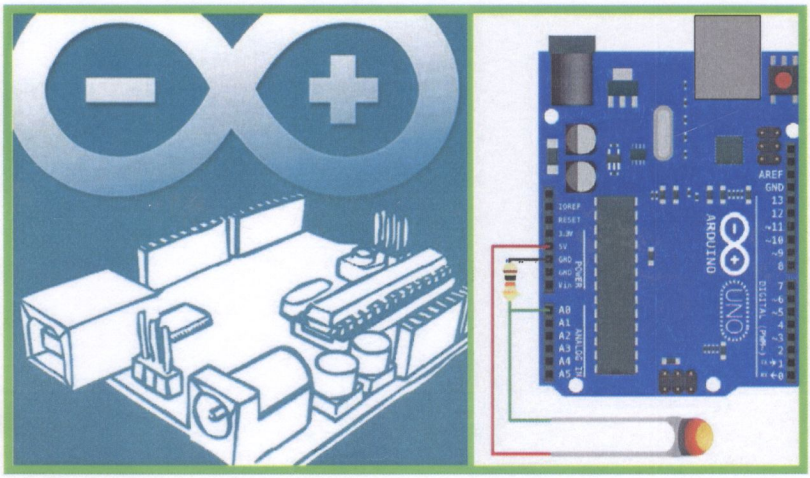
Д. В. Гаврілов, О. В. Осадчук, О. С. Звягін

ОСНОВИ КОМП'ЮТЕРНОГО ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ РЕА

Частина 1



Міністерство освіти і науки України
Вінницький національний технічний університет

Д. В. Гаврілов, О. В. Осадчук, О. С. Звягін

ОСНОВИ КОМП'ЮТЕРНОГО ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ РЕА. ЧАСТИНА 1

Лабораторний практикум

Вінниця
ВНТУ
2016

УДК 621.37
ББК 32
Г12

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 6 від 29.01.2015 р.)

Рецензенти:
В. Ю. Кучерук, доктор технічних наук, професор
В. Г. Петрук, доктор технічних наук, професор
О. М. Шинкарук, доктор технічних наук, професор

Гаврілов, Д. В.

Г12 Основи комп'ютерного проектування та моделювання РЕА. Частина 1 : лабораторний практикум / Гаврілов Д. В., Осадчук О. В., Звягін О. С. – Вінниця : ВНТУ, 2016. – 104 с.

Лабораторний практикум поєднує виконання практичних завдань і лабораторних досліджень пристроїв на сучасних мікроконтролерах (МК). Одночасно з вивченням МК і елементної бази передбачено опанування повноциклової системи автоматизованого проектування Arduino, програмного пакета створення електричних схем Fritzing та комплексного засобу для розробки друкованих плат Eagle. З метою активізації навчального процесу завдання охоплює створення діючого макета для експериментального дослідження певного пристрою згідно з варіантом індивідуального завдання.

Матеріал є базою також для курсового і дипломного проектування з можливістю виготовлення експериментального макета.

УДК 621.37
ББК 32

ЗМІСТ

ВСТУП

Лабораторний практикум охоплює матеріал до виконання практичних завдань і лабораторних досліджень. Засвоєння основ комп'ютерного проектування та моделювання радіоелектронної апаратури здійснюється на базі сучасних мікроконтролерів (МК) відповідно до останніх досягнень мікроелектроніки. Проте одночасно вивчається також типова елементна база, яку зручно запроваджувати до МК у вигляді стандартних структурних компонентів. Крім того, метою лабораторного практикуму є набуття студентами навичок сучасної інженерної праці, коли левова частка розробки і налагодження нової радіоелектронної апаратури (РЕА) та діагностики під час експлуатації діючої РЕА має припадати на системи автоматизованого проектування (САПР) [1].

Для засвоєння більшості систем мікроконтролерів користувачу необхідно опрацювати велику кількість джерел інформації, а також засвоїти достатньо складну технічну документацію. Середовища програмування, як правило, достатньо громіздкі і розраховані на професійних програмістів. Таким чином, доступ в світ мікроконтролерів залишається для деяких назавжди закритим.

Arduino – це проста для освоєння платформа з відкритим кодом на основі вбудованого мікроконтролера і середовища розробки з програмним інтерфейсом API для мікроконтролерів. Для взаємодії між людиною і мікроконтролером можуть підключатись різні аналогові і цифрові датчики, які реєструють стан навколишнього середовища і передають дані в мікроконтролер. Мікроконтролер обробляє вхідні дані, а програма видає нові дані у вигляді аналогових або цифрових значень. У результаті відкриваються широкі горизонти для творчості [2].

Через складність архітектури сучасних МК застосовуються нові автоматизовані методи синтезу цифрових структур і комплексні засоби проектування РЕА на персональних комп'ютерах. Зокрема, для МК фірми Atmel, які часто розглядаються як стандарти нової елементної бази, впроваджено декілька пакетів САПР, версії яких постійно оновлюються. Хоч САПР різних фірм мають відмінності, методика проектування на їх основі є багато в чому спільною. У цьому лабораторному практикумі користуватимемося САПР Arduino, яка підтримує університетську освіту і пристосована для нових МК фірми Atmel і деяких інших фірм. На відміну від попередніх САПР (типу CAD – computer aided design) така система є *повноцикловою* (типу EDA – electronic design automation). Це означає, що вона являє собою закінчене проектне середовище для реалізації повнофункціонального пристрою,

тобто забезпечує всі етапи проектування – від введення проекту до моделювання і фізичного програмування та тестування результатів.

У САПР *проектом* прийнято називати комплект файлів, створених користувачем і програмним забезпеченням для досягнення поставленої мети. *Введення проекту* здійснюється інструментальними засобами у традиційній для інженера графічній формі (принципова електрична схема) або в текстовій формі (програма апаратною мовою високого рівня) з використанням бібліотеки бази даних – стандартних компонентів САПР або створених користувачем блоків. Для побудови РЕА найдоцільнішою є ієрархічна структура, за якою проект вводиться на рівні блок-схеми (проект верхнього рівня), а окремі блоки подаються у текстовому редакторі (проекти нижніх рівнів). Складні проекти, компонентами яких можуть бути мікропроцесори, програмовані логічні інтегральні схеми, цифрові сигнальні процесори, блоки пам'яті та ін., створюються спеціальними засобами САПР на рівні системи. Результатом введення проекту є низка сформованих проектних файлів.

Другим і найважливішим етапом створення проекту є його *компіляція* (compilation), яка виконується автоматично, в декілька кроків. Після завантаження проектних файлів до компілятора здійснюється їх *аналіз*, щоб сформувати рівняння залежності вихідних функцій від вхідних змінних, і *синтез*, щоб реалізувати проект за використання мінімуму доступного ресурсу МК. При цьому виявляються помилки, наприклад, коли відсутнє джерело сигналу на деякому вході, і застереження, наприклад, про невикористовуваний деякий компонент (до усунення помилок компіляція зупиняється). За результатами модуль компілятора генерує об'єктні файли для виготовлення пристрою, що працює реально. Останнім кроком компіляції є *часовий аналіз* (timing analyse), який полягає у вимірюванні затримки сигналів різними шляхами їх поширення, визначенні критичних шляхів і, отже, найгіршої швидкодії, тобто придатності проекту для реалізації заданої мети.

Під час *функціонального моделювання* з переліку сигналів, що міститься в службових файлах компілятора, формується файл часових діаграм. Завдяки цьому функціональне моделювання дозволяє випробувати і налагодити пристрій перед втіленням його в кристалі, що значно скорочує термін проектування.

Завершальним етапом є фізичне *програмування/конфігурування* мікросхеми, яке здійснюється з об'єктних файлів програматором. Крім того, залежно від використовуваних апаратних засобів програматор може виконувати перевірку функціонування готового виробу шляхом подачі на його входи випробувальних сигналів і порівняння вихідних сигналів зі сформованими часовими діаграмами.

З метою активізації навчання лабораторний практикум має варіантний характер. Для зарахування кожної роботи необхідно: а) засвоїти теоретичний матеріал з даної теми; б) виконати практичне завдання згідно зі своїм варіантом; в) виконати загальну частину лабораторного завдання ознайомчого характеру; г) створити проекти за індивідуальним завданням з моделюванням результатів; д) створити лабораторний макет згідно з заданим варіантом та виконати експериментальні дослідження (відомості про лабораторний стенд містяться в додатках). Щоб виконати завдання, потрібно засвоїти також процедури і елементи програмного забезпечення САПР Arduino (рекомендується обмежитися мінімумом відомостей про саму систему, а описувати тільки функції, достатні для виконання даної роботи). Методично практикум побудовано таким чином, аби практичні і лабораторні завдання можна було виконати самостійно, у дистанційному режимі (експериментальні дослідження з необхідними комплектуючими виконуються в лабораторії за наявності заздалегідь підготовлених файлів).

Зміст звіту з виконання завдання

Назва і мета роботи; результати виконання домашнього завдання; зміст кожного пункту лабораторного завдання: бібліотечні файли (частинами або в скороченому вигляді, щоб помітно було деталі), файли зі схемами, діаграмами та іншими результатами зі стислими поясненнями і висновками; результати експериментальних досліджень, які можна подавати скорочено у формі таблиць і стислих висновків [1].

ЛАБОРАТОРНА РОБОТА № 1

Вступ до основ програмування МК у середовищі Arduino IDE

Мета роботи. Засвоєння основ програмування та отримання початкових знань для створення початкових проектів з Arduino і засвоєння роботи з програмним забезпеченням.

Програмне забезпечення для лабораторної роботи. Лабораторна робота виконується за допомогою програми «Arduino IDE». Опис програми наведений в додатку А.

Короткі теоретичні відомості та перші навички

Для засвоєння принципів роботи з МК необхідно виконати декілька ознайомчих проектів. Дані проекти використовують світлодіоди в різні способи, це дозволить ознайомитись з управлінням виходами та входами в Arduino. Також виконання проектів дозволить ознайомитись з такою елементною базою, як: світлодіоди, кнопки і резистори, в тому числі дасть можливість вибирати і застосовувати резистори необхідного номіналу, що є важливим в забезпеченні роботи пристроїв введення та виведення. У загальному, ви ознайомитесь з основними концепціями програмування в Arduino.

Традиційно, ознайомлення з програмуванням МК починається з проекту «Привіт, світ!» («Hello, World!»), в якому плата Arduino буде блимати зовнішнім світлодіодом.

Тепер створимо першу роботоздатну програму Arduino. Для тренування можна просто перенабрати код програми. Відкрийте ваш Arduino IDE і введіть код з лістингу, який наведений на рис. 1.1.

```
//  Flasher
int led = 13;
void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Рисунок 1.1 – Лістинг програми

Натисніть кнопку «Проверить / Компилировать» у верхній частині IDE, щоб переконатися, що немає ніяких помилок у коді. Якщо все пройде успішно, натисніть кнопку «Загрузить», щоб завантажити код у ваш

Arduino. Приклад коду у середовищі Arduino IDE наведено на рис. 1.2. Якщо все зроблено правильно, то ви повинні побачити блимання червоного світлодіода на платі.

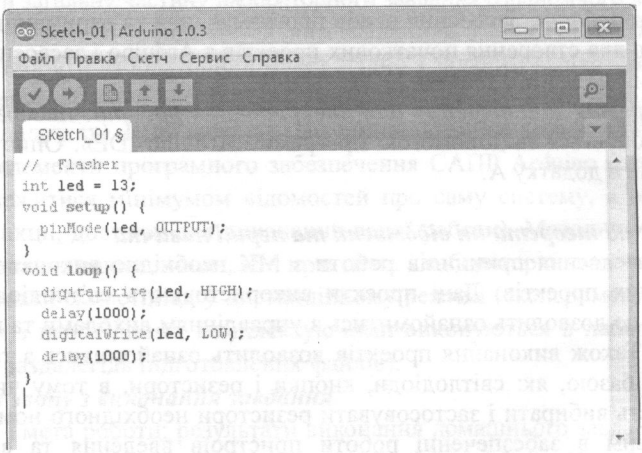


Рисунок 1.2 – Лістинг програми у Arduino IDE

Ваша програма розраховує суму двох чисел і виводить результати через термінал. Після виведення результату розпочне мигати світлодіод L, який розташований на платі. З натисканням кнопки «Сброс» програма розпочне працювати з початку.

Необхідні комплектуючі:

– плата мікроконтролера Arduino / Freeduino;

– панель з контактними гніздами;

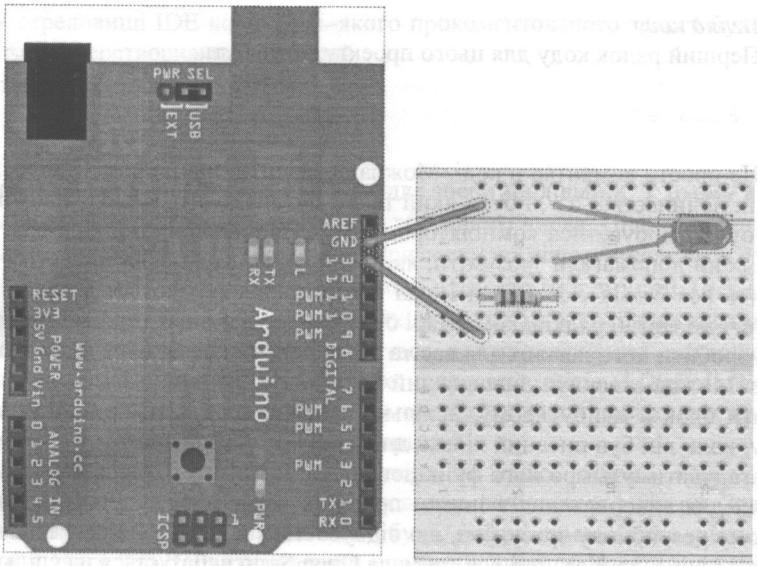
– резистор 100 Ом (номінал резистора може відрізнятися залежно від обраного світлодіода);

– світлодіод;

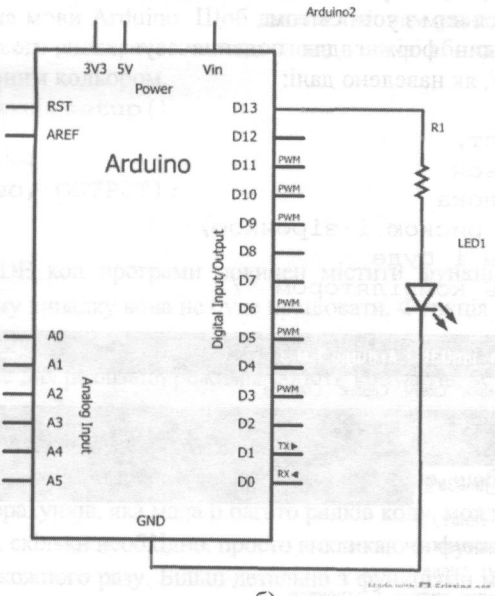
– два гнучких монтажних провідники завдовжки приблизно 5 см.

Підключення всіх елементів

По-перше, переконайтеся, що ваша плата Arduino вимкнена, тобто відключена від USB-кабеля. Тепер візьміть ваш макет, LED1 (світлодіод), резистор R1 і провідники та з'єднайте все, як зображено на рис. 1.3. Якщо ви впевнені, що все підключено правильно, продемонструйте проект викладачу, після перевірки можна під'єднати USB-кабель. Давайте подивимося на код та обладнання, щоб з'ясувати, як вони працюють.



а)



б)

Рисунок 1.3 – Зовнішній вигляд і підключення (а) та схема електрична принципова (б) пристрою

Огляд коду

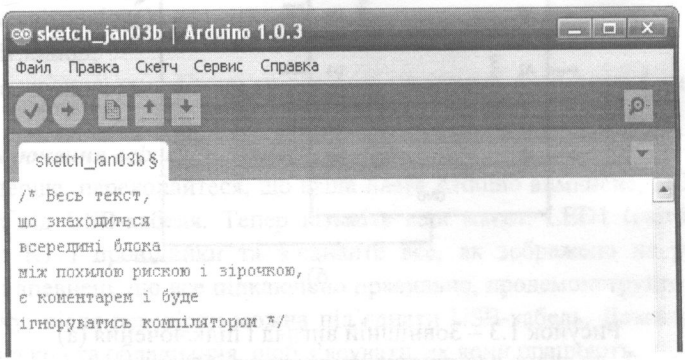
Перший рядок коду для цього проекту може бути:

