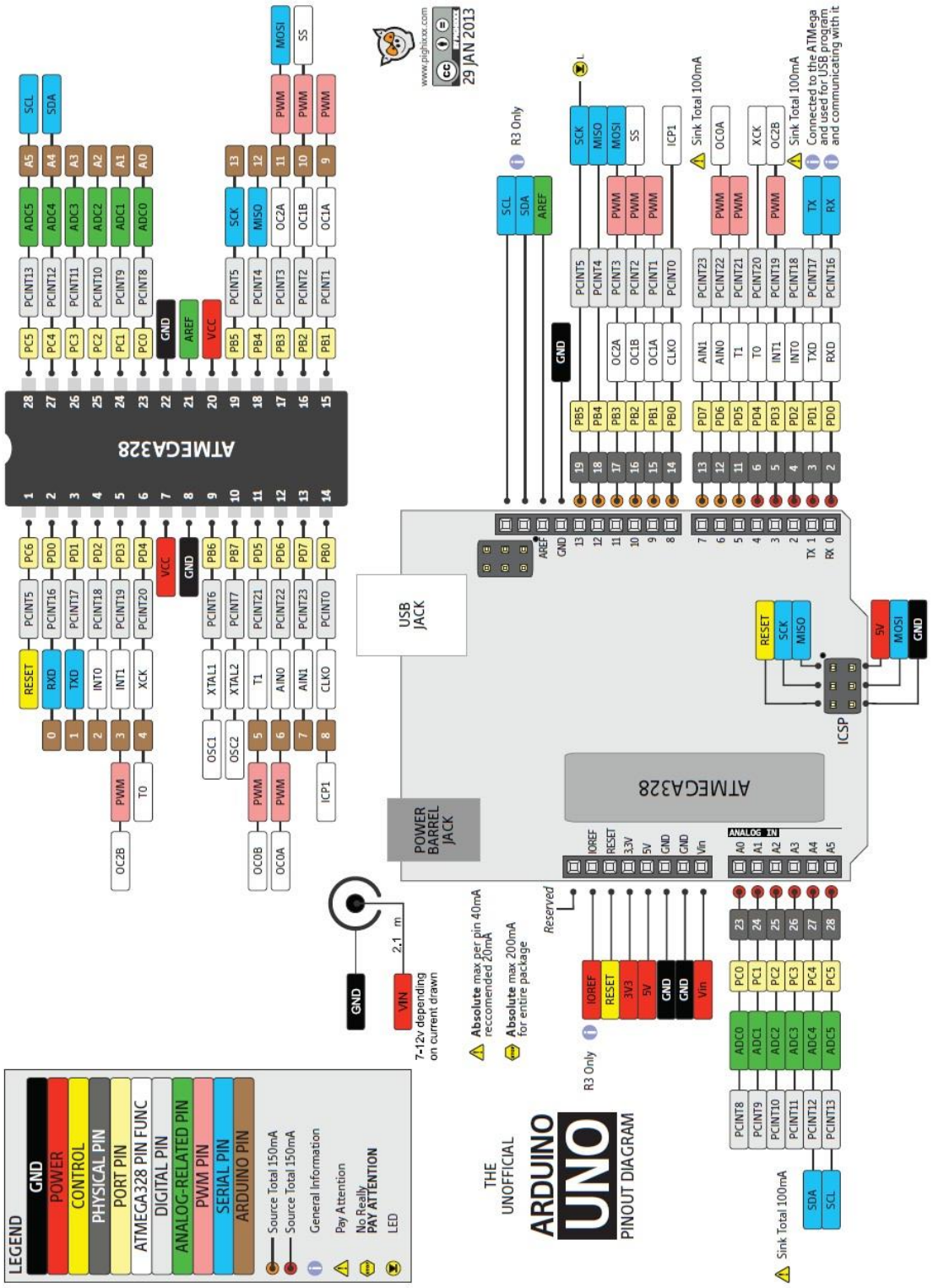


Рисунок А. 1 — Діаграма виводів Arduino [10]

ДОДАТОК В. ДОВІДНИК ПРОГРАМУВАННЯ ARDUINO

В додатку наведені тільки основні функції, директиви та дані, які використовуються під час виконання завдань з цього навчального посібника. Повний перелік можна знайти на офіційному сайті: https://www.arduino.cc/reference/en/

Напруги для логічного нуля та одиниці наведені для випадку Arduino UNO з 5 В живленням.

Синтаксис мови

Враховуючи, що для програмування *Arduino* використовується С-подібна мова програмування, то і синтаксис у них однаковий. До основних елементів синтаксису можна віднести:

«;» — крапка з комою ставиться в кінці оператора.

«{}» — фігурні дужки виокремлюють тіло функції, циклу, оператора умови. Кожній відкритій фігурній дужці повинна бути відповідна закриваюча фігурна дужка.

«//» — коментар. Текст, записаний після подвійного слешу ігнорується компілятором.

«/*» та «*/» — відкриваючий та закриваючий символ багаторядкового коментаря.