```
//  Flasher
```

Це просто коментар в коді. Можна визначити, що це коментар, оскільки він починається з //, і будь-який текст, що знаходиться після двох похилих буде ігноруватися компілятором. Коментарі необхідні в коді для того, щоб вони допомагали зрозуміти, як працює код. Коли ваші проекти ставатимуть все більш складними і ваш код розширюватиметься до сотні або, може, і до тисячі рядків, коментарі будуть мати життєво важливе значення, щоб зробити його легким для вас та побачити, як кожен блок коду функціонує. Можна створити дивовижний блок коду, але буде важко використовувати блок в майбутньому, оскільки складно зрозуміти принцип роботи коду, коли він був писаний кілька днів, тижнів або місяців тому. Коментарі ж нагадуватимуть про його функціональності. Крім того, якщо код призначений для використання в іншим проектах, коментарі будуть допомагати іншому розробнику зрозуміти, що відбувається в коді. Усі користувачі систем Arduino, та й взагалі вся громада Open Source, базується на спільному використанні коду та схем. Сподіваємось, що коли ви почнете більш плідно працювати з МК, то будете мати свій власний цікавий матеріал і будете готові поділитися ним з усім світом.

Існує ще один формат для подання зауважень, це оператор блока bookended /* і */, як наведено далі:

```
/* Весь текст,
що знаходиться
всередині блока
між похилою рискою і зірочкою,
є коментарем і буде
ігноруватись компілятором */
```



У середовищі IDE колір будь-якого прокоментованого тексту буде автоматично перетворюватися у сірий.

Наступний рядок програми

```
int led = 13;
```

називається змінною. Змінна є місцем для зберігання даних. У цьому випадку налаштовується змінна типу int або цілого числа. Ціле – це число в діапазоні від -32768 до +32767. Далі призначаємо цілому числу ім'я led і надаємо йому значення 13 (необов'язково надавати ім'я змінній led, можна було б назвати її як завгодно. Але якщо даємо ім'я led, тим самим показуємо, що ця змінна використовується для підключення світлодіода). У цьому випадку ви використовуєте цифровий вихід / *Digital Pin 13*. Наприкінці цієї команди є крапка з комою. Цей символ повідомляє компілятору, що ця команда в даний час завершена.

Для того, щоб ви змогли звернутись до вашої змінної, кожне ім'я змінної в мові програмування Сі (С) повинно починатися з літери. Залишкова частина назви може складатися з букв, цифр і символів підкреслення. Зверніть увагу, що (С) визнає верхні і рядкові символи. Нарешті, ви не можете використовувати будь-яке з ключових слів класу (С) та перемикання як ім'я змінної. Ключові слова, постійні, змінні і імена функцій визначаються як частина мови Arduino. Щоб допомогти вам уникнути імені змінної після ключового слова, всі ключові слова в ескізі або скетчі (scetch) виділяються червоним кольором.

Далі йде функція setup()

```
void setup() {
  pinMode(led, OUTPUT);
}
```

У Arduino IDE код програми повинен містити функції setup() та loop(), в іншому випадку вона не буде працювати. Функція setup() запускається один раз на початку програми перед функцією loop(), оголошення функції необхідне для реалізації режимів роботи контактів, встановлення швидкості передачі даних і таке інше.

У принципі, функція – це частина коду, що зібрана в одному зручному блоці. Наприклад, якщо створено свою власну функцію, щоб провести серію математичних обрахунків, яка мала б багато рядків коду, можна запустити цей код стільки разів, скільки необхідно, просто викликаючи функцію замість того, щоб писати код кожного разу. Більш детально з функціями ми ознайомимось пізніше, коли почнемо створювати свої власні проекти.

Функція починається з

```
void setup()
```

Це повідомляє компілятору, що ваша функція, яка називається `setup` не повертає дані `void`, а також використовується без параметрів () – порожні дужки. Якщо ваша функція повертає ціле значення і до функції надходять теж цілі значення (наприклад, для функції обробки), то це буде виглядати так:

```
int myFunc(int x, int y)
```

Тут створено функцію (або блок коду) під назвою `MyFunc`. Ця функція передає два цілих значення x і y. Після того, як функція закінчена, вона буде повертати ціле значення в точку, після якої ваша функція оголошена в програмі. Весь код у функції міститься у фігурних дужках: { – символ початку блока коду і } – символ завершення блока коду. Все, що знаходиться між цими двома символами, – це код, який належить функції. У цій програмі є дві функції. Перша функція називається налаштування і її мета полягає в установленні всього необхідного для вашої програми, щоб працювати до початку роботи функції циклу:

```
void setup() {
  pinMode(led, OUTPUT);
}
```

Функція встановлення має тільки один оператор, і це `pinMode`, який говоритиме Arduino, що ви хочете встановити один з ваших виводів у режим виведення, а не введення. У дужках вписується номер виводу (pin number) і режим роботи (виведення даних – OUTPUT або введення даних – INPUT). Номер `led`, який був раніше встановлений як порт виведення 13. Таким чином, це твердження просто говорить Arduino, що цифровий вивід (pin) 13 повинен перейти на режим виведення даних. Оскільки функція `setup()` працює тільки один раз, то перейдемо до основної функції циклу (петля, тобто `loop`)

```
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Функція *петля* `loop()` є основною функцією програми і працює безперервно, поки Arduino включений. Кожне твердження (statement) у функції `loop()` (всередині фігурних дужок) виконується, одне за одним, крок

за кроком, поки останній рядок функції не буде досягнутий, потім цикл починається знову у верхній частині функції, і так до нескінченності або поки ви не натиснете кнопку «Сброс». Перше твердження

```
digitalWrite(led, HIGH);
```

Дане твердження встановлює вивід у високий HIGH або низький LOW стан (в даному випадку світлодіод, що підключений до порту 13). При встановленні виводу в високий стан на нього подається напруга 5 В. Коли встановлюється низький стан, напруга на контакті стає рівною 0 В, або заземлення. Ця команда надсилає 5 В до контакту 13 і вмикає світлодіод.

Після цього надходить наступна команда

```
delay(1000);
```

яка вказує Arduino чекати 1000 мілісекунд перед виконанням наступної команди

```
digitalWrite(led, LOW);
```

яка буде вимикати живлення на цифровому виході 13, тобто відключить світлодіодний індикатор. Після чого задається наступна затримка ще на 1000 мілісекунд, після чого функція закінчується. Однак, оскільки це основна функція циклу (петлі), то функція буде починатись знову з самого початку. Ідучи крок за кроком, ви можете бачити, що це дуже просто [1].

Хід лабораторної роботи

1. Здобути навички з користування програмним забезпеченням Arduino IDE:
   - набрати код програми;
   - перевірити код на помилки, виконавши компіляцію;
   - завантажити код до мікроконтролера Atmel, що знаходиться на платі Arduino;
   - перевірити функціональність пристрою;
   - підключити зовнішній світлодіод;
   - перевірити функціональність пристрою з зовнішнім світлодіодом.
2. Засвоїти лістинг програми та зрозуміти призначення команд.
3. Виконати індивідуальне завдання за варіантом:
   - створити код програми відповідно до індивідуального варіанта завдання, навести коментарі до функцій програми;
   - перевірити код на помилки;
   - завантажити код до мікроконтролера Atmel, що знаходиться на платі Arduino;
   - перевірити функціональність пристрою;

- оформити звіт за результатами роботи.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрій для періодичного вмикання та вимикання на двох світлодіодах з різними затримками та інтервалом світіння.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 1.1.

Таблиця 1.1 – Варіанти завдань

Номер варіанта	Затримка 1-го світлодіода	Затримка 2-го світлодіода	Порти виходу
1	2000	500	11, 12
2	250	5000	A2, 6
3	50	50	2, 6
4	400	800	A3, 4
5	2500	100	3, 5
6	100	500	13, A4

ЛАБОРАТОРНА РОБОТА № 2

Конфігурація входу / виходу та встановлення порту

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також отримання початкових навичок програмування з використанням команд встановлення порту.

Короткі теоретичні відомості

Для того щоб використовувати вивід МК Atmel на платі Arduino як порт введення або виведення, потрібно вказати це у функції `void setup()`. Далі ми розглянемо, як це зробити.

Команда `pinMode(pin, mode);` використовується у функції `void setup()`, щоб конфігурувати контакт плати або як вхід, або як вихід:

```
pinMode(pin, OUTPUT); // Вивід використовуєть-
                      // ся як вихід
```

У МК ATmega також є вбудований підтягувальний резистор (20 кОм), який управляється за допомогою програмного забезпечення. До вбудованого підтягувального резистора можна звертатися таким чином:

```
pinMode(pin, INPUT);       // Вивід встанов-
                           // люється як вхід
digitalWrite(pin, HIGH); // Підключається
                           // підтягуючий
                           // резистор
```

Підтягувальний резистор зазвичай підключають, щоб приєднувати входи як комутатор. Видно, що вивід конфігурується як вхід і на ньому встановлюється високий рівень. Це лише метод активізувати внутрішній підтягувальний резистор.

Виводи, що конфігуруються як вихід, мають малий повний опір і можуть навантажуватися приєднаними елементами і схемами зі струмом максимум 40 мА. Цього вистачить для запалювання світлодіода (з урахуванням послідовно включеного опору), але недостатньо, щоб експлуатувати більшість типів реле, соленоїдів або електродвигунів. Короткі замикання виводів плати Arduino , а також занадто великі струми можуть пошкодити вихідний вивід і навіть вивести з ладу мікроконтролер. Тому краще підключати зовнішні компоненти до виходу через резистор 470 Ом або 1 кОм.

Функція `digitalRead(pin)` зчитує значення заданого цифрового виводу з результатом `HIGH` або `LOW`, що відповідає логічній 1 або логічному 0. Номер виводу може встановлюватися або як змінна, або як константа (від 0 до 13).

```
value = digitalRead(pin); // Встановлює value,
                          // рівне значенню
                          // на вхідному виводі
```

Функція `digitalWrite(pin, value)` встановлює рівень `HIGH` або `LOW` на заданому виводі. Номер виводу може задаватися або як змінна, або як константа (від 0 до 13).

```
digitalWrite(pin, HIGH); // Встановлює на виводі
                         // високий рівень (+5В)
```

Зчитування стану кнопки

У наступному прикладі (рис. 2.1) зчитується стан кнопки, підключеної до цифрового входу (контакт 12), і результат відображається за допомогою світлодіода. При натисканні на кнопку світлодіод гасне, а при відпусканні – спалахує. Тут справа в тому, що було активізовано внутрішній підтягувальний резистор і, таким чином, вхідний вивід 12 має високий рівень. Якщо при натисканні кнопки, яка підключає вивід до GND (загальний провід або заземлення), світлодіод не горить, то на вході буде низький рівень. Зовнішній вигляд макета наведено на рис. 2.1, а електрична принципова схема показана на рис. 2.2 [1, 2]. Лістинг програми наведено на рис. 2.3.

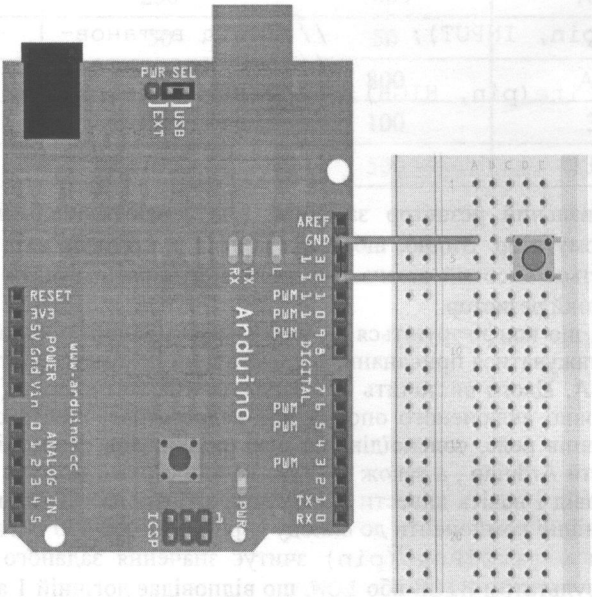


Рисунок 2.1 – Зовнішній вигляд макета

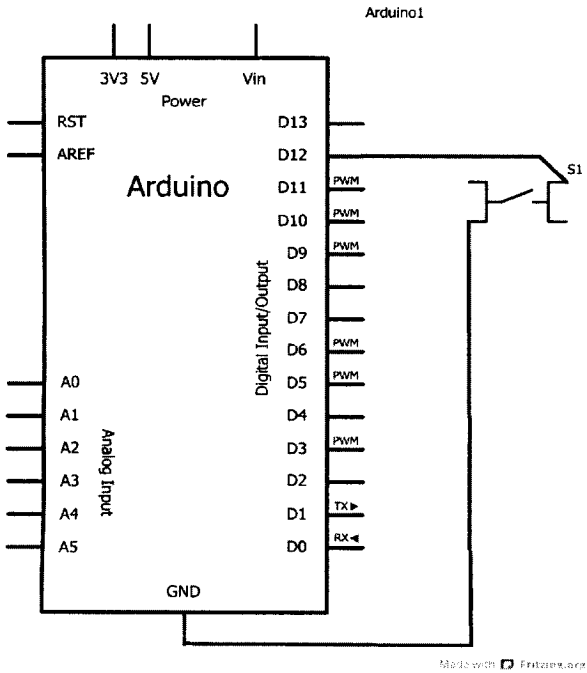
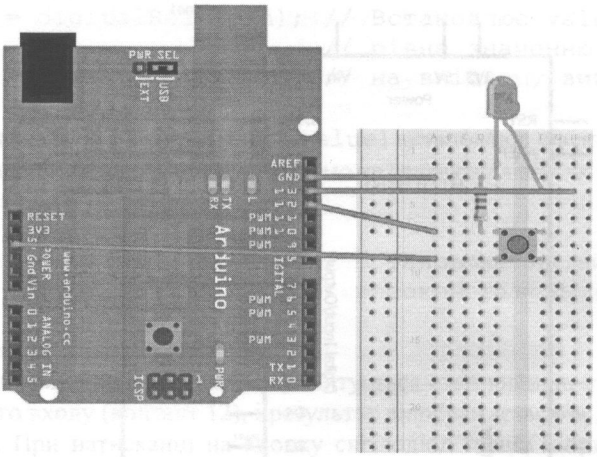


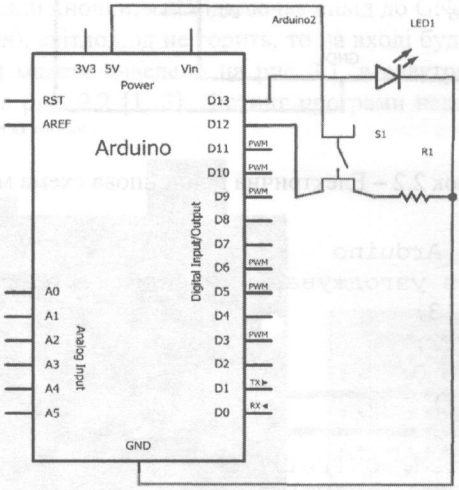
Рисунок 2.2 – Електрична принципова схема макета

```
// Franzis Arduino
// Кнопка з узгоджувальним резистором
int led = 13;
int pin = 12;
int value = 0;
void setup()
{
  pinMode(led, OUTPUT);
  pinMode(pin, INPUT) ;
  digitalWrite(pin, HIGH);
}

void loop()
{
  value = digitalRead(pin);
  digitalWrite(led, value);
}
```

Рисунок 2.3 – Лістинг програми



а)



б)

Рисунок 2.4 – Зовнішній вигляд (а)
та електрична принципова схема (б) макета

Необхідні комплектуючі:

– плата мікроконтролера Arduino / Freeduino;
– панель з контактними гніздами;
– кнопка;
– резистор 10 кОм;

– світлодіод;

– чотири гнучких монтажних провідники завдовжки приблизно по 5 см.

Хід лабораторної роботи

1. Моделювання пристрою з внутрішнім підтягувальним резистором:

- скласти макет, зображений на рис. 2.1;
- підключити макет до ПК;
- запустити програму Arduino IDE;
- набрати лістинг програми, поданий на рис. 2.3;
- перевірити формування файлу, який буде записаний на мікроконтролер;
- записати файл на мікроконтролер;
- перевірити функціональність пристрою.

2. Дослідження та розробка власного пристрою із зовнішнім підтягувальним резистором:

- скласти макет, зображений на рис. 2.4;
- набрати лістинг програми для свого варіанта;
- перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, що містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, який виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої з зовнішнім та внутрішнім підтягувальним резистором.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 2.1.

Таблиця 2.1 – Варіанти завдань

Номер варіанта	Зміна 1	Зміна 2	Зміна 3	Порти входу
01	Bulb1	put1	Nals1	11,6
02	Bulb2	Put2	Nals1	10,2
03	Bulb3	Put3	Nals1	9,3
04	Bulb4	Put4	Nals1	8,4
05	Bulb5	Put5	Nals1	7,5
06	Bulb6	Put6	Nals1	6,11

ЛАБОРАТОРНА РОБОТА № 3

Робота зі світлодіодами

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування та принципів підбору додаткового опору світлодіода.

Короткі теоретичні відомості

У більшості описаних експериментів для тестування програмного забезпечення застосовувалися один або декілька світлодіодів. Людині, яка не є інженером-електронником, напевно цікаво знати, як розраховується додатковий опір.

Світлодіод можна розглядати як звичайний діод, включений в прямому напрямку (анод, підключений до плюса джерела живлення, і катод – до загального проводу). У цьому випадку до світлодіода прикладається напруга (1,6...3,5 В), яка залежить від кольору випромінювання. Точні дані можна знайти в паспорті світлодіода, де вказано пряму напругу V_f (Forward Voltage). Для світіння світлодіода потрібен певний струм; у паспорті вказано струм I_f.

Приклад розрахунку

Нехай прямий струм $I_f = 2$ мА (світлодіод з малим робочим струмом). Прямий спад напруги $V_f = 2{,}2$ В. Напруга джерела живлення $V_{cc} = 5$ В. Шукана величина – опір додаткового резистора R = ? Ом.

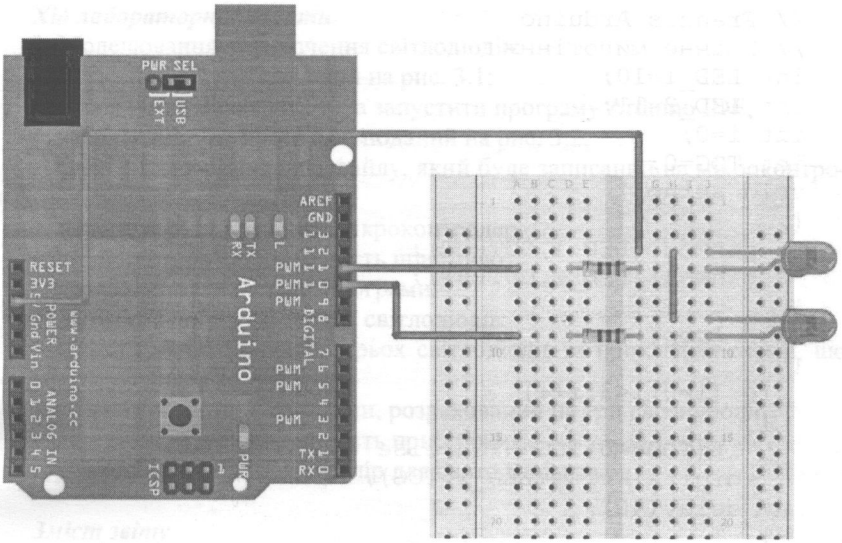
$$R = (V_{cc}-V_f) / I_f.$$

Для нашого прикладу:

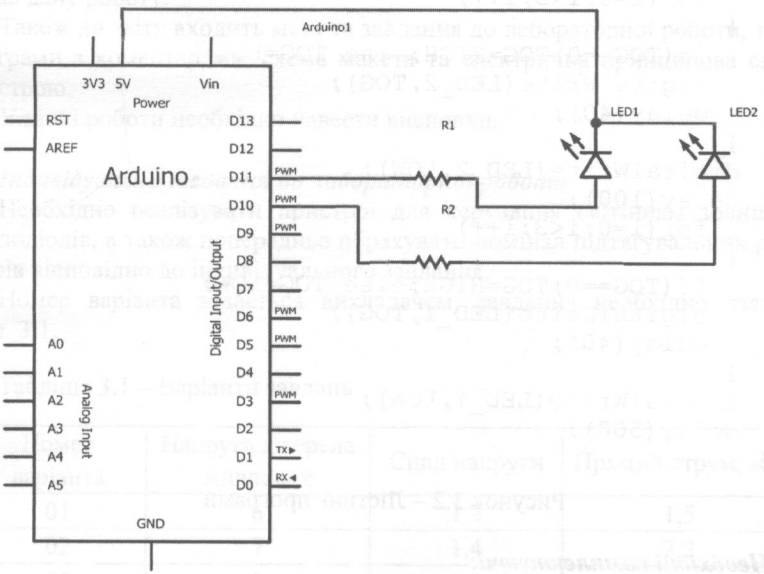
$$R = (5 \text{ В} - 2{,}2 \text{ В}) / 2 \text{ мА} = 1400 \text{ Ом}.$$

Доцільно вибрати опір з ряду E12 трохи більшого номіналу. Для надійності ставиться резистор 1,5 кОм, щоб виключити пошкодження світлодіода. Якщо уважно розглянути монтажну схему експериментальної плати, то можна виявити перед діодами опір 1,5 кОм.

Розглянемо практичний приклад включення світлодіодів з попереміннім миготінням. Світлодіоди, підключені до виводів 10 і 11 (рис. 3.1), спалахують почергово, що дає в підсумку світловий ефект, схожий на мигалку машини швидкої допомоги, навіть якщо колір червоний, а не синій [2]. Лістинг програми наведено на рис. 3.2.



а)



б)

Рисунок 3.1 – Підключення та зовнішній вигляд (а)
й електрична принципова схема макета (б)

```
// Franzis Arduino
// Змінне миготіння
int LED_1=10;
int LED_2=11;
int i=0;
int TOG=0;
void setup()
{
   pinMode(LED_1,OUTPUT);
   pinMode(LED_2,OUTPUT);
}
void loop()
{
  for(i=0;i<3;i++)
  {
    if(TOG==0)TOG=HIGH;else TOG=LOW;
    digitalWrite(LED_1,TOG);
    delay(40);
  }
  digitalWrite(LED_1,LOW);
  delay(100);
    for(i=0;i<3;i++)
  {
    if(TOG==0)TOG=HIGH;else TOG=LOW;
    digitalWrite(LED_2,TOG);
    delay(40);
  }
  digitalWrite(LED_2,LOW);
  delay(100);
    for(i=0;i<3;i++)
  {
    if(TOG==0)TOG=HIGH;else TOG=LOW;
    digitalWrite(LED_1,TOG);
    delay(40);
  }
  digitalWrite(LED_1,LOW);
  delay(500);
}
```

Рисунок 3.2 – Лістинг програми

Необхідні комплектуючі:

– три світлодіоди червоного кольору та три резистори 1,5 кОм;

– чотири гнучких монтажних провідники довжиною приблизно по 10 см;

– два гнучких монтажних провідники довжиною приблизно по 5 см.

Хід лабораторної роботи

1. Моделювання підключення світлодіодів:
- скласти макет, зображений на рис. 3.1;
- підключити макет до ПК та запустити програму Arduino IDE;
- набрати лістинг програми, поданий на рис. 3.2;
- перевірити формування файлу, який буде записаний на мікроконтролер;
- завантажити програму у мікроконтролер;
- перевірити функціональність пристрою;
- проаналізувати лістинг програми.

2. Дослідження підключення світлодіодів:
- скласти макет на основі трьох світлодіодів за прикладом схеми, що зображена на рис. 3.1;
- розробити лістинг програми, розрахований на три світлодіоди;
- перевірити функціональність пристрою;
- розрахувати додатковий опір для свого варіанта.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої для керування світінням зовнішніх світлодіодів, а також попередньо обрахувати номінал підтягувальних резисторів відповідно до індивідуального завдання.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 3.1.

Таблиця 3.1 – Варіанти завдань

Номер варіанта	Напруга джерела живлення	Спад напруги	Прямий струм, мА
01	6	1,2	1,5
02	7	1,4	2,2
03	8	1,8	3
04	9	2	3,5
05	10	2,3	4
06	15	2,5	4,2

ЛАБОРАТОРНА РОБОТА № 4

Генератор випадкових чисел на МК

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування, уміння використовувати властивості генератора випадкових чисел та розробляти пристрої з застосуванням таких генераторів.

Короткі теоретичні відомості

За допомогою Arduino і трьох світлодіодів (рис. 4.1, рис. 4.2), а також генератора випадкових чисел можна імітувати світло свічки з мерехтінням. Програма (рис. 4.3) використовує аналогові виходи мікроконтролера. Однак, тепер на виході не постійне значення або синусоїдальний сигнал, а послідовність імпульсів випадкової тривалості. Світлодіоди (червоного і жовтого кольору) завжди мають різну яскравість. Якщо встановити світлодіоди в скляну посудину молочного кольору і обклеїти кришку алюмінієвою фольгою, світло буде розподілятися рівномірно. Тоді вже буде складно визначити, світлодіоди там чи справжні свічки. Можна вчинити ще простіше: склеїти маленьку паперову коробочку (10×10×10 см) і помістити всередину неї світлодіоди.

Генератор випадкових чисел – обчислювальний або фізичний пристрій, спроектований для генерації послідовності номерів чи символів, які не відповідають будь-якому шаблону, тобто є випадковими. Широко використовуються комп'ютерні системи для генерації випадкових чисел, але часто вони малоефективні. Ці функції, можливо, забезпечують достатньо випадковості для певних завдань, але є непридатними в тих випадках, коли потрібна «високоякісна випадковість», як наприклад, у криптографічних програмах, статистиці або чисельному аналізі. Методи добування випадкових результатів існували здавна, зокрема, використання грального костей, підкидання монети, тасування ігрових карт та ін.

Існує багато різних методів отримання випадкових даних. Ці методи можуть відрізнятися тим, які непередбачувані чи статистично випадкові дані вони видають, а також як швидко вони можуть генерувати випадкові номери.

До появи обчислювальних генераторів випадкових чисел, для отримання великої кількості достатньо випадкових номерів (важливо в статистиці), треба було виконувати велику кількість обрахунків. Результати іноді узагальнювали й розповсюджували як таблиці випадкових чисел.

Виділяють такі класи датчиків (генераторів) випадкових чисел.

Табличний датчик випадкових чисел являє собою таблицю, заповнену реалізаціями випадкової величини з заданим розподілом. Подані вибірки у таких таблицях дуже якісні, та вони мають обмежений розмір і кількість таких вибірок невелика, що суттєво обмежує їх використання.

Фізичний датчик випадкових чисел конструюється на основі певного електронного пристрою, на виході якого спостерігають потрібну реалізацію випадкової величини. Ці генератори дозволяють отримувати вибірку довільного обсягу та вони мають інший недолік – кожна отримана вибірка унікальна, і її неможливо повторити. Зате наступний клас генераторів не має цього недоліку.

Програмний датчик випадкових чисел будується на базі деякої програми, на виході якої формується потрібна реалізація. Ці програми, зазвичай, базуються на деякій рекурентній формулі. Задаючи однакові початкові члени послідовності, можна щоразу отримувати однакові послідовності, однак ці генератори, у свою чергу, мають інший недолік – вони періодичні. Числа, які вони генерують, називають «псевдовипадковими», бо вони отримуються за чітким детермінованим алгоритмом.

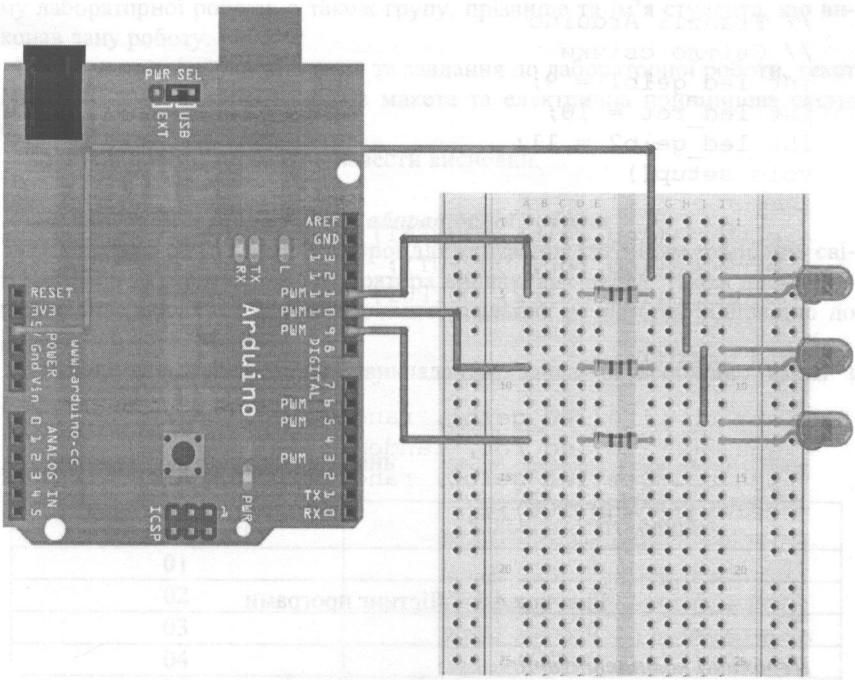


Рисунок 4.1 – Підключення та зовнішній вигляд макета

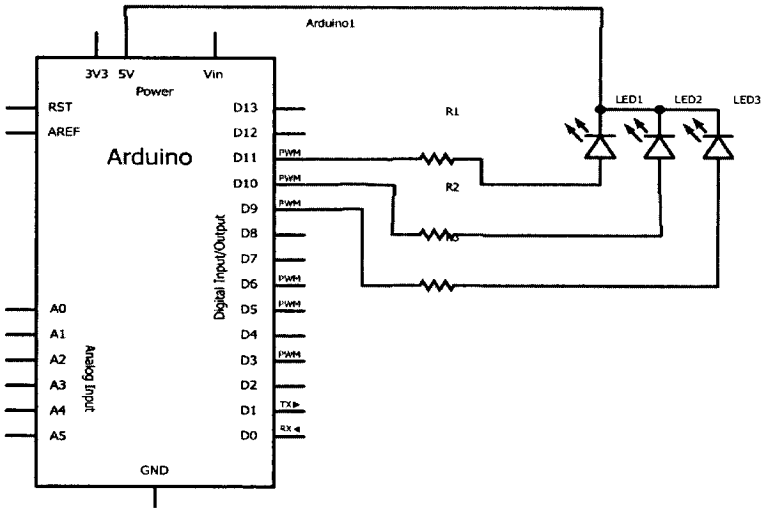


Рисунок 4.2 – Електрична принципова схема макета

```
// Franzis Arduino
// Світло свічки
int led_gelb1 = 9;
int led_rot = 10;
int led_gelb2 = 11;
void setup()
{
  pinMode(led_gelb1, OUTPUT);
  pinMode(led_rot, OUTPUT);
  pinMode(led_gelb2, OUTPUT);
}
void loop()
{
  analogWrite(led_gelb1, random(120)+135);
  analogWrite(led_rot, random(120)+135);
  analogWrite(led_gelb2, random(120)+135);
  delay(random(100));
}
```

Рисунок 4.3 – Лістинг програми

Необхідні комплектуючі:

– два світлодіоди червоного кольору;

– чотири світлодіоди жовтого кольору;

– шість резисторів 1,5 кОм;
– сім гнучких монтажних провідників довжиною приблизно по 10 см;
– п'ять гнучких монтажних провідників довжиною приблизно по 5 см.

Хід лабораторної роботи

1. Моделювання світіння свічки за допомогою МК:
- скласти макет, зображений на рис. 4.1;
- підключити макет до ПК та запустити програму Arduino IDE;
- набрати лістинг програми, поданий на рис. 4.2;
- перевірити формування файлу, який буде записаний на мікроконтролер;
- записати файл на мікроконтролер;
- перевірити функціональність пристрою.