Норми програмування вимагають гарного документування кода з використанням коментарів. Таким чином інший програміст або і автор програми зможе простіше розібратися в написаному коді.

#define *constantName value*

Це директива, яка створює константу з ім'ям заданим в полі ***constantName*** та присвоює їй значення ***value.*** ЇЇ доречно використовувати в тих випадках, коли потрібно вказати значення, які можуть змінюватися в процесі налагодження Вашого пристрою. Наприклад, номера виводів контролера, кількість світлодіодів тощо. Значення директиви не може змінитися в процесі виконання програми! Адже в процесі компіляції всі ***constantName*** просто замінюються на ***value***.

#include <*name*>

Це директива, яка використовується для включення сторонніх бібліотек до вашого скетчу, де ***name*** — ім'я бібліотеки. Використання подібної

директиви надає доступ до великої кількості стандартних бібліотек C (бібліотекою називають групи попередньо написаних функцій), і бібліотек написаних спеціально для *Arduino*.

Типи даних

Наведемо основні типи даних:

void

Використовується лише при оголошенні функцій та вказує на те, що функція не повертає ніякого значення. Наприклад:

```
void setup()
{
}
```

boolean

Змінні типу boolean можуть набувати одне з двох значень: ***true*** (логічна одиниця) або ***false*** (логічний нуль), при цьому вона займає в пам'яті один байт.

Приклад:

```
boolean button = false;
```

char

Тип даних, який займає у пам'яті 1 байт і зберігає символьне значення. Хоча фактично в пам'яті зберігається не сам символ, а його числовий код. Прикладом такого кодування є таблиця ASCII символів. Наприклад нижченаведені записи є еквівалентними:

```
char CharN = 'A';
char CharN = 65;
```

byte

Змінна типу byte зберігає 8-бітове без знакове число, тобто числі від 0 до 255.

int

Це основний тип даних, який призначений для зберігання цілих чисел. В *Arduino Uno* розмірність int складає 2 байти, відповідно змінна може містити числа в діапазоні від −32768 до 32767.

float

Змінні цього типу призначені для зберігання чисел з рухомою комою. Займають в пам'яті 4 байти та можуть містити числа в діапазоні від −3.4028235E+38 до 3.4028235E+38 проте з обмеженою точністю. Варто розуміти, що використання змінних типу float значно сповільнює виконання програми, так як потребують більше ресурсів мікроконтролера на їх обрахунок. Варто уникати використання цього типу змінних в чутливих до часу виконання частинах програми.

Оператори

Арифметичні оператори

При програмуванні Arduino використовуються типові арифметичні оператори: «=» — присвоєння, «+» —додавання, «−» — віднімання, «*» — множення; «/» — ділення та «%» — залишок від ділення.

Використовуючи арифметичні оператори варто звертати увагу на тип змінних до яких вони застосовуються. Адже, якщо поділити число 5 на 2, то по замовчуванню розрахунок буде проведений для типу int і результат складе 2, а не очікувані 2,5. Також, якщо, наприклад, до змінної int, яка містить число 32767 додати 1, то в результаті буде число −32768 (від'ємне).

При використанні оператора «%» отримуємо залишок від ділення, тобто, для виразу: 19%5 результат буде рівний 4. Цей оператор гарно підходить для обмеження величини змінної, так наприклад, якщо вираз number = (number+1)%10; помістити в безкінечний цикл, то змінна number буде змінюватися в діапазоні від 0 до 9.

Також можна використовувати подвійні комбінації арифметичних операторів, так наприклад «++» (інкремент) — збільшує змінну на одиницю, «−−» (декремент) — зменшує змінну на одиницю. Також їх можна поєднувати із оператором присвоєння, наприклад:

```
x += y;   // відповідає виразу x = x + y;
x −= y;   // відповідає виразу x = x − y;
x *= y;   // відповідає виразу x = x * y;
x /= y;   // відповідає виразу x = x / y;
```

Оператори порівняння

При програмуванні Arduino використовуються типові оператори порівняння: «==» — рівно, «!=» — не рівно, «<» — менше, «>» — більше, «<=»

— менше або рівно, «>=» — більше або рівно. Оператори порівняння застосовуються разом з операторами умовного переходу.

Логічні оператори

Також з операторами умови часто використовують логічні оператори:

&& (І) — виконує логічну функцію «І», тобто повертає значення істини, коли обидва операнда істині (наприклад a = b && c, a прийме значення true тільки в тому випадку, якщо b і c також будуть істиною);

|| (АБО) — виконує логічну функцію «АБО», тобто повертає значення істини, коли хоча б один з операндів є істинним;

! (НІ) — виконує логічну функцію заперечення, тобто перетворює істину на хибу або навпаки.

Бітові оператори

Повноцінний опис бітових операторів можна знайти, наприклад, у Вікіпедії: https://uk.wikipedia.org/wiki/Бітові_операції

Оператори керування

Наведемо основні оператори умовних переходів та циклів.