2. Дослідження пристрою за індивідуальним завданням:
- скласти макет для свого варіанта;
- набрати лістинг програми для свого варіанта;
- перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої для керування світінням зовнішніх світлодіодів з використанням генератора випадкових чисел, також необхідно попередньо обрахувати номінал підтягувальних резисторів відповідно до індивідуального завдання.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 4.1.

Таблиця 4.1 – Варіанти завдань

Номер варіанта	Кількість світлодіодів
01	4
02	5
03	6
04	6
05	5
06	4

ЛАБОРАТОРНА РОБОТА № 5

Світлофор

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування. Розробка власного світлофора.

Короткі теоретичні відомості

У даній роботі необхідно створити світлофор, який змінює колір із зеленого на жовтий, на червоний і назад, після певного проміжку часу, використовуючи чотири алгоритми керування світлофором. Цей проект може бути використаний, щоб зробити набір роботи світлофорів для моделювання роботи реального блока світлофора.

Підключіть схему, як показано на рис. 5.1 (електрична принципова схема показана на рис. 5.2). На цей раз ви підключите три світлодіоди, анод кожного з яких підключається до цифрових контактів 12, 11 і 10 кожен через резистор 150 Ом (або будь-яке інше значення залежно від світлодіодів).

Візьміть перемичку від виходу землі на Arduino, що знаходиться у верхній частині макетної плати; дріт заземлення йде від катода кожного світлодіода на масу через резистор (в такій простій схемі немає значення, до якого виводу світлодіода підключається резистор – до анода або катода).

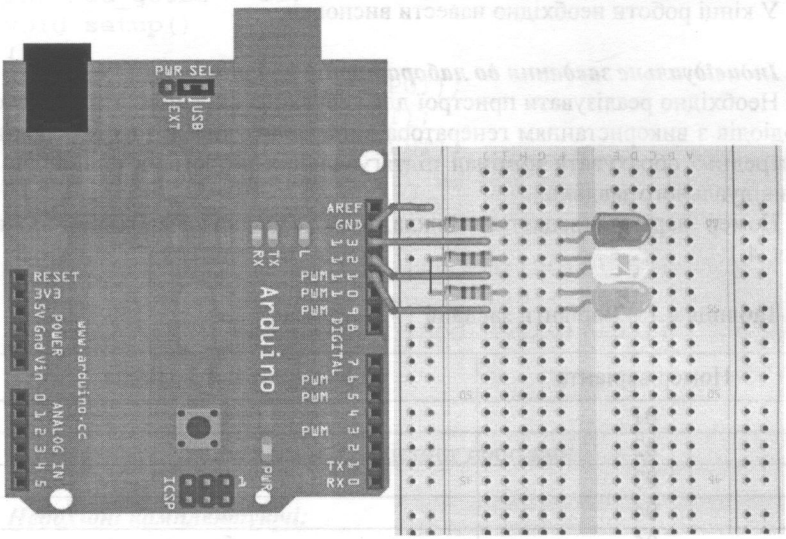
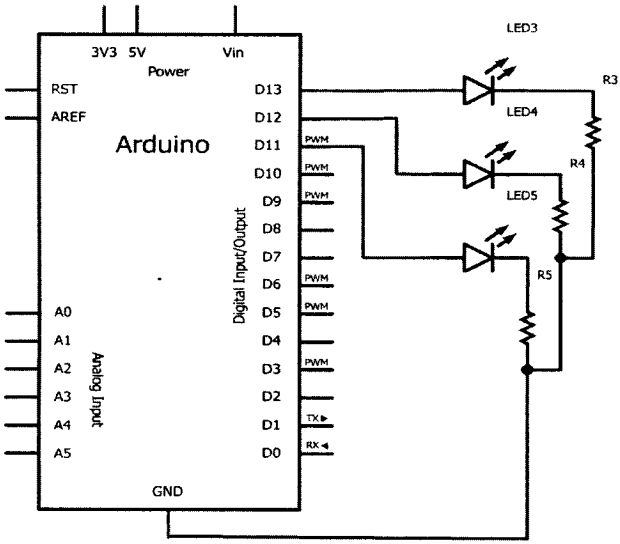


Рисунок 5.1 – Підключення та зовнішній вигляд макета світлофора



б)

Рисунок 5.2 – Електрична принципова схема макета світлофора

Введіть код з лістингу на рис. 5.3, перевірте його, і завантажте у Arduino, після цього світлодіоди засвічуватимуться відповідно до Європейської схеми роботи світлофора, як показано на рис. 5.4.

```
// Світлофор
int ledDelay = 10000; // затримка між перемиканнями
int redPin = 10;
int yellowPin = 9;
int greenPin = 8;
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}
void loop() {

  digitalWrite(redPin, HIGH); // ввімкнути червоне
світло
  delay(ledDelay); // зачекати 5 секунд
```

Рисунок 5.3 – Лістинг програми

```
    digitalWrite(yellowPin, HIGH); // ввімкнути жовте
світло
    delay(2000); // зачекати 2 секунди

    digitalWrite(greenPin, HIGH); // ввімкнути зелене
світло
    digitalWrite(redPin, LOW); // вимкнути червоне
світло
    digitalWrite(yellowPin, LOW); // вимкнути жовте
світло
    delay(ledDelay); // затримка між перемиканнями

    digitalWrite(yellowPin, HIGH); // ввімкнути жовте
світло
    digitalWrite(greenPin, LOW); // вимкнути зелене
світло
    delay(2000); // зачекати 2 секунди

    digitalWrite(yellowPin, LOW); // вимкнути жовте
світло
    // після чого цикл повторюється
}
```

Рисунок 5.3, аркуш 2

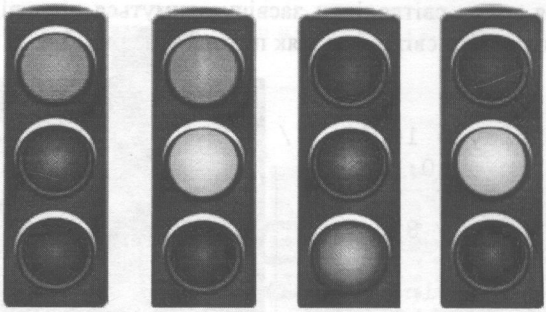


Рисунок 5.4 – Чотири стану світлофора Європейської системи

Необхідні комплектуючі:

– плата мікроконтролера Arduino / Freeduino;
– панель з контактними гніздами;
– резистори 150 Ом;
– світлодіоди трох кольорів;
– чотири гнучких монтажних провідники довжиною приблизно по 5 см.

Хід лабораторної роботи

1. Моделювання роботи пристрою з освоєнням програми:
- скласти макет, зображений на рис. 5.1;
- підключити макет до ПК;
- запустити програму Arduino IDE;
- набрати лістинг програми, поданий на рис. 5.2;
- перевірити формування файлу, який буде записаний на мікроконтролер;
- записати файл на мікроконтролер;
- перевірити функціональність пристрою.

2. Дослідження пристрою за індивідуальним завданням:
- набрати лістинг програми для свого варіанта;
- перевірити функціональність пристрою;
- розробити світлофор реального часу.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої для керування світінням зовнішніх світлодіодів з використанням схеми роботи світлофора, також необхідно попередньо обрахувати номінал підтягувальних резисторів відповідно до індивідуального завдання.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 5.1.

Таблиця 5.1 – Варіанти завдань

Номер варіанта	int …Pin	ledDelay	delay	delay
01	8,3,5	10000	1000	6000
02	2,7,1	500	2000	5000
03	6,4,9	100	3000	4000
04	5,3,12	1000	4000	3000
05	11,9,4	3800	5000	2000
06	10,6,1	5000	6000	1000

ЛАБОРАТОРНА РОБОТА № 6

Біжучий вогонь

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування. Розробка власного біжучого рядка.

Короткі теоретичні відомості

У цій лабораторній роботі ми продовжимо роботу зі світлодіодами, але кількість світлодіодів збільшимо до 5, а також зробимо ефект біжучого вогню (рис. 6.1, рис. 6.2). Для управління світлодіодами будемо використовувати маніпуляції з портами Arduino. Ми будемо напряму записувати дані в порти Arduino. Це краще, ніж працювати з конкретними входами / виходами контролера. Це дозволить встановити значення для світлодіодів за допомогою однієї лише операції.

У Arduino UNO є 3 порти:

- B (цифрові входи / виходи з 8 по 13);
- C (аналогові входи);
- D (цифрові входи / виходи з 0 по 7).

Кожен порт керується 3 регістрами. Регістр DDR визначає, буде вивід (pin) входом чи виходом. За допомогою регістра PORT можна встановити вивід в стан HIGH або LOW.

Ми будемо використовувати порт B. Спочатку ми повинні встановити всі виводи порту B як цифрові виходи. У порту B є тільки 6 ніжок. Біти регістра для B-порту DDRB повинні бути встановлені в 1, якщо вивід буде використовуватися як вихід (OUTPUT), і в 0, якщо ніжка буде використовувати як вхід (INPUT). Біти портів нумеруються з 0 по 7, але не завжди містять усі 8 виводів.

Приклад:

```
DDRB = B00111110; // Встановити виводи порту B
                  // з 1 по 5 як виходи,
                  // а 0 як вхід.
```

Зверніть увагу, що в мікроконтролерах фірми Microchip все навпаки. 0 біт – вивід працює як вихід, а 1 – як вхід.

У нашому проекті біжучого вогню ми будемо використовувати 5 виходів:

```
DDRB = B00011111; / / Встановити виводи порту B з 0
по 4 як виходи
```

Для записування значень у порт B необхідно використовувати регістр PORTB. Засвітити перший світлодіод можна командою:

```
PORTB = B00000001;
```

перший і четвертий:

```
PORTB = B00001001;
```

Тепер розглянемо оператори зсуву. Є 2 оператори двійкового зсуву: оператор зсуву вліво << і оператор зсуву вправо >>. Оператор зсуву вліво << змушує всі біти зсуватися вліво, відповідно оператор зсуву вправо >> зсуває біти вправо.

Приклад:

```
varA = 1;       // 00000001
varA = 1 << 0; // 00000001
varA = 1 << 1; // 00000010
varA = 1 << 2; // 00000100
```

Тепер повернемося до нашої програми, яка показана на рис. 6.3. Нам потрібно ввести 2 змінні: перша *upDown* міститиме значення, куди рухатися – вгору або вниз, а друга *cylon*, які світлодіоди засвітити.

У функції `setup()` ми визначаємо, які ніжки повинні працювати як виходи.

У головному циклі програми `loop()` світлодіоди по черзі загоряються знизу вгору шляхом збільшення змінної `cylon`, а коли доходить до верхнього, то змінній `upDown` присвоюється 0 і світлодіоди загоряються зверху вниз по черзі [3, 5, 6].

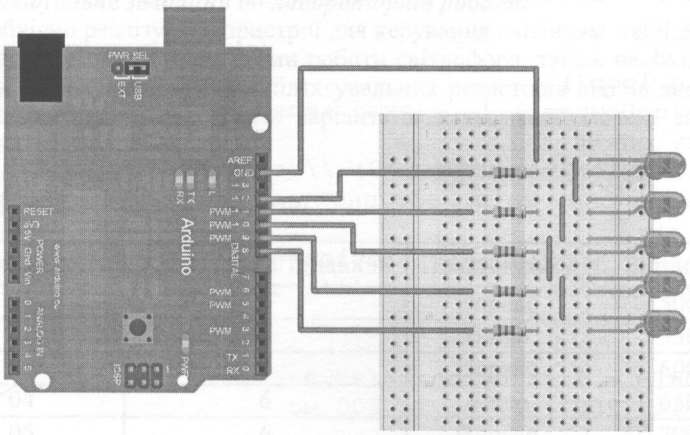


Рисунок 6.1 – Підключення та зовнішній вигляд макета

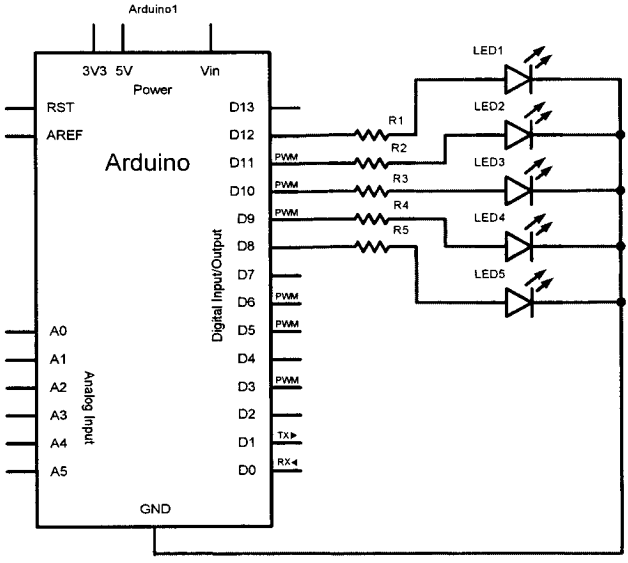


Рисунок 6.2 – Електрична принципова схема макета

```
// Біжучі вогні
unsigned char upDown=1;// розпочинаємо з «руху» вверх
unsigned char cylon=0; // визначаємо черговість засвічування світлодіодів
void setup(){
 DDRB = B00011111; // налаштовуємо порт B з 0 по 4 як виходи
}
void loop() {
 if(upDown==1) { // якщо «йдемо» вгору, то
  cylon++;
  if(cylon>=4) upDown=0; // якщо досягли найбільшого номера LED, то в наступному циклі «йдемо» вниз
    }
 else    { cylon--; if(cylon==0) upDown=1; // якщо досягли найменшого номера LED, то в наступному циклі йдемо вверх
  }
 PORTB = 1 << cylon; //зсув
 delay(200); // пауза 200 мс
}
```

Рисунок 6.3 – Лістинг програми

Необхідні комплектуючі:

– шість світлодіодів червоного кольору;
– шість резисторів 1,5 кОм і один резистор 4,7 кОм;
– сім гнучких монтажних провідників довжиною приблизно по 10 см;
– п'ять гнучких монтажних провідників завдовжки приблизно по 5 см.

Хід лабораторної роботи

1. Моделювання біжучого вогню:
- скласти макет, зображений на рис. 6.1;
- підключити макет до ПК і запустити програму Arduino IDE;
- набрати лістинг програми, поданий на рис. 6.3;
- перевірити формування файлу, який буде записаний на мікроконтролер;
- записати файл на мікроконтролер;
- перевірити функціональність пристрою.

2. Дослідження біжучого вогню за індивідуальним завданням:
- скласти макет, зображений на рис. 6.1;
- набрати лістинг програми для свого варіанта;
- перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та схема електрична принципова пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої для керування світінням зовнішніх світлодіодів з використанням схеми роботи світлофора, також необхідно попередньо обрахувати номінал підтягувальних резисторів відповідно до індивідуального завдання. Номер варіанта задається викладачем, завдання необхідно взяти з табл. 6.1.

Таблиця 6.1 – Варіанти завдань

Номер варіанта	Кількість світлодіодів	Початок руху	Затримка
01	6	Вправо	500
02	6	Вліво	550
03	6	Вправо	600
04	6	Вліво	650
05	6	Вправо	700
06	6	Вліво	750

ЛАБОРАТОРНА РОБОТА № 7

Сенсорний датчик

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також придбання навичок програмування з використанням команд налаштування порту. Реалізація сенсорного датчика.

Короткі теоретичні відомості

Існують пристрої, що реагують на дотик пальця. При торканні поверхні без натискання при цьому на механічний перемикач або кнопку, пристрій вмикається або вимикається, звук стає голоснішим або тихішим – залежно від функції. Сенсорні датчики широко застосовуються для вмикання, перемикання або регулювання. Такі сенсорні датчики можна запрограмувати за допомогою аналогових входів. Аналогові входи є високоомними, що і дозволяє вимірювати значення (напруги) при простому контакті пальцями.

Якщо в нашому експерименті злегка торкнутися пальцями аналогового входу, то під час виконання програми засвітиться світлодіод, а при наступному дотику – згасне. Тут значення порога включення становить 50, а відключення – 5. Схема монтажу наведена на рис. 7.1, а код програми – на рис. 7.2 [2].

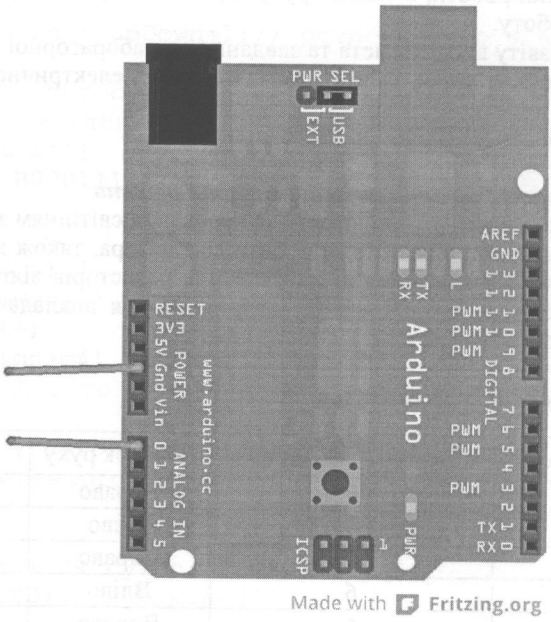


Рисунок 7.1 – Підключення та зовнішній вигляд сенсорного датчика

```
// Franzis Arduino
// Сенсорная кнопка
int LED=13;
int Sensor=0;
int Flag1,Flag2,tog=0;
void setup()
{
  pinMode(LED,OUTPUT);
}
void loop()
{
 if((analogRead(Sensor)>50)&&(!Flag2))
 {
   delay(50);
   if((analogRead(Sensor)>50)&&(!Flag2))
   {
     if(!tog)tog=1;else tog=0;
     Flag1=tog;
     Flag2=1;
   }
 }
 else
 {
   if((analogRead(Sensor)<5)&&(Flag2))
   {
     delay(50);
     if((analogRead(Sensor)<5)&&(Flag2))
     {
       Flag2=0;
     }
   }
 }

 if(!Flag1)digitalWrite(LED,LOW);
 if(Flag1)digitalWrite(LED,HIGH);
}
```

Рисунок 7.2 – Лістинг програми

Необхідні комплектуючі деталі:

– плата мікроконтролера Arduino / Freeduino;
– два гнучких монтажних провідники.

Хід лабораторної роботи

1. Дослідження роботи пристрою з освоєнням програми:

- скласти макет, зображений на рис. 7.1;
- підключити макет до ПК;
- запустити програму Arduino IDE;
- набрати лістинг програми, поданий на рис. 7.2;
- перевірити формування файлу, який буде записаний на мікроконтролер;
- записати файл на мікроконтролер;
- перевірити функціональність пристрою.

2. Замінити вхідний аналоговий порт.
3. Замінити вихідний порт інформації.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої з керуванням за допомогою сенсорної кнопки відповідно до індивідуального завдання.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 7.1.

Таблиця 7.1 – Варіанти завдань

Номер варіанта	Аналоговий вихід	Чутливість
01	1	мінімальна
02	2	максимальна
03	3	мінімальна
04	4	максимальна
05	5	мінімальна
06	1	максимальна

ЛАБОРАТОРНА РОБОТА № 8

Аналоговий вихід з широтно-імпульсною модуляцією

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також придбання початкових навичок програмування з використанням аналогового виходу з широтно-імпульсною модуляцією (ШІМ).

Короткі теоретичні відомості

На платі Arduino знаходяться в розпорядженні шість виходів сигналів з ШІМ: контакти 3, 5, 6, 9, 10 і 11 (на попередніх контролерах ATmega8 тільки контакти 9, 10 і 11). Вони можуть використовуватися для цифро-аналогового перетворення, управління кроковим двигуном або для формування звукових сигналів. У процесі ШІМ (PWM – Pulse Width Modulation) змінюється шпаруватість імпульсної послідовності. Шпаруватість показує співвідношення тривалості включеного стану до періоду повторювання імпульсів. При цьому частота і рівень сигналу залишаються завжди однаковими. Змінюється тільки тривалість переходу від високого до низького рівня (рис. 8.1–8.3).

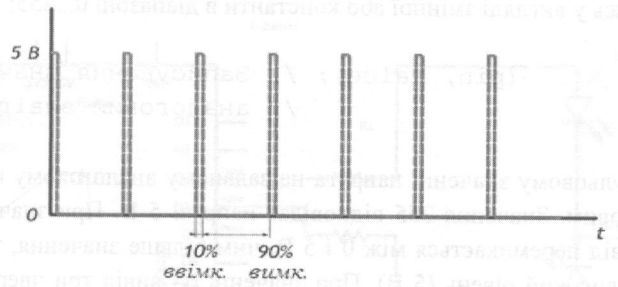


Рисунок 8.1 – Шпаруватість ШІМ 10%

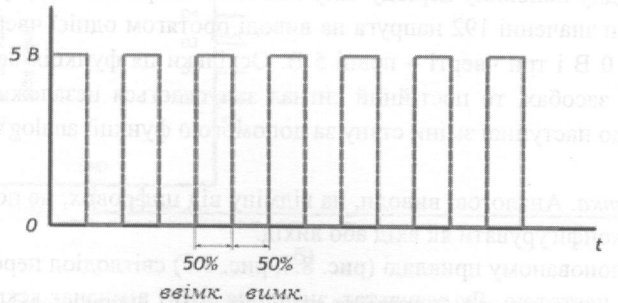


Рисунок 8.2 – Шпаруватість ШІМ 50%

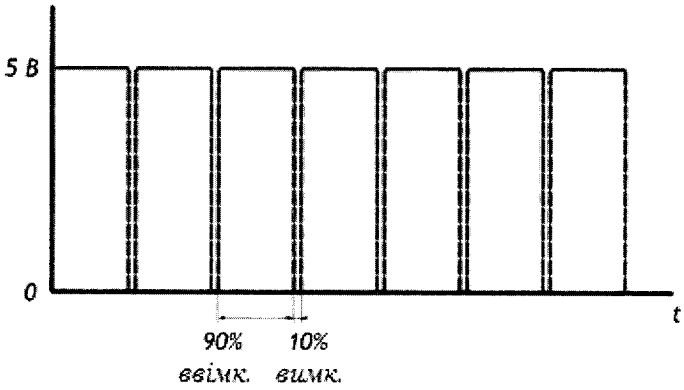


Рисунок 8.3 – Шпаруватість ШІМ 90%

```
analogWrite(pin, value)
```

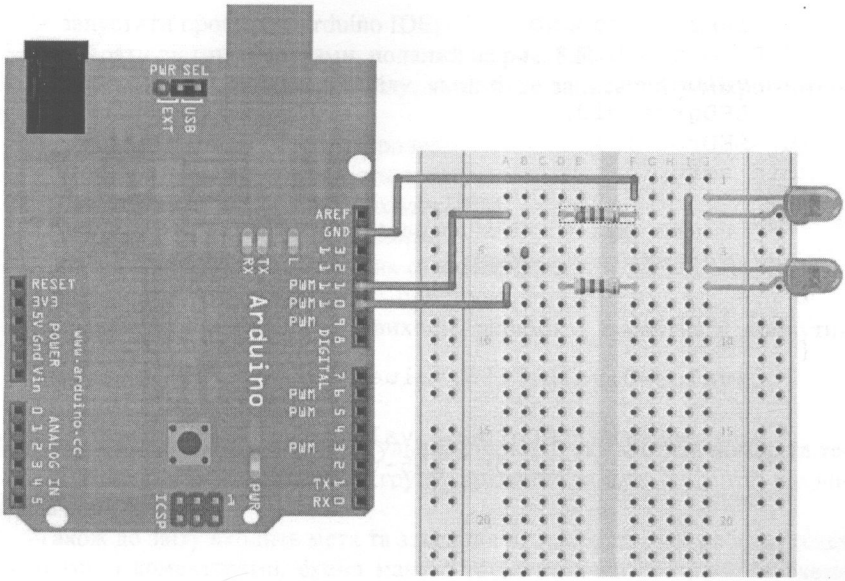
Ця команда формує на виході псевдоаналогові значення за допомогою модуляції ширини імпульсної послідовності (ШІМ). Значення може встановлюватись у вигляді змінної або константи в діапазоні 0...255:

```
analogWrite(pin, value); // Записування значення в
                         // аналоговий вивід
```

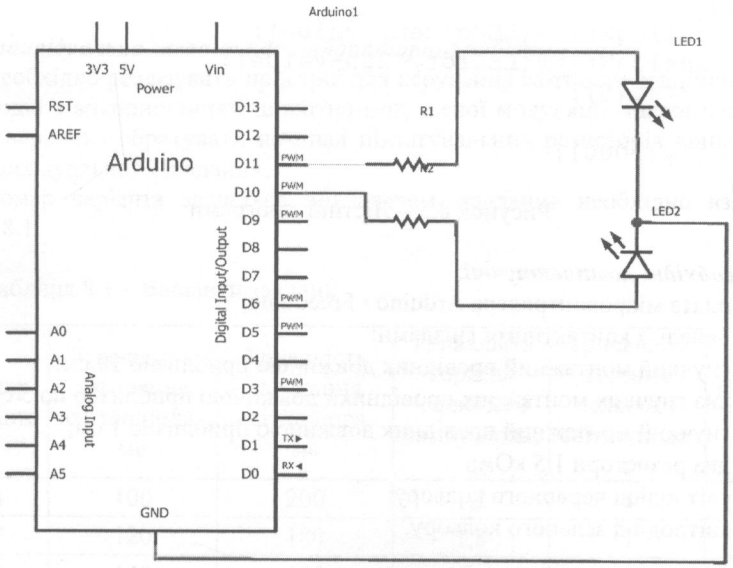
При нульовому значенні напруга на заданому аналоговому виході теж буде нульовим. Значення 255 відповідає напрузі 5 В. При значеннях між 0 і 255 вивід перемикається між 0 і 5 В, чим більше значення, тим довше вивід має високий рівень (5 В). При значенні 64 вивід три чверті періоду має 0 В і одну чверть 5 В. Значення 128 призводить до того, що вихідна напруга одну половину періоду часу має високий рівень і другу – низький рівень. При значенні 192 напруга на виводі протягом однієї чверті періоду становить 0 В і три чверті – повні 5 В. Оскільки ця функція базується на апаратних засобах, то постійний сигнал запускається незалежно від програми аж до наступної зміни стану за допомогою функції analogWrite.

Примітка. Аналогові виводи, на відміну від цифрових, не потрібно попередньо конфігурувати як вхід або вихід.

У пропонованому прикладі (рис. 8.4, рис. 8.5) світлодіод перемикається з високою частотою. Як результат, значення ШІМ визначає яскравість світіння [2].



а)



б)

Рисунок 8.4 – Підключення та зовнішній вигляд (а)
і електрична принципова схема макета (б)

```
// Francis Arduino
// Робота з ШІМ
int value;
int LEDgruen=10;
int LEDrot=11;
void setup()
{
  // В цей раз ніяких додаткових
  // налаштувань немає
}
void loop()
{
 for(value=0;value<255;value++)
 {
   analogWrite(LEDgruen, value);
   analogWrite(LEDrot, 255-value);
   delay(5);
 }
 delay(1000);

 for(value=255;value!=0;value--)
 {
   analogWrite(LEDgruen, value);
   analogWrite(LEDrot, 255-value);
   delay(5);
 }
 delay(1000);
 }
```

Рисунок 8.5 – Лістинг програми

Необхідні комплектуючі:

– плата мікроконтролера Arduino / Freeduino;
– панель з контактними гніздами;
– гнучкий монтажний провідник довжиною приблизно 10 см;
– два гнучких монтажних провідники довжиною приблизно по 5 см;
– гнучкий монтажний провідник довжиною приблизно 1 см;
– два резистори 1,5 кОм;
– світлодіод червоного кольору;
– світлодіод зеленого кольору.

Хід лабораторної роботи

1. Моделювання аналогових виходів ШІМ:
- скласти макет, зображений на рис. 8.4;
- підключити макет до ПК;

- запустити програму Arduino IDE;
- набрати лістинг програми, поданий на рис. 8.5;
- перевірити формування файлу, який буде записаний у мікроконтролер;
- записати файл на мікроконтролер;
- перевірити функціональність пристрою.

2. Дослідження аналогових виходів ШІМ:
- скласти схему для свого варіанта;
- набрати лістинг програми для свого варіанта;
- перевірити функціональність пристрою;
- підключити до одного із виходів динамік і спробувати «почути» ШІМ-сигнал.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої для керування світінням зовнішніх світлодіодів з використанням широтно-імпульсної модуляції, також необхідно попередньо обрахувати номінал підтягувальних резисторів відповідно до індивідуального завдання.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 8.1.

Таблиця 8.1 – Варіанти завдань

Номер варіанта	Тривалість загоряння світлодіода, мс	Тривалість затухання світлодіода, мс	Тривалість горіння червоного світлодіода, с	Тривалість горіння зеленого світлодіода, с	Порти ШІМ
01	100	200	1,2	4	3,5
02	120	180	1,8	3	6,9
03	140	160	2	2,5	10,5
04	160	140	2,5	2	11,3
05	180	120	3	1,8	3,9
06	200	100	4	1,2	11,6

ЛАБОРАТОРНА РОБОТА № 9

Регулятор рівня яскравості світлодіода з транзистором

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування з використанням транзисторів та ШІМ-сигналу.

Короткі теоретичні відомості

У лабораторній роботі № 8 ви вже познайомилися з формуванням на аналоговому виході плати Arduino ШІМ-сигналу і тепер можна скласти регулятор яскравості світіння світлодіода (рис. 9.1, рис. 9.2).

```
pinMode(pin, mode)
```

Функція використовується в програмі `void setup()`, щоб конфігурувати контакт плати як вхід або як вихід:

```
pinMode(pin, OUTPUT); // Вивід встановлюється
                      // як вихід
digitalRead(pin)
```

Функція `digitalRead()` зчитує значення заданого цифрового виводу з результатом `HIGH` або `LOW` , що відповідає 1 або 0. Номер виводу може встановлюватися або як змінна, або як константа (від 0 до 13).

```
value = digitalRead(Pin); // Встановлює value,
                          // рівне значенню
                          // на вхідному виводі
digitalWrite(pin , value)
```

Встановлює рівень `HIGH` або `LOW` на заданому виводі. Номер виводу може задаватися або як змінна , або як константа (від 0 до 13).

```
digitalWrite(SW1); // Зчитування стану кнопки
analogWrite(pin, value)
```

Ця команда формує на виході псевдоаналогові значення за допомогою модуляції ширини імпульсної послідовності (ШІМ). Значення може встановлюватися у вигляді змінної або константи в діапазоні 0...255.

```
analogWrite(pin, value); // Записування значення в
                         // аналоговий вивід
```

У пропонованому експерименті червоний світлодіод можна безпосередньо підключити до аналогового виходу (контакт 3). Якщо взяти більш яскраві світлодіоди, наприклад, компанії Luxeon, то потрібно додати транзистор як підсилювач. За допомогою кнопок S1 («світліше») і S2 («темніше») можна змінювати яскравість світлодіода [2]. Лістинг програми наведено на рис. 9.3.