Цикл for()

for (*initialization*; *condition*; *increment*) {функції, які виконуються в циклі}

Ця конструкція дозволяє створити цикл (тобто повторювати вказані функції певну кількість разів, функції, які повторюються повинні розміщуватися в фігурних дужках {}). ***Initialization*** виконується один раз і використовується для встановлення початкового значення змінної. ***Condition*** — тут вказується умова не виходу з циклу. ***Increment*** — вказується операція над змінною з поля ***Initialization*** при кожній ітерації, наприклад її збільшення чи зменшення. Наприклад:

```
for (int i = 2; i<=6; i++){
   digitalWrite (i, HIGH);
  }
```

Словами це можна описати наступним чином: змінна *«i»* дорівнює 2, якщо змінна *«i»* менше або рівня 6, то виконується функція *digitalWrite (i, HIGH)*, після цього до *«i»* додається 1 (саме це означає запис *i*++), знову порівнюємо і з 6, і якщо умова все ще виконується, то знову повторюємо функцію *digitalWrite (i, HIGH)*. Тому в нашому випадку мікроконтролер в

результаті виконає наступні дії:

```
digitalWrite (2, HIGH)
digitalWrite (3, HIGH)
digitalWrite (4, HIGH)
digitalWrite (5, HIGH)
digitalWrite (6, HIGH)
```

Тобто 5 разів викличе функцію *digitalWrite* з різним значенням *«i»*

Оператори умовного переходу if()/else

```
if (умова)
{
// частина коду, яка повинна виконуватися, при виконанні умови оператора if
}
else
{
// частина коду, яка повинна виконуватися, якщо умова оператора if не виконується
}
```

В якості умови можна використати будь-який з операторів порівняння та комбінувати їх за допомогою логічних операторів.

Також даний оператор підтримує використання додаткової умови ***else if***

```
if (x > 5){
  // виконується, якщо змінна x > 5
}else if (x < 3){
  // виконуються, якщо x < 3, тобто перша умова хиба, а друга істина
}else{
  // виконуються, якщо x буде лежати в діапазоні від 3 до 5 включно, тобто перша та друга умови хибні.
}
```

Оператор умовного переходу ***if*** може використовуватися і без ***else***, в такому випадку, якщо умова оператора не виконується, то програма переходить до виконання наступної функції після конструкції ***if***.

Оператор switch()/case

Цей оператор дозволяє просто організувати множинний вибір коду, який буде виконуватися в залежності від різних умов. Синтаксис:

```
 switch (mode){
  case 0:
// код для першого випадку
```

```
    break;
   case 1:
// код для другого випадку
    break;
default:
// код, який виконується, якщо не спрацював жоден з case
```

Програма порівнює значення змінної ***mode*** із значеннями, які йдуть після ***case*** і, відповідно, виконується та частина коду, які йде після ***case*** з потрібним значенням. Ключове слово ***break*** потрібно для того, щоб завершити виконання блоку потрібного коду і вийти з оператора ***case***. Його потрібно вставляти після кожного ***case***, в іншому випадку буде виконана не тільки та частина коду, яка йде у потрібному ***case***, а й та, яка йде у наступному. Параметр default не є обов'язковим, при його відсутності, якщо не спрацював жоден з ***case*** то програма переходить до виконання коду за межами ***switch***.

Оператор циклу While()

Приклад ще одного циклу, який буде виконуватися неперервно, доки виконується умова:

```
while(умова){
 // тіло цикла
}
```

На відміну від циклу *for* цей оператор використовується тоді, коли кількість повторень нам не відома, часто використовують для очікування настання певної умови, наприклад, натиснення кнопки, появи даних в буфері COM-порту тощо.

Основні функції

setup()

Обов'язкова функція програми для *Arduino*. Дана функція запускається один раз при подачі живлення на мікроконтролер. Використовується для конфігурації режимів роботи виводів, ініціалізації бібліотек та виконання інших функцій, які повинні відпрацювати лише один раз. Приклад:

```
int Led_Pin = 3;
void setup()
{
 Serial.begin(9600);
 pinMode(Led_Pin, OUTPUT);
}
```

loop()

Друга обов'язкова функція програми для *Arduino*, вона викликається після виконання функції *setup()*, і виконується циклічно весь час, поки присутнє живлення на мікроконтролері. Тобто це безкінечний цикл, в середині якого необхідно писати функції, які повинен виконувати мікроконтролер.

Цифрове введення/виведення даних

pinMode()

Функція **pinMode*(pin, mode)*** конфігурує вивід з номером, який записаний на місці *pin* мікроконтролера на виконання певної функції. Є три варіанти значення для *mode*:

– *INPUT* (або ставлять 0) — зчитування інформації з виводу, власне даний режим встановлюється по замовчуванню, тому викликати дану функцію з цим значенням в більшості випадків немає потреби;

– *OUTPUT* (або «1») — виведення інформації;

– *INPUT_PULLUP* — увімкнення внутрішнього підтягувального резистора та зчитування інформації з виводу. При такій конфігурації з виводу по замовчуванню буде зчитуватися значення логічної одиниці, так як підтягувальний резистор підключається до +5 В.

digitalWrite()

Функція **digitalWrite(*pin, value*)** виводить цифрове значення ***value*** на вивід з номером ***pin***.