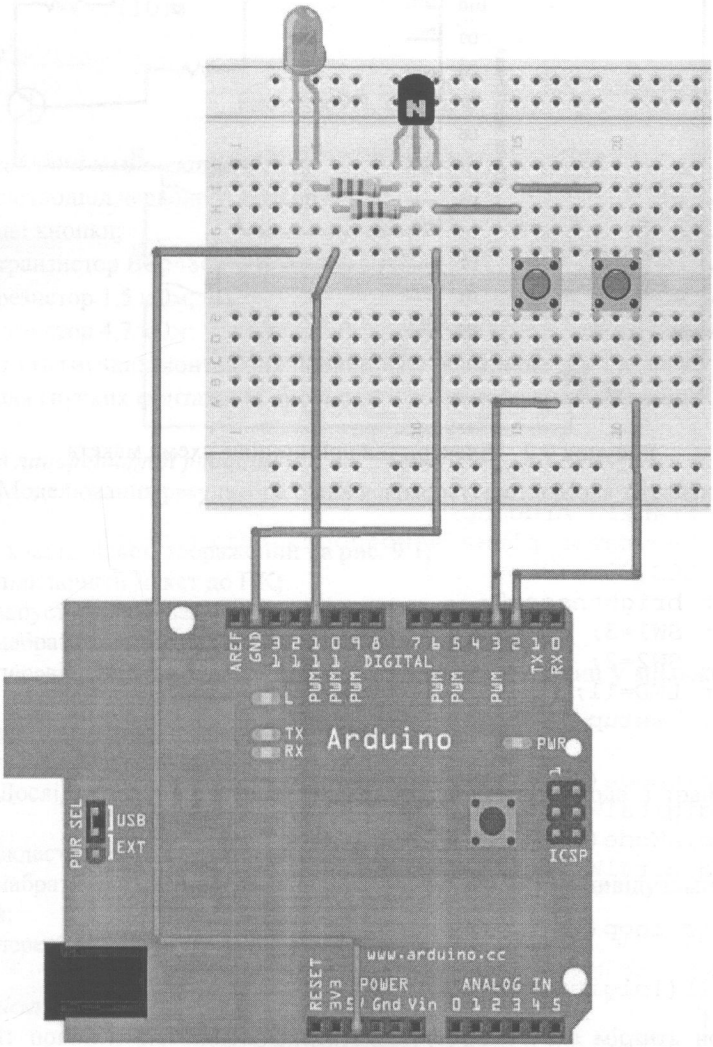


Рисунок 9.1 – Підключення та зовнішній вигляд макета

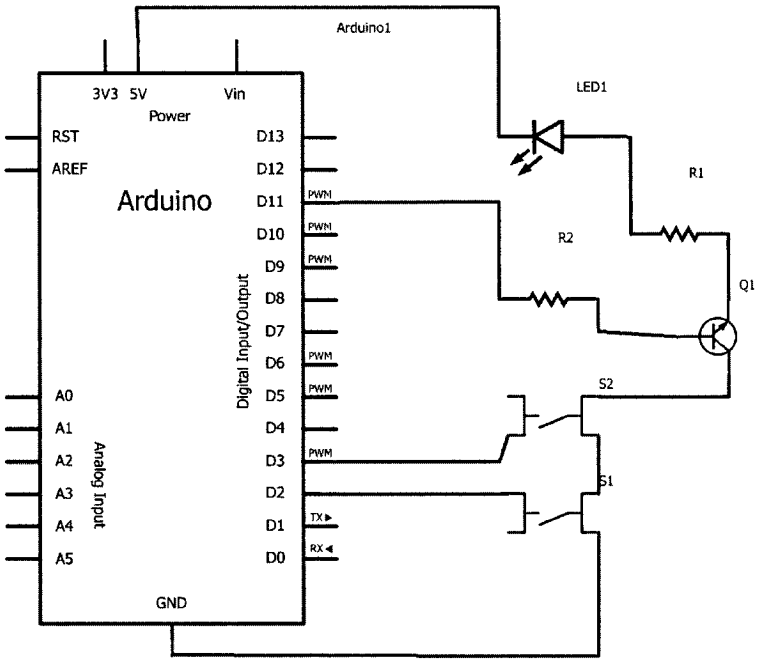


Рисунок 9.2 – Електрична принципова схема макета

```
// Franzis Arduino
// Регулятор рівня яскравості світлодіода
// LED Dimmer
int brightness=0;
int SW1=3;
int SW2=2;
int LED=11;
void setup()
{
  pinMode(SW1,INPUT);
  digitalWrite(SW1,HIGH);
  pinMode(SW2,INPUT);
  digitalWrite(SW2,HIGH);
}
void loop()
{
  if(!digitalRead(SW1)&&digitalRead(SW2))
  {
    if(brightness<255)brightness++;
```

Рисунок 9.3 – Лістинг програми

```
    analogWrite(LED,brightness);
    delay(10);
  }
  else if(digitalRead(SW1)&&!digitalRead(SW2))
  {
    if(brightness!=0)brightness--;
    analogWrite(LED,brightness);
    delay(10);
  }
}
```

Рисунок 9.3, аркуш 2

Необхідні комплектуючі:

– світлодіод червоного кольору;
– дві кнопки;
– транзистор BC548C;
– резистор 1,5 кОм;
– резистор 4,7 кОм;
– п'ять гнучких монтажних провідників довжиною приблизно по 10 см;
– два гнучких монтажних провідники довжиною приблизно по 5 см.

Хід лабораторної роботи

1. Моделювання регулятора рівня яскравості світлодіода з транзистором:

- скласти макет, зображений на рис. 9.1;
- підключити макет до ПК;
- запустити програму Arduino IDE;
- набрати лістинг програми. поданий на рис. 9.2;
- перевірити формування файлу, який буде записаний у мікроконтролер;
- записати файл на мікроконтролер;
- перевірити функціональність пристрою.

2. Дослідження регулятора рівня яскравості світлодіода з транзистором:

- скласти макет, поданий на рис. 9.1;
- набрати лістинг програми відповідно до варіанта індивідуального завдання;
- перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер і тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої для керування світінням зовнішніх світлодіодів з використанням широтно-імпульсної модуляції та керуванням рівня за допомогою транзистора, також необхідно попередньо обраховувати номінал підтягувальних резисторів відповідно до індивідуального завдання.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 9.1.

Таблиця 9.1 – Варіанти завдань

Номер варіанта	Крок збільшення яскравості	Крок зменшення яскравості	Порти входу / виходу
01	12	60	9, 7, 6
02	15	75	6, 5, 4
03	18	90	5, 6, 2
04	21	105	3, 5, 7
05	24	220	10, 3, 5
06	27	235	6, 7, 5

ЛАБОРАТОРНА РОБОТА № 10

Робота із зовнішніми пристроями формування звукових сигналів

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування. Ознайомлення з зовнішніми пристроями формування звукових сигналів, реалізація власної музичної композиції.

Короткі теоретичні відомості

Arduino може бути «музикальним». Щоб дізнатися наскільки це так, приєднаємо п'єзоакустичний перетворювач до експериментальної плати, як показано на рис. 10.1.

З'єднаємо п'єзоакустичний перетворювач з цифровим виходом 11 і GND. Для генерації звуків Arduino пропонує нам команду `tone()`. З її допомогою можна формувати звуковий сигнал з будь-якою частотою виведення. Генерація частоти проходить чисто програмним шляхом (рис. 10.2).

У команді є ще різні параметри: номер виводу, частота і тривалість звукового сигналу:

```
tone(pin, frequency); // Зупиняє безперервний тон
tone(pin, frequency, duration) // Звук певної тривалості
```

Наприклад:

```
// Franzis Arduino
// Звук
int Speaker=8;
void setup()
{
   pinMode(Speaker, OUTPUT);
}
void loop()
{
   tone(Speaker,550,450);
   delay(3000);
}
```

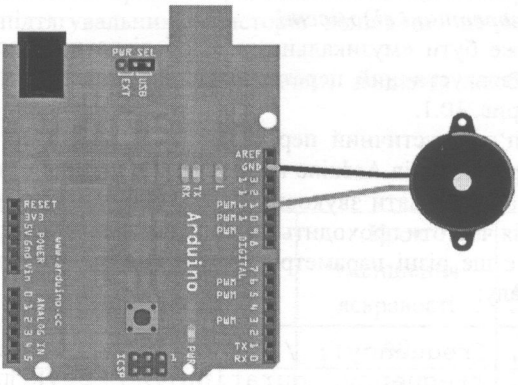
Команда `tone()` формує тільки прості звуки. Як можна створити власну програму для генерації мелодій, пояснює лістинг (рис. 10.2). Тут звуки генеруються перемиканням цифрового виводу між високим (HIGH) і низьким (LOW) рівнем:

```
digitalWrite(Speaker, HIGH);
delayMicroseconds(tone);
```

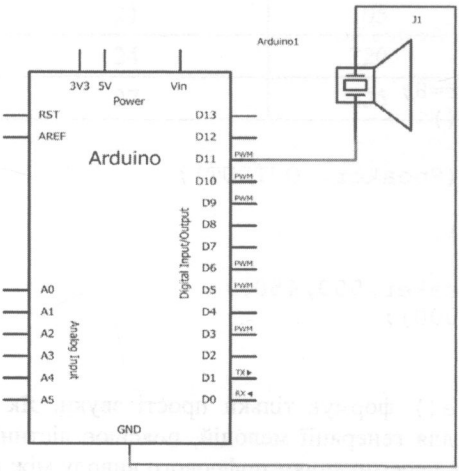
```
digitalWrite(Speaker, LOW);
delayMicroseconds(tone);
```

При запуску коду програми (рис. 10.2) на цифровому виході формується мелодія. Період повторення імпульсів задається у функції `tone()`.

Якщо приробити паперовий рупор до п'єзоакустичного перетворювача, то звук стане значно голоснішим. Навіть якщо просто приклеїти до гучномовця невеликий шматок паперу, гучність збільшиться [2].



а)



б)

Рисунок 10.1 – Підключення та зовнішній вигляд (а) і електрична принципова схема макета (б)

```
// Franzis Arduino
// Мелодія
int Speaker = 8;
int length = 15;
char notes[] = "ccggaagffeeddc ";
int beats[] = { 1,1,1,1,1,1,2,1,1,1,1,1,1,2,4 };
int tempo = 300;
void setup()
{
  pinMode(Speaker, OUTPUT);
}
void loop()
{
  for (int i = 0; i < length; i++)
  {
    if (notes[i] == ' ')
    {
      delay(beats[i] * tempo);
    }
    else
    {
      playNote(notes[i], beats[i] * tempo);
    }
       delay(tempo / 2);
  }
}
void playTone(int tone, int duration)
{
  for (long i = 0; i < duration*1000L; i += tone * 2)
  {
    digitalWrite(Speaker, HIGH);
    delayMicroseconds(tone);
    digitalWrite(Speaker, LOW);
    delayMicroseconds(tone);
  }
}
void playNote(char note, int duration)
{
  char names[] = { 'c','d','e','f','g','a','b','C' };
  int tones[] = {1915,1700,1519,1432,1275,1136,1014,956};
  for (int i = 0; i < 8; i++)
  {
    if (names[i] == note)
    {
      playTone(tones[i], duration);
    }
  }
}
```

Рисунок 10.2 – Лістинг програми

Хід лабораторної роботи

1. Робота з пристроями формування звукових сигналів:
- скласти макет, зображений на рис. 10.1;
- підключити макет до ПК;
- запустити програму Arduino IDE;
- набрати лістинг програми, поданий на рис. 10.2;
- перевірити формування файлу, який буде записаний на мікроконтролер;
- записати файл у мікроконтролер;
- перевірити функціональність пристрою.

2. Створення власного проекту з використанням пристроїв формування звукових сигналів:
- скласти макет, зображений на рис. 10.1;
- набрати лістинг програми своєї мелодії;
- перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої для керування зовнішніми пристроями формування звукових сигналів відповідно до індивідуального завдання.

Індивідуальне завдання видається викладачем.

ЛАБОРАТОРНА РОБОТА № 11

Азбука Морзе

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування. Реалізація пристрою формування повідомлення з використанням азбуки Морзе.

Короткі теоретичні відомості

Для цього проекту ви будете використовувати код, що дозволить сформувати світлодіодні сигнали, які відображають літери азбукою Морзе. Азбука Морзе є способом кодування тексту, який передає букви і цифри, використовуючи шаблони, що містять довгі (тире) та короткі (крапка) сигнали. Тобто, використовуючи дані шаблони, цілком можливим є формування літер та цифр за допомогою світлодіода.

Наприклад для передачі сигналу SOS потрібно реалізувати три коротких спалахи (Dits), а потім три довгих (Dahs), а потім знову три коротких спалахи.

Для передачі світлодіодом сигналу SOS, використовуйте код, що поданий на рис. 11.1.

```
// Сигнал SOS
int ledPin = 12;
void setup()
{
  pinMode(ledPin, OUTPUT);
}
void loop()
{
  for (int x=0; x<3; x++) {
    digitalWrite(ledPin, HIGH);
    delay(150);
    digitalWrite(ledPin, LOW);
    delay(100);
}
  delay(100);
  for (int x=0; x<3; x++) {
    digitalWrite(ledPin, HIGH);
    delay(400);
    digitalWrite(ledPin, LOW);
  delay(100);
}
  delay(100);
```

Рисунок 11.1 – Лістинг програми

```
  for (int x=0; x<3; x++) {
    digitalWrite(ledPin, HIGH);
    delay(150);
    digitalWrite(ledPin, LOW);
  delay(100);
}
  delay(5000);
}
```

Рисунок 11.1, аркуш 2

Створіть новий ескіз, а потім введіть код із рис. 11.1. Переконайтеся, що ваш код не містить помилок, а потім завантажте його на свій Arduino (рис. 11.2). Якщо все добре, ви побачите індикатор, що відображає сигнал азбуки Морзе SOS, після паузи 5 секунд він повториться.

Проаналізуємо, яким чином працює цей код. Перша частина коду вже відома, де ініціалізується змінна, а потім налаштовується цифровий вивід 12, який буде портом виведення (виходом). У головному коді циклу ви можете побачити такі ж заяви, щоб вмикати і вимикати світлодіоди протягом встановленого періоду часу.

Далі розглянемо три блоки коду, які є однотипними.

Перший блок виводить три крапки:

```
  for (int x=0; x<3; x++) {
    digitalWrite(ledPin, HIGH);
    delay(150);
    digitalWrite(ledPin, LOW);
    delay(100);
}
```

Ви можете бачити, що світлодіод вмикається на 150 мс, а потім вимикається на 100 мс. Ці команди знаходяться у фігурних дужках, тобто в окремому блоці коду. Але, коли завантажується програма, можна побачити спалахи світла в три рази, а не один раз.

Це робиться за допомогою циклу:

```
  for (int x=0; x<3; x++) {
```

Ця команда у лістингу виконується тричі. Є три параметри, які необхідно задати для циклу: ініціалізація, умова і приріст. Ініціалізація відбувається вперше і точно один раз. Кожен раз через цикл умова перевіряється; якщо це правда, то блок операторів і приріст виконується, потім умова перевіряється знову. Коли умова стає помилковою, цикл закінчується.

Отже, спочатку необхідно ініціалізувати змінну як початкове число циклу. У цьому випадку створено змінну x і встановлено її значення у нуль.

```
int x=0;
```

Потім поставили умову, щоб вирішити, скільки разів код в циклі буде виконуватися:

```
x<3;
```

У цьому випадку код буде циклічно виконуватись, якщо x менше, ніж (<) 3.

Символ < відомий як один із операторів порівняння, які використовуються для прийняття рішень в коді і для порівняння двох значень. Ось ці знаки:

```
==   (дорівнює)
!=   (не дорівнює)
<    (менше)
>    (більше)
<=   (менше або дорівнює)
>=   (більше або дорівнює)
```

У коді порівнюється x зі значенням 3, щоб побачити, чи він менший за 3. Якщо x менший трьох, код у блоці буде повторюватися знову.

У кінцевому параметрі x++ виконує збільшення значення x на 1. Крім того, можна було записати x = x + 1, що б присвоїти x значення x + 1.

Примітка. Немає необхідності ставити крапку з комою після кінцевої заяви в циклі.

Можна зробити прості математичні операції за допомогою символів «+», «-», «*» і «/» (додавання, віднімання, множення і ділення). Наприклад:

```
1 + 1 = 2
3 - 2 = 1
2 * 4 = 8
8 / 2 = 4
```

Таким чином, код у петлі ініціалізує значення x = 0, після чого виконується код у блоці (фігурні дужки). Потім збільшується приріст (в даному випадку, додається 1 до x) і відбувається перевірка, чи x менший за 3, і якщо так, то дія повторюється.

Тобто є три цикли в коді: два цикли відображають три крапки та один цикл – три тире.

Слід зазначити, що змінна x є локальною, а це означає, що змінна є дійсною тільки в межах власного блока коду, якщо ви не ініціалізували її перед `void setup()`, в цьому випадку вона є глобальною і може бути видимою по всій програмі. Якщо ви намагаєтеся отримати доступ до x поза циклом, ви отримаєте повідомлення про помилку [3].

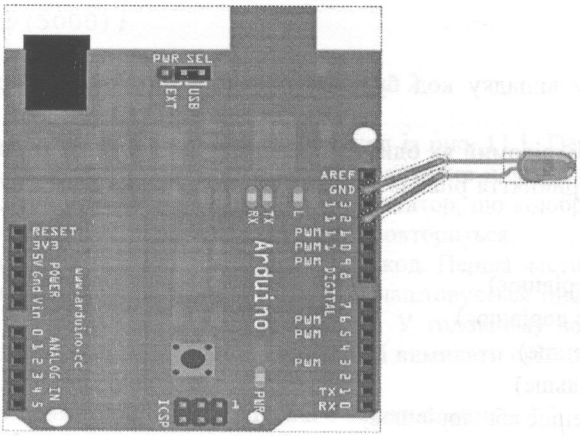


Рисунок 11.2 – Схема макета

Символ	Код	Символ	Код	Символ	Код
А	. –	С	. . .	Ь	– . . –
Б	– . . .	Т	–	Ю	. . – –
В	. – –	У	. . –	Я	. – . –
Г	. . . .	И	– . – –		
Ґ	– – .	І	. .	1	. – – – –
Д	– . .	Ї	. – – – .	2	. . – – –
Е	.	Й	. – – –	3	. . . – –
Є	. . – . .	К	– . –	4	. . . . –
Ж	. . . –	Л	. – . .	5	. . . . .
З	– – . .	Ф	. . – .	6	– . . . .
М	– –	Х	– – – –	7	– – . . .
Н	– .	Ц	– . – .	8	– – – . .
О	– – –	Ч	– – – .	9	– – – – .
П	. – – .	Ш	– – . –	0	– – – – –
Р	. – .	Щ	– – . – –		

Рисунок 11.3 – Азбука Морзе

Для прикладу розглянемо лістинг програми для літер «З» і «Ц».

```
// Символ З
int ledPin = 12;
void setup()
{
    pinMode(ledPin, OUTPUT);
  }
  void loop()
  {
    for (int x=0; x<2; x++) {
    digitalWrite(ledPin, HIGH);
    delay(400);
    digitalWrite(ledPin, LOW);
    delay(100);
  }
    delay(100);
    for (int x=0; x<2; x++) {
    digitalWrite(ledPin, HIGH);
    delay(150);
    digitalWrite(ledPin, LOW);
    delay(100);
  }
    delay(5000);
}
```

Рисунок 11.4 – Лістинг програми для літери «З»

```
// Символ Ц
int ledPin = 12;
void setup()
{
    pinMode(ledPin, OUTPUT);
  }
  void loop()
  {
    for (int x=0; x<2; x++) {
    digitalWrite(ledPin, HIGH);
    delay(400);
    digitalWrite(ledPin, LOW);
    delay(100);
    digitalWrite(ledPin, HIGH);
    delay(150);
    digitalWrite(ledPin, LOW);
    delay(100);
  }
    delay(5000);
}
```

Рисунок 11.5 – Лістинг програми для літери «Ц»

Необхідні комплектуючі:

– плата мікроконтролера Arduino / Freeduino;

– один світлодіод.

Хід лабораторної роботи

1. Моделювання пристрою формування сигналів з використанням азбуки Морзе:

- скласти макет, зображений на рис. 11.2;
- підключити макет до ПК;
- запустити програму Arduino-x.x.x;
- набрати лістинг програми, поданий на рис. 11.1;
- перевірити формування файлу, який буде записаний на мікроконтролер;
- записати файл на мікроконтролер;
- перевірити функціональність пристрою.

2. Дослідження власного пристрою формування сигналів з використанням азбуки Морзе:

- скласти макет, зображений на рис. 11.2;
- набрати лістинг програми, який відтворює ваші ініціали;
- перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрої для керування зовнішніми пристроями формування звукових сигналів відповідно до індивідуального завдання.

Індивідуальне завдання полягає у відображенні імені студента за допомогою азбуки Морзе.

ЛАБОРАТОРНА РОБОТА № 12

Годинник реального часу

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування. Навчитися самостійно створювати годинник з функцією таймера та можливістю давати дзвінки за розкладом.

Короткі теоретичні відомості

У багатьох додатках потрібен годинник з програмним управлінням (RTC – Real Time Clock). Це може бути просте реле часу, що спрацьовує в точно встановлений термін, або лічильник робочого часу. Виведення на термінал теж відбувається по секундах. Старе значення секунд завжди порівнюється з новим. Наш світлодіод L також блимає щосекунди. Завдяки йому можна спостерігати за функціонуванням програми, чи працює вона без збоїв або є помилка при програмуванні.

Але слід враховувати, що такі датчики часу не мають високої точності (похибка може досягати до 1 хвилини за добу), оскільки тактова частота і похибка кварцового резонатора набагато більша, ніж у прецизійному годинниковому кварці. До того ж точність відліку часу істотно залежить від коливань температури. Код програми наведений на рис. 12.1.

```
// Franzis Arduino
// Годинник

int cnt, Second, Minute, Hour=0;
int LED=13;

void setup()
{
  Serial.begin(9600);
  pinMode(LED,OUTPUT);

  // Завдання часу
  Second=13;
  Minute=10;
  Hour=0;
}

void loop()
{
  cnt++;
  if(cnt==50)digitalWrite(LED,LOW);
```

Рисунок 12.1 – Лістинг програми «Годинник реального часу»

```
    if(cnt==100)
  {
    digitalWrite(LED,HIGH);
    Serial.print(Hour);
    Serial.print(":");
    Serial.print(Minute);
    Serial.print(":");
    Serial.println(Second);
    Second++;
    if(Second==60)
    {
      Second=0;
      Minute++;
      if(Minute==60)
      {
        Minute=0;
        Hour++;
        if(Hour==24)
        {
          Hour=0;
        }
      }
    }
    cnt=0;
  }
  delay(10);
}
```

Рисунок 12.1, аркуш 2

Програма годинника

Одне із застосувань годинника – це, наприклад, подання дзвінка за розкладом. У певний час (кожні 45 хв) дзвенить дзвінок. Для цього потрібно додати в попередню програму годинника кілька рядків коду і запросити при цьому контроль часу через умову if. Якщо умова виконується, то звучить п'єзоакустичний перетворювач (рис. 12.2). Макет установки наведено на рис. 12.3.

Розклад занять в університеті такий: початок 1-го заняття о 8:15, тривалість заняття 45 хв. Перерва між першими шістьма заняттями 15 хв. О 14:00 велика перерва тривалістю 45 хв. Перерва між 7 та 12 заняттями 15 хв.

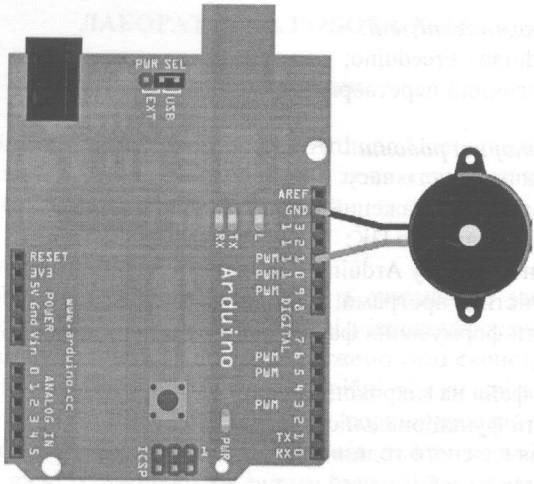
На початку і наприкінці кожної академічної години (45 хв) повинен звучати дзвінок. Дзвінок у нас подає п'єзоакустичний перетворювач [2].

```
// Franzis Arduino
// Розклад занять
int cnt, Sekunde, Minute, Stunde=0;
int LED=13;
int Speaker=11;
void setup()
{
  Serial.begin(9600);
  pinMode(LED,OUTPUT);
  pinMode(Speaker,OUTPUT);

  // Zeitvorgabe
  Stunde=6;
  Minute=59;
  Sekunde=58;
}
void loop()
{
  cnt++;
  if(cnt==50)digitalWrite(LED,LOW);

  if(cnt==100)
  {
    digitalWrite(LED,HIGH);
    Serial.print(Stunde);
    Serial.print(":");
    Serial.print(Minute);
    Serial.print(":");
    Serial.println(Sekunde);
    Sekunde++;
    if(Sekunde==60)
    {
      Sekunde=0;
      Minute++;
      if(Minute==60)
      {
        Minute=0;
        Stunde++;
        if(Stunde==24)
        {
          Stunde=0;
        }
      }
    }
```

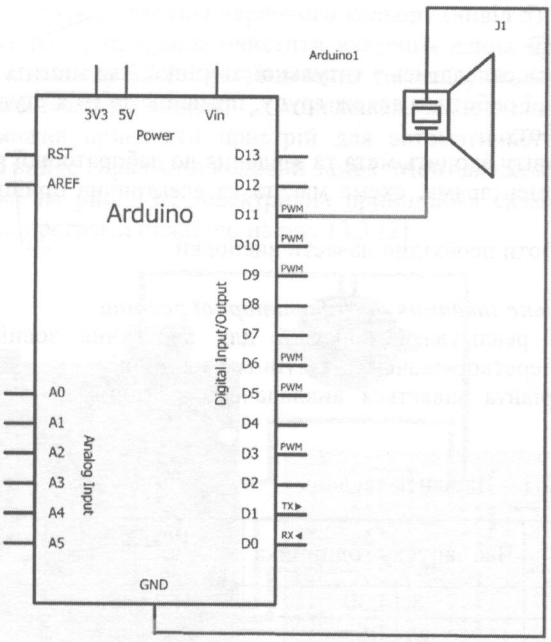
Рисунок 12.2 – Лістинг програми академічних годин

```
    cnt=0;
  }
  delay(10);

  //  Час дзвінка
    // 1. Заняття
  if(Stunde==7&&Minute==0)Bell();
  if(Stunde==7&&Minute==45)Bell();
  // 2. Заняття
  if(Stunde==7&&Minute==55)Bell();
  if(Stunde==8&&Minute==40)Bell();
  // Перерва
  // 3. Заняття
  if(Stunde==9&&Minute==0)Bell();
  if(Stunde==9&&Minute==45)Bell();
  // 4. Заняття
  if(Stunde==9&&Minute==55)Bell();
  if(Stunde==10&&Minute==40)Bell();
  // 5. Заняття
  if(Stunde==10&&Minute==50)Bell();
  if(Stunde==11&&Minute==35)Bell();
  // Обідня перерва
  // 6. Заняття
  if(Stunde==12&&Minute==05)Bell();
  if(Stunde==12&&Minute==50)Bell();
  // 7. Заняття
  if(Stunde==13&&Minute==0)Bell();
  if(Stunde==13&&Minute==45)Bell();

}
void Bell(void)
{
  if(Sekunde<5)
  {
    tone(Speaker,500);
  }
  else
  {
    noTone(Speaker);
  }
}
```

Рисунок 12.2, аркуш 2



а)



б)

Рисунок 12.3 – Підключення та зовнішній вигляд (а)
і електрична принципова схема макета (б)

Необхідні комплектуючі:

– плата Arduino / Freeduino;

– п'єзоакустичний перетворювач.

Хід лабораторної роботи

1. Годинник реального часу:

– скласти макет, зображений на рис. 12.3;

– підключити макет до ПК;

– запустити програму Arduino IDE;

– набрати лістинг програми, поданий на рис. 12.2;

– перевірити формування файлу, який буде записаний на мікроконтролер;

– записати файл на мікроконтролер;

– перевірити функціональність пристрою.

2. Реалізація власного годинника реального часу:

– скласти макет, зображений на рис. 12.3;

– набрати лістинг програми для свого варіанта;

– перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрій для керування зовнішнім п'єзоелектричним перетворювачем та світінням зовнішніх.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 12.1.

Таблиця 12.1 – Варіанти завдань

Номер варіанта	Час запуску годинника	Розклад занять	Тривалість сигналу
01	8:14:30	1, 2 урок	3 с
02	9:13:55	2, 3 урок	4 с
03	10:14:45	3, 4 урок	5 с
04	11:14:50	4, 5 урок	6 с
05	12:14:20	5, 6 урок	7 с
06	13:14:55	6, 7 урок	8 с

ЛАБОРАТОРНА РОБОТА № 13

Кодовий замок

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування. Розробка власного кодового замка.

Короткі теоретичні відомості

Справжній інженер-електронник замість звичайного замка неодмінно встановить у себе кодовий замок на основі мікроконтролера. Оскільки ми вже досвідчені програмісти Arduino, то можемо самі сконструювати замок з шифром. Для нашого замка будуть потрібні тільки дві кнопки: кнопка SW1 (вивід 2) і кнопка SW2 (вивід 3) на експериментальній платі. Для введення коду потрібно натискати, залежно від цифри коду, наприклад, на кнопку SW1 двічі і кнопку SW2 три рази. Натискання кнопок буде підтверджуватися за допомогою червоного світлодіода, підключеного до виводу 4, і звукового перетворювача. Якщо код введений правильно, то на 5 секунд вмикається світлодіод червоного кольору (вивід 5). Якщо код набраний з помилкою, то можна очистити введення даних більш тривалим натисканням на кнопку SW2. Про видалення сигналізують миготливий світлодіод, підключений до виводу 7, і «пищалка». Замість світлодіода через транзистор можна приєднати пристрій для автоматичного відкривання дверей, і тоді вийде справжній кодовий замок. Монтаж схеми кодового замка показаний на рис. 13.1 (електрична принципова схема показана на рис. 13.2). Код програми показано на рис 13.3 [2].

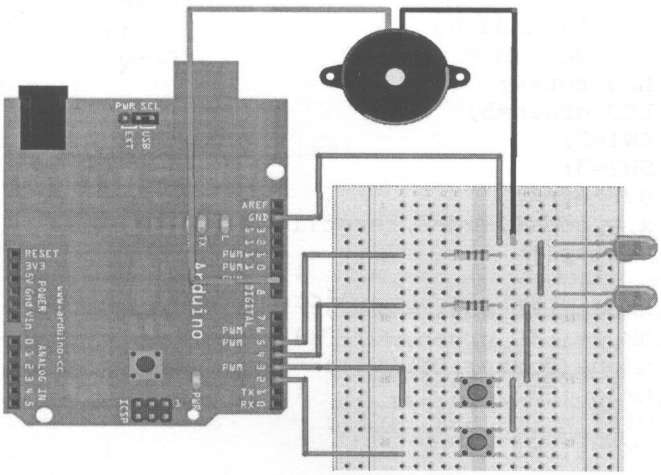


Рисунок 13.1 – Підключення та зовнішній вигляд макета

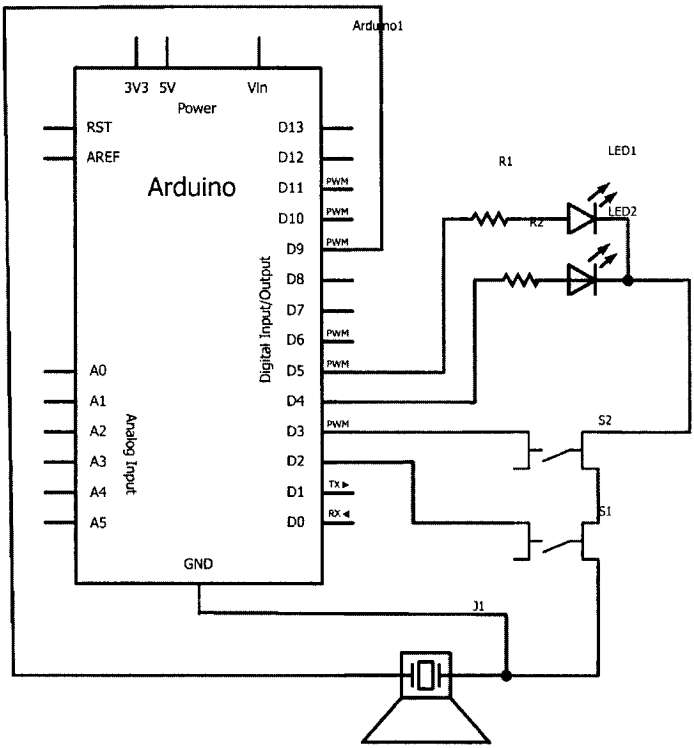


Рисунок 13.2 – Електрична принципова схема макета

```
// Franzis Arduino
// Кодовий замок
int LED_rot=4;
int LED_gruen=5;
int SW1=2;
int SW2=3;
int Buzzer=8;
int x,y,code1,code2,resetTimer=0;
void setup()
{
  pinMode(LED_rot,OUTPUT);
  pinMode(LED_gruen,OUTPUT);
  pinMode(Buzzer,OUTPUT);
  pinMode(SW1,INPUT);
  digitalWrite(SW1,HIGH);
  pinMode(SW2,INPUT);
```