Може виводити два значення: логічний нуль «0» або ***LOW*** та логічну одиницю «1» або ***HIGH*** (Наприклад, записи digitalWrite*(led_pin_2, LOW)* та digitalWrite*(led_pin_2, 0)* ідентичні та виконують одне й теж, а саме: виведення на вивід *led_pin_2* логічного нуля).

При виконанні функції на виводі за номером ***pin*** мікроконтролера встановлюється напруга +5 В, якщо ***value*** відповідає логічній одиниці, та 0 В, якщо логічному нулю.

digitalRead()

Функція ***val* = digitalRead(*pin*)** присвоює змінній ***val*** значення логічної одиниці (***true***), якщо до виводу за номером ***pin*** мікроконтролера прикладено позитивну напругу від 3 до 5 В, та значення логічного нуля (***false***), якщо прикладено напругу від 0 до 1,5 В.

Аналогове введення/виведення

analogRead()

Для зчитування даних з аналогового входу використовується функція **analogRead(*pin*)**, де ***pin*** — номер виводу, з якого необхідно зчитати значення напруги.

Апаратна платформа *Arduino UNO* має 6 аналогових входів, які позначаються A0 – A5, сигнали з них подаються на 10-розрядний АЦП. Опорна напруга АЦП може змінюватися, проте за замовчуванням використовується $U_{оп} = 5\ B$.

Для вибору виводу, з якого будуть зчитані дані можна використати позначення 0 – 5 або ж A0 – A5. Функція повертає ціле число в діапазоні від 0 до 1023, яке прямо пропорційна прикладеній до аналогового входу напруги.

Для роботи цієї функції використовується АЦП, більш докладно про його роботу написано в теоретичних відомостях до Лабораторної роботи №9.

analogWrite()

Для встановлення певного значення напруги на відповідному виводі використовується функція **analogWrite(*pin, value*)**, де ***pin*** — номер виводу, на якому потрібно встановити значення напруги, ***value*** — число від 0 до 255, яке пропорційне значенню напруги. Дана функція працює тільки з тими цифровими виводами, які мають позначку у вигляді «~», це виводи 3, 5, 6, 9, 10, 11 для Arduino Uno.

Наприклад, для встановлення на 11 виводі напруги 3 В, потрібно розрахувати значення ***value***: $3 \cdot 255 / 5 = 153$ та викликати функцію analogWrite(11, 153). На 11 виводі буде постійно (до наступного звернення до цього виводу) генерувати послідовність прямокутних імпульсів з частотою 500 Гц та коефіцієнтом заповнення 0,6, що відповідає середньому значенню напруги в 3 В.

Для роботи цієї функції використовується Широтно-імпульсний модулятор, більш докладно про його роботу написано в теоретичних відомостях до Лабораторної роботи №11.

Функції для роботи з часом

delay()

Функція **delay(*ms*)** призупиняє виконання програми на час ***ms*** (значення в мілісекундах).

delayMicroseconds()

Функція **delayMicroseconds(*us*)** призупиняє виконання програми на час ***us*** в мікросекундах. Функція коректно працює доки значення затримки не перевищує 16383 мікросекунд. Якщо потрібно більше, то використовуйте функцію **delay()**

millis()

Функція **millis()** повертає кількість мікросекунд, які пройшли з моменту старту програми Arduino. Максимальний термін — орієнтовно 50 днів. Формат повернення даних: unsigned long. В основному використовується для керування часом виконання певних частин коду, як альтернатива функції **delay()**

Інші функції

tone()

Для генерування сигналу у вигляді прямокутних імпульсів різної частоти (з коефіцієнтом заповнення 50%) використовуються внутрішній таймер/лічильник. Для виведення сигналу використовується функція **tone()**.

Використовується два варіанти синтаксису: **tone(*pin, frequency*)** та **tone(*pin, frequency, duration*)**, де ***pin*** — номер виводу, на який необхідно подати сигнал, ***frequency*** — частота сигналу в Герцах, ***duration*** — тривалість сигналу в мілісекундах. Якщо тривалість звучання не вказана, то генерування сигналу буде відбуватися до тих пір, доки не буде вимкнене командою **noTone(*pin*)** або змінене командою **tone()**.

За допомогою команди **tone()** не можна одночасно подавати сигнали на різні виводи. Для виведення сигналу на інший вивід потрібно перш за все вимкнути сигнал на поточному виводі за допомогою команди **noTone(*pin*).**

map()

Функція **map(*value, fromLow, fromHigh, toLow, toHigh*)** пропорційно перетворює значення змінної ***value*** з діапазону від ***fromLow*** (нижня межа поточного діапазону чисел до якого належить ***value***) до ***fromHigh*** (верхня межа поточного діапазону чисел до якого належить ***value***) в діапазон від

toLow (нижня межа нового діапазону) до ***toHigh*** (верхня межа нового діапазону). Функція працює тільки з цілими числами, тому якщо в процесі перетворення виникають дробові числа, то функція повертає тільки цілу частину.

Бібліотеки

Клас Serial

Основним завданням класу *Serial* є забезпечення зв'язку плати *Arduino* з комп'ютером чи іншими пристроями. Для зв'язку з комп'ютером використовується універсальний послідовний порт *UART* (англ. *universal asynchronous receiver/transmitter* — універсальний асинхронний приймач/передавач). Він пов'язаний з виводами 0 (*RX*) та 1 (*TX*) та забезпечує зв'язок *Arduino* з комп'ютером через *USB*. Власне тому, якщо Ви плануєте використовувати передавання даних до комп'ютера, то цифрові виводи 0 та 1 повинні бути вільними.

Основні функціє даного класу:

Serial.begin() — функція, яка задає швидкість передавання даних по послідовному інтерфейсу в бітах за секунду. Синтаксис **Serial.begin(*speed*)**, де ***speed*** традиційно задається з ряду 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 або 115200. Дана функція є обов'язковою, якщо заплановане передавання даних через послідовний інтерфейс. Записується один раз в функції **void setup ()**.

Serial.print() та **Serial.println()** — виводить через послідовний порт інформацію у вигляді, зрозумілому для людини. Функція **Serial.println()** додатково переводить курсор на початок наступного рядка. Наприклад, команда **Serial.print(**"Hello world."**)** виведе напис «Hello world.».

За допомогою команди **Serial.print** можна виводити дані різного типу: текстові, числові, значення змінних тощо. На відміну від мови C/C++ не обов'язково вказувати формат виведення даних, проте, якщо Вам потрібно отримати специфічний формат, то вказати його можна. Крім того, функція **Serial.print()** та **Serial.println()** повертають кількість виведених байт.

Serial.write(*buffer, length*) — записує до послідовного порту дані з буферу ***buffer***, та довжиною ***length*** (не є обов'язковим).

Serial.available() — функція кількість байт, які доступні для зчитування з буферу послідовного порту. Тобто такі байти, які вже були прийняті і зберігаються в прийомному буфері, який може накопичити максимум 64

байти. Типове використання — для визначення чи є дані в послідовному порті, щоб виконати команду їх зчитування.

Serial.read() — зчитує один байт даних, які надходять з послідовного порту.

Serial.readBytes(*buffer, length*) — зчитує задану в ***length*** кількість символів, які надходять з послідовного порту і записує їх в буфер ***buffer***. Повертає кількість зчитаних байт.

Serial.readBytesUntil(*character, buffer, length*) — зчитування даних з послідовного порту до тих пір, доки не зустрінеться символ ***character*** або не буде досягнуто максимальної кількості ***length***, зчитані символи записуються до буферу ***buffer***, також функція повертає кількість реально зчитаних символів.

Приклад програми з вищенаведеними функціями:

```
#define len_max 250 // Задавання максимального розміру буферу для даних

//Створюємо буфер для даних заданої довжини
byte DATA_BUFFER[len_max];

void setup()
{
  Serial.begin(115200); // Задаємо швидкість передавання даних
}

void loop(){
if (Serial.available() > 0){ //перевіряємо, чи є дані в послідовному порті
    while (len == 0) {
      // Зчитування даних введених в послідовний порт до тих пір, доки не зустрінеться символ переведення каретки 0x0D ( Enter ) але не більше 250 символів та записування їх в буфер.
       len = Serial.readBytesUntil(0x0D, DATA_BUFFER, 250);
    }

    // Виведення введених даних з послідовного порту назад до Монітору ПП для створення ефекту чату.
    Serial.print("I say: ");
    Serial.write(DATA_BUFFER, len);
    Serial.println();
  }
}
```

Бібліотека RF24

Типові функції бібліотеки для роботи з радіомодулями nRF24L01+

.begin() — ініціалізація об’єкту;

.setDataRate(*value*) — встановлення швидкості передавання даних. Типові значення ***value***: RF24_250KBPS, RF24_1MBPS, RF24_2MBPS;

.setPALevel(*value*) — встановлення потужності передавача, типові значення ***value***: RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH RF24_PA_MAX, що відповідає потужності: –18dBm, –12dBm, –6dBm, 0dBm;

.setChannel(*value*) — призначення номеру каналу, ***value*** може приймати значення від 0 до 127;

.startListening() та **.stopListening()** — початок та завершення прослуховування каналу. **Забороняється під час прослуховування каналу намагатися передавати дані!**

.openWritingPipe(*CHANNEL_PIPE*) — відкриття каналу з унікальним іменем ***CHANNEL_PIPE*** на передавання даних;

.openReadingPipe(1, CHANNEL_PIPE) — відкриття каналу з унікальним іменем ***CHANNEL_PIPE*** на приймання даних;

.available() — перевірка наявності даних для зчитування;

.write(*DATA_BUFFER, len*) — передавання даних до радіоканалу з буферу ***DATA_BUFFER*** довжиною ***len***;

.read(*DATA_BUFFER, len_max*) — приймання даних з радіоканалу та записування їх до буферу ***DATA_BUFFER*** максимальною довжиною ***len_max***;

.radio.getPayloadSize() — визначає, скільки було зчитано байт з радіоканалу.

Приклад використання:

```
#include "SPI.h"
#include "RF24.h"

#define len_max 250 // Задавання максимального розміру буферу для даних

RF24 radio(9, 10); // Створення об’єкту радіо
```

```
//Встановлюємо унікальне ім'я каналу приймання/передавання
const uint64_t CHANNEL_PIPE = 0xF0F0F0F0E1LL;

//Створюємо буфер для даних заданої довжини
byte DATA_BUFFER[len_max];

void setup()
{
  Serial.begin(115200); // Задаємо швидкість передавання даних
  radio.begin(); // Ініціалізуємо об'єкт radio
// Встановлюємо швидкість передавання
  radio.setDataRate(RF24_250KBPS);
// Встановлюємо потужність передавача
  radio.setPALevel(RF24_PA_LOW);
  // Призначаємо номер каналу
  radio.setChannel(100);
  //Починаємо слухати канал
  radio.startListening();
}

void loop(){
// Відкрити канал з унікальним іменем CHANNEL_PIPE для зчитування з нього даних
    radio.openReadingPipe(1, CHANNEL_PIPE);
    // Починаємо прослуховувати канал
    radio.startListening();
    // Перевіряємо, чи вже почалося передавання даних, функція radio.available() повертає true до тих пір, доки в каналі є дані для приймання
      while (radio.available())
      {
        // Зчитати дані та записати їх до буферу DATA_BUFFER
        radio.read(DATA_BUFFER, len_max);
        int len_read = radio.getPayloadSize(); // Визначити, скільки було зчитано реальних даних
      }
}
```

Бібліотека TM1637

Огляд функцій бібліотеки для роботи з модулем TM1637

.init() — ініціалізація підключеної бібліотеки;

.set(*bright*) — встановлення яскравості світіння індикатора. Змінна

bright може приймати значення від 0 до 7, типовими значеннями (вбудованими константами) є *BRIGHT_DARKEST*, яка рівна 0, *BRIGHT_TYPICAL* — 2 та *BRIGHTEST* — 7.

.point(*value*) — вмикає або вимикає двокрапки між розрядами, може приймати значення 0 або 1. Типові константи: *POINT_ON* та *POINT_OFF*. Оновлення відбувається тільки після звернення до 2 розряду індикатора.

.display(*BitAddr, DispData*) — виводить дані ***DispData*** на один з чотирьох розрядів індикатора, задається змінною ***BitAddr***. ***DispDat*** може приймати значення від 0 до 15, а ***BitAddr*** — значення 0, 1, 2 або 3.

.display(*Decimal*) — виведення десяткового числа, значення ***Decimal*** можуть бути в діапазоні від –999 до 9999.

.displayByte(*BitAddr, DispData*) — виводить байт даних ***DispData*** на один з чотирьох розрядів індикатора, задається змінною ***BitAddr***. ***DispDat*** може приймати значення від 0 до 255, символ для відображення формується за принципом: 0bDpGFEDCBA — тобто послідовності з 8 бітів у двійковому представленні числа.

.clearDisplay() — вимикає всі сегменти семисегментного індикатора.

Приклад використання:

```
#include <TM1637.h> // Підключення бібліотеки

// Задавання виводів Arduino до яких підключено модуль
#define CLK 3
#define DIO 2

TM1637 tm1637(CLK, DIO);
/*створення екземпляру об’єкту типу TM1637. Ім’я tm1637 може бути довільним.*/
void setup()
{
tm1637.init();// ініціалізація бібліотеки «TM1637.h»
tm1637.set(BRIGHT_TYPICAL);//встановлення значення яскравості
}
void loop() {
tm1637.point(POINT_ON);
tm1637.display(0,0);
```

```
tm1637.display(1,1);
tm1637.display(2,2);
tm1637.display(3,3);
delay(1000);
tm1637.point(POINT_OFF);
tm1637.display(1324);
delay(2000);
}
```

ДОДАТОК Г. ПРИКЛАД ПРОГРАМИ

Нижченаведений код та музика розроблена студентом радіотехнічного факультету КПІ ім. Ігоря Сікорського Красницьким Микитою Олександровичем.

```
#define sound   2  // Вивід п'єзоелементу
#define mus_var 1  // Варіант композиції(1-4)

void Melody();       // Прототип функції відтворення мелодії
void note(int, int); // Прототип функції відтворення ноти


/* Темп для кожної композиції і окремі значення для пауз
*  Чим більше число t - тим довші будуть паузи та звуки.
*/
#define t1   180    // Temp for music 1
#define t2   600    // Temp for music 2
#define t3   90     // Temp for music 3
#define t32  300    // Особлива пауза, яку важко розрахувати
#define t4s  180    // Temp for music 4
#define t4ws 720    // Temp for music 4

// Визначення Нот в Герцах. Перша буква - нота, Цифра - октава, приставка pp - Дієз
#define C3    131   // До 3-ї октави
#define Cpp3  139   // До дієз (#, ++)
#define D3    147   // Ре
#define Dpp3  156   // Ре дієз
#define E3    165   // Мі
#define F3    175   // Фа
#define Fpp3  185   // Фа дієз
#define G3    196   // Соль
#define Gpp3  208   // Соль дієз
#define LA3   220   // Ля
#define LApp3 233   // Ля дієз
#define H3    247   // Сі
#define C4    262
#define Cpp4  277
#define D4    294
#define Dpp4  311
#define E4    330
#define F4    349
#define Fpp4  370
```

```
#define G4     392
#define Gpp4   415
#define LA4    440
#define LApp4  466
#define H4     494
#define C5     523
#define Cpp5   554
#define D5     587
#define Dpp5   622
#define E5     659
#define F5     698
#define Fpp5   740
#define G5     784
#define Gpp5   831
#define LA5    880
#define LApp5  932
#define H5     988
#define C6     1047
#define Cpp6   1109
#define D6     1175
#define Dpp6   1245
#define E6     1319
#define F6     1397
#define Fpp6   1480
#define G6     1568
#define Gpp6   1661
#define LA6    1760
#define LApp6  1865
#define H6     1976
#define C7     2093
#define Cpp7   2217
#define D7     2349
#define Dpp7   2489
#define E7     2637
#define F7     2794
#define Fpp7   2960
#define G7     3136
#define Gpp7   3322
#define LA7    3520
#define LApp7  3729
#define H7     3951
```

//Масив, який зберігає порядок нот, за ним йде масив, який зберігає тривалість звучання кожної ноти.

//Якщо замість ноти стоїть 0, то замість звуку буде пауза відповідної тривалості

int tone_mus1[] = {Dpp6, D6, C6, Dpp6, D6, C6, Dpp6, D6, C6, D6, LApp5, G5, Dpp6, D6,
C6, Dpp6, D6, C6, D6, Dpp6, F6, G6, D6, Dpp6, F6, G6, D6, Dpp6, 0,
Dpp6, D6, C6, Dpp6, D6, C6, Dpp6, D6, C6, D6, LApp5, G5, C6, LApp5, Gpp5, C6, LApp5,
Gpp5, G5, F5, Dpp5, G5, Dpp5, G5, D6,
D5, G5, D6, Dpp6, D6, C6, G6, F6, Dpp6, D6, D5, G5, D6, Dpp6, D6, C6, G6, F6, Dpp6,
D6, D5, G5, D6, Dpp6, D6, C6,
Gpp6, G6, F6, G6, G6, F6, Dpp6, D6, C6, LApp5, C6, D6, Dpp6, D6, C6, LApp5, Gpp5,
LApp5, C6, D6, Dpp6, D6, C6, D6, Dpp6, F6, G6, Dpp6, G6, Dpp6,
D6, C6, LApp5, C6, 0, C6, LApp5, G5, Dpp5, 0, LApp5, LA5, G5, F5, LA5, G5, F5, 0, G5, F5,
G5};
*int temp_mus1[] = {t1, t1, 4.5*t1, t1, t1, t1, t1, t1, t1, 4.5*t1, t1, 6*t1, t1, t1, 4.5*t1, t1, t1, t1,*
*t1, t1, t1, 3*t1, t1, t1, t1, 3*t1, t1, 3*t1, t1/18,*
*t1, t1, 4.5*t1, t1, t1, t1, t1, t1, t1, 4.5*t1, t1, 6*t1, t1, t1, 4.5*t1, t1, t1, t1, t1, t1, t1, 3*t1, t1,*
*2*t1, 3*t1,*
*t1, t1, t1, 1.5*t1, t1/1.425, t1/1.425, 1.5*t1, t1/1.425, t1/1.425, 3*t1, t1, t1, t1, 1.5*t1,*
t1/1.425, t1/1.425,
*1.5*t1, t1/1.425, t1/1.425, 3*t1, t1, t1, t1, 1.5*t1, t1/1.425, t1/1.425, 1.5*t1, t1/1.425, t1/1.425,*
*3*t1,*
*t1, t1, t1, t1/1.375, t1/1.375, t1/1.375, t1/1.375, 4.5*t1, t1, t1, t1/1.375, t1/1.375, t1/1.375,*
*t1/1.375, 6*t1, t1, t1, t1,*
*t1, t1, 6*t1, 2*t1, 6*t1, t1/3, t1/3, 4*t1,1.5*t1, 3*t1, 1.5*t1, 4.5*t1, t1, 3*t1,*
*1.25*t1, 1.25*t1, 6*t1, 2*t1, 4.5*t1, 4.5*t1, 2*t1, 4.5*t1, 6*t1, 3*t1, 6*t1, 4*t1, 1.5*t1,*
*1.5*t1, 9*t1};*

int tone_mus2 [] = {D4, 0, D4, 0, D4, 0, LApp3, F4, D4, 0, LApp3, F4, D4, 0, G4, 0, G4,
0, G4, 0, Gpp4,F4, D4, 0, LApp3, F4, D4};
*int temp_mus2 [] = {t2, t2/4, t2, t2/4, t2, t2/4, t2, t2/4, t2, t2/4, t2, t2/4, 2*t2, t2/4, t2, t2/4,*
*t2, t2/4, t2, t2/4, t2, t2/4, t2, t2/4, t2, t2/4, t2, t2/4, 2*t2};*

int tone_mus3 [] ={H4, 0, H4, 5, H4, 0, H4, 5, H4, 0, C5, 5, C5, 5, D5, 0, D5, 5, D5, 0,
D5, 5, D5, 0, C5, 5, C5, 5, H4, 0, H4, 5, H4, 0, H4, 5, H4, 0, C5, 5, C5, 5, D5, 0, D5,
*5, D5, 0, D5, 5, D5, 0, C5, 5, C5, 5/*55*/,*
D6, LA6, G6, LApp6, LA6, 0, F6, G6, LA6, LApp6, LA6, D6, };
*int temp_mus3 [] ={t3, 2*t3, t3, t3, t3, 2*t3, t3, t3, t3, 2*t3, 2*t3, t3, 2*t3, t3, t3, 2*t3, t3, t3,*
*t3, 2*t3, t3, t3, t3, 2*t3, 2*t3, t3, 2*t3, t3, t3, 2*t3, t3, t3, t3, 2*t3, t3, t3, t3, 2*t3, 2*t3, t3,*
*2*t3, t3, t3, 2*t3, t3, t3, t3, 2*t3, t3, t3, t3, 2*t3, 2*t3, t3, 2*t3, t3/*55*/,*
*t32/2, t32/2, t32/2, t32/2, t32, t3/2, t32/2, t32/2, t32/2, t32, t32/2, 1.5*t32 };*

int tone_mus4 [] = {C7, H6, LA6, 0, C7, H6, LA6, 0, G6, Fpp6, E6, 0, E6, Fpp6, G6, LA6,
H6, 0, C7, H6, LA6, 0, C7, H6, LA6, 0, H6, 0, H6, 0, C7, 0, D7, 0, E7, D7, C7, 0,
E7, D7, C7, 0, H6, LA6, G6, 0, G6, LA6, H6, C7, D7, 0, E7, D7, C7, 0, E7, D7, C7, 0, H6, 0};
int temp_mus4 [] = {t4s, t4s, t4ws, 10, t4s, t4s, t4ws, 10, t4s, t4s, t4ws, 10, t4s, t4s, t4s, t4s,

```
t4ws/2, 10, t4s, t4s, t4ws, 10, t4s, t4s, t4ws, 10, 1.5*t4ws, 10, /*29*/ t4ws/2, 10, t4ws/2, 10,
t4ws/2, 10, t4s, t4s, t4ws, 10,
 t4s, t4s, t4ws, 10,  t4s, t4s, t4ws, 10,  t4s, t4s, t4s, t4s, t4ws/2, 10,  t4s, t4s, t4ws, 10,  t4s, t4s,
t4ws, 10,  1.5*t4ws, 10};

int* tone_mus[] = {tone_mus1, tone_mus2, tone_mus3, tone_mus4}; // Масив вказівників на
масиви нот в Герцах
int* temp_mus[] = {temp_mus1, temp_mus2, temp_mus3, temp_mus4}; // Масив вказівників
на масив тривалості нот
int number[4] = {0}; // Масив, який зберігає кількість нот для кожної композиції


void setup() {
  pinMode(sound, OUTPUT);
  number[0] = sizeof(tone_mus1)/sizeof(tone_mus1[0]); // Операція отримання кількості
елементів масиву tone_mus1[]
  number[1] = sizeof(tone_mus2)/sizeof(tone_mus2[0]);
  number[2] = sizeof(tone_mus3)/sizeof(tone_mus3[0]);
  number[3] = sizeof(tone_mus4)/sizeof(tone_mus4[0]);
}

void loop() {
  if(mus_var<5 && mus_var>0){   // Перевіряємо, чи відповідає варіант мелодії дозволе-
ним значенням
     Melody();   // Виклик мелодії
   }
}

/* Функція відтворення мелодії */
void Melody(){
  int num = mus_var - 1; // Переведення в потрібні одиниці 1 - 4 ---> 0 - 3
  for(int i=0; i<number[num]; i++){  // Цикл для відтворення всіх нот композиції
     note(*(tone_mus[num]+i), *(temp_mus[num]+i)); // Виклик функції відтворення ноти
   }
  delay(2000);
}

 /* Функція для відтворення ноти з частотою n та тривалістю t*/
void note(int n, int t){
  if(n!=0){ // Якщо частота не нульова, то відтворюємо ноту
    tone(sound, n, t);        // Відтворення ноти
   }
  delay(t);  // Затримка виконання програми на час звучання ноти
}
```

Реєстр. № НП 22/23-390. Обсяг 5 авт. арк.

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
проспект Перемоги, 37, м. Київ, 03056
https://kpi.ua
Свідоцтво про внесення до Державного реєстру видавців, виготовлювачів
і розповсюджувачів видавничої продукції ДК № 5354 від 25.05.2017 р.