Рисунок 13.2 – Лістинг програми

```
  digitalWrite(SW2,HIGH);
  Clr_Code();
}
void loop()
{
  // Code 1 = 5
    if(!digitalRead(SW1))
    {
      delay(50);
      if(!digitalRead(SW1))
      {
        Blink();
        x++;
        if(x==5)
        {
          code1=true;
        }else code1=false;
        do{
        }while(!digitalRead(SW1));
      }
    }
    // Code 2 = 3
    if(!digitalRead(SW2))
    {
      delay(50);
      if(!digitalRead(SW2))
      {
        Blink();
        y++;
        if(y==3)
        {
          code2=true;
        }else code2=false;
         do
        {
          delay(50);
          resetTimer++;
          if(resetTimer>50)
          {
            Toggle_Flash();
            Clr_Code();
            break;
          }
```

Рисунок 13.2, аркуш 2

```
        }while(!digitalRead(SW2));
        resetTimer=0;
      }
    }

    if(code1==true&&code2==true)
    {
      digitalWrite(LED_gruen,HIGH);
      Clr_Code();
      delay(5000);
      digitalWrite(LED_gruen,LOW);
    }
    else
    {
      digitalWrite(LED_gruen,LOW);
    }
}
void Blink(void)
{
  digitalWrite(LED_rot,HIGH);
  tone(Buzzer,500,150);
  delay(200);
  digitalWrite(LED_rot,LOW);
}
void Toggle_Flash(void)
{
  int tog=0;
  for(x=0;x<6;x++)
  {
    if(tog==0)tog=1;else tog=0;
    digitalWrite(LED_rot,tog);
    tone(Buzzer,500,250);
    delay(300);
  }
}
void Clr_Code(void)
{
  x=0;
  y=0;
  code1=0;
  code2=0;
  resetTimer=0;
  delay(1000);
}
```

Рисунок 13.2, аркуш 3

Необхідні комплектуючі:

– плата Arduino / Freeduino і панель з контактними гніздами;
– дві кнопки та два світлодіоди червоного і зеленого кольору;
– п'єзопретворювач звуку та два резистори по 1,5 кОм;
– сім гнучких монтажних провідників довжиною приблизно по 5 см;
– гнучкий монтажний провідник довжиною приблизно 10 см.

Хід лабораторної роботи

1. Реалізація кодового замка:
– скласти макет, зображений на рис. 13.1;
– підключити макет до ПК і запустити програму Arduino IDE;
– набрати лістинг програми, поданий на рис. 13.2;
– перевірити формування файлу, який буде записаний на мікроконтролер;
– записати файл на мікроконтролер;
– перевірити функціональність пристрою.
2. Реалізація власного кодового замка:
– скласти макет, зображений на рис. 13.1;
– набрати лістинг програми для свого варіанта;
– перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрій для реалізації функції кодового замка.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 13.1.

Таблиця 13.1 – Варіанти завдань

Номер варіанта	Код замка		Тон сигналу	Тривалість горіння зеленого світлодіода
	SW1	SW2		
01	2	6	За бажанням	6000
02	4	4	За бажанням	7000
03	5	3	За бажанням	8000
04	3	5	За бажанням	9000
05	1	7	За бажанням	10000
06	6	2	За бажанням	15000

ЛАБОРАТОРНА РОБОТА № 14

Підключення семисегментного індикатора

Мета роботи. Вивчення структури Arduino Uno, організації портів та форматів команд, а також удосконалення початкових навичок програмування. Реалізація пристрою відображення інформації на семисегментному індикаторі.

Короткі теоретичні відомості

Семисегментний індикатор – це набір світлодіодів, згрупованих в одному корпусі. Світлодіоди утворюють сегменти-палички, шляхом підсвічування яких можна формувати цифри. Індикатор можна безпосередньо підключати до Arduino (рис. 14.1, рис. 14.2), але при цьому буде задіяно 7 виводів контролера на один дисплей, і доведеться замислитись над доданням у програму коду, що реалізує відображення цифр на індикаторі (рис. 14.3). Цей спосіб поганий також і тим, що обмежена кількість підключених індикаторів – більше двох розрядів відобразити вже не вийде, на контролері не вистачить виводів. Для вирішення цієї проблеми пропонується використовувати спеціальну мікросхему, що називається 7-сегментним драйвером (74HC595).

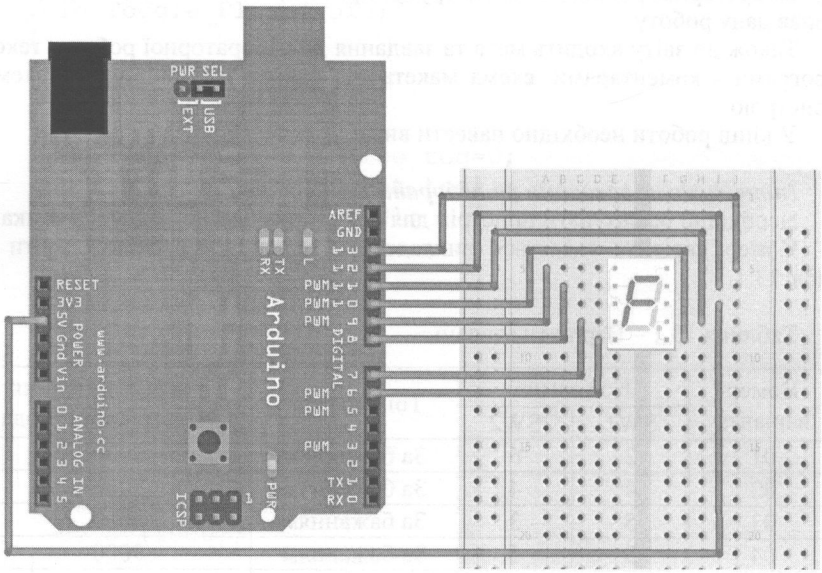


Рисунок 14.1 – Підключення та зовнішній вигляд макета

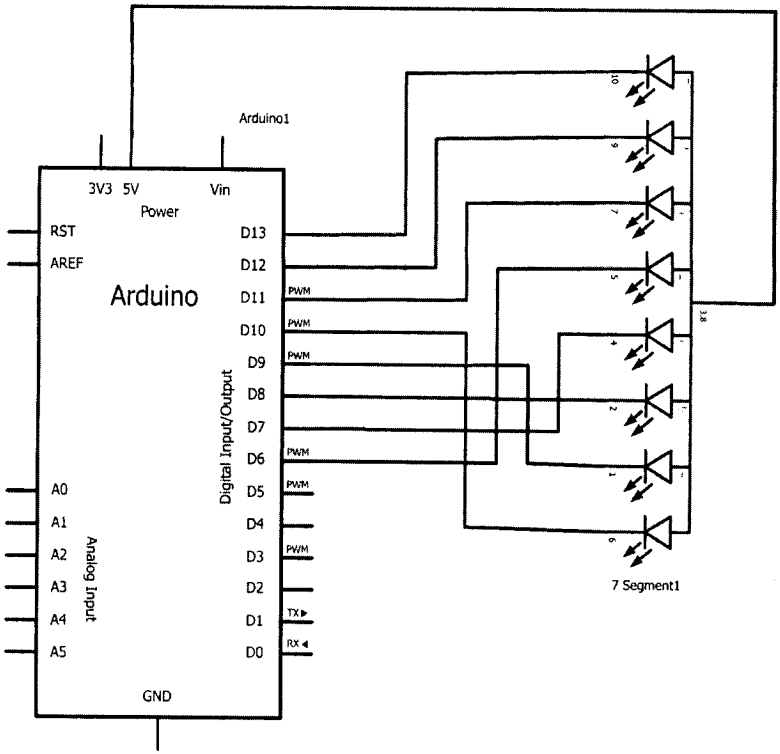


Рисунок 14.2 – Електрична принципова схема макета

7-сегментний дисплей

Зазвичай окремий семисегментний дисплей має 10 контактів, з яких 2 – загальні, а інші 8 – окремі сегменти, включно з десятковою крапкою в правому нижньому кутку. Семисегментні індикатори можуть бути із загальним анодом або із загальним катодом. Необхідно чітко розуміти, якого саме типу семисегментний індикатор знаходиться у вашому користуванні.

На рис. 14.4 com – загальні контакти. Якщо подати до com потенціал +5 В, а до виводу A – 0 В, то загориться, відповідно, сегмент A. Якщо у дисплея загальний катод, то потенціали потрібно підключати навпаки.

Якщо ми подамо цифри на 7-сегментному дисплеї у вигляді байтів, то, подаючи їх по SPI шині в регістр побітно, у момент замикання будемо отримувати потрібну цифру. Таким чином, ми отримуємо можливість керувати семисегментним дисплеєм по чотирьох провідниках, причому кожен наступний семисегментний індикатор не буде потребувати великої кількості додаткових провідників.

```
int a=13;
int b=12;
int c=11;
int d=10;
int e=9;
int f=8;
int g=7;
int p=6;
 void setup ()
 {
   pinMode (a, OUTPUT);
   pinMode (b, OUTPUT);
   pinMode (c, OUTPUT);
   pinMode (d, OUTPUT);
   pinMode (e, OUTPUT);
   pinMode (f, OUTPUT);
   pinMode (g, OUTPUT);
   pinMode (p, OUTPUT);
 }
 void loop ()
 {
   digitalWrite (a, LOW);
   digitalWrite (b, LOW);
   digitalWrite (c, LOW);
   digitalWrite (d, LOW);
   digitalWrite (e, LOW);
   digitalWrite (f, LOW);
   digitalWrite (g, LOW);
   digitalWrite (p, LOW);
 }
```

Рисунок 14.3 – Лістинг програми

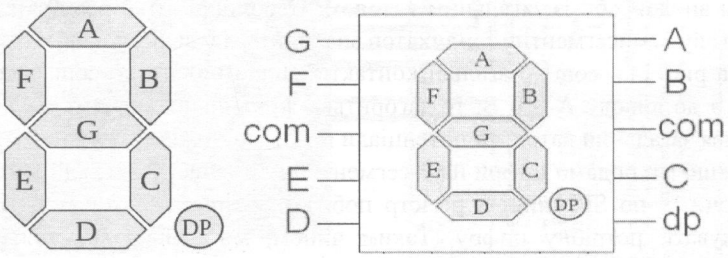


Рисунок 14.4 – Нумерація сегментів та схема розташування виводів семисегментного індикатора

Шина SPI

SPI (Serial Peripheral Interface) – це послідовний синхронний стандарт передачі даних, який призначений для спілкування контролера з периферією. Більшість сучасних мікроконтролерів підтримують SPI, оскільки вона дуже проста і зручна у використанні. У Arduino є бібліотека для роботи з SPI, яка складається всього з шести простих функцій.

Один з пристроїв на шині є ведучим (найчастіше, сам контролер), а інші – веденими. Дані передаються по шині від ведучого до одного обраного веденого або від веденого до ведучого синхронно за тактовим сигналом ведучого.

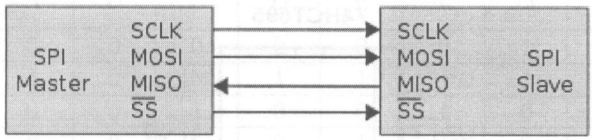


Рисунок 14.5 – Структурна схема шини SPI

Для передачі даних в інтерфейсі SPI використовуються чотири сигнали:

MOSI або SI – вихід ведучого, вхід веденого (англ. Master Out Slave In). Служить для передачі даних від ведучого пристрою до веденого.

MISO або SO – вхід ведучого, вихід веденого (англ. Master In Slave Out). Служить для передачі даних від веденого пристрою до ведучого.

SCLK або SCK – послідовний тактовий сигнал (англ. Serial Clock). Служить для передачі тактового сигналу для ведених пристроїв.

CS або SS – сигнал вибору мікросхеми (англ. Chip Select, Slave Select). Служить для вибору необхідного веденого пристрою.

Передача даних від ведучого проводиться так: ведучий виставляє низький рівень на проводі CS веденого пристрою, з яким він збирається спілкуватися, після чого по MOSI передаються біти. Після закінчення передачі ведучий «відпускає» провід CS – повертає його на високий рівень.

Щоб підключити декілька пристроїв до шини SPI, для кожного пристрою відводиться свій провід CS, і ведучий, почергово «смикаючи» ними, спілкується з тим чи іншим пристроєм. Детальніше підключення декількох пристроїв розглянемо далі.

Регістр зсуву 74HC595

Для реалізації системи формування символів будемо використовувати як ведений пристрій – регістр зсуву 74HC595, до якого підключимо 7-сегментний дисплей. Повна його назва – «8-bit serial-in, serial or parallel-out shift register with output latches» – «восьмибітовий регістр з послідовним введенням і послідовним або паралельним виведенням із вихідними засувами».

Принцип дії такого регістра полягає в тому, що він послідовно отримує біти по лінії DS, а при виставленні низького рівня на виводі STCP «фіксує» отримані біти так, що на його паралельних виводах формується весь отриманий байт. Вихід Q7S «повторює» біти на вході DS, що робить можливим роботу регістра ніби в режимі байпас (*англ.* bypass – обхід, тобто функція, що дозволяє виконати комутацію вхідного сигналу безпосередньо на вихід, минаючи всі функціональні блоки), що можна використовувати у разі послідовного з'єднання декількох регістрів (дана можливість буде використовуватись пізніше).

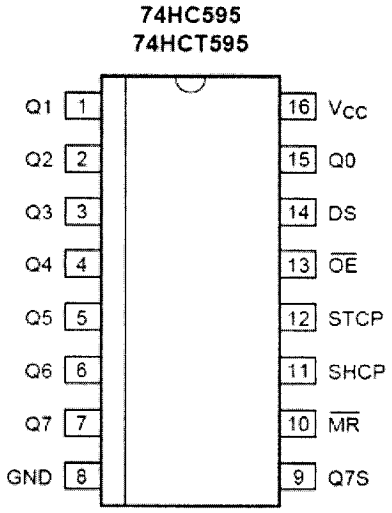


Рисунок 14.6 – Нумерація та позначення виводів регістра зсуву

Де:

– VCC – живлення;

– GND – заземлення;

– DS – послідовний ввід. Повинен підключатися до проводу MOSI шини SPI;

– Q0 : Q7 – паралельні виходи;

– SHCP – тактовий вхід. Повинен підключатися до проводу SCK шини SPI;

– STCP – закривальний вивід. Повинен підключатися до проводу CS шини SPI;

– OE – дозвіл роботи виходів. При низькому рівні виходи працюють;

– Q7S – послідовний вивід;

– MR – загальне скидання. Якщо притягнути його до заземлення – входи обнуляться.

Робота з байтами

У документації до семисегментного індикатора є таблиця, у якій подано який логічний рівень потрібно подавати на той чи інший вхід, щоб загорілася та чи інша цифра. Найімовірніше, ця таблиця виглядає так (замість 0 і 1 можуть бути L і H, відповідно):

Таблиця 14.1 – Подання символів шістнадцяткової системи числення на семисегментному індикаторі

Символ	Сегмент								
	A	B	C	D	E	F	G	dp	hex
0	0	0	0	0	0	0	1	1	C0
1	1	0	0	1	1	1	1	1	F9
2	0	0	1	0	0	1	0	1	A4
3	0	0	0	0	1	1	0	1	B0
4	1	0	0	1	1	0	0	1	99
5	0	1	0	0	1	0	0	1	92
6	0	1	0	0	0	0	0	1	82
7	0	0	0	1	1	1	1	1	F8
8	0	0	0	0	0	0	0	1	80
9	0	0	0	0	1	0	0	1	90
a	0	0	0	1	0	0	0	1	88
b	1	1	0	0	0	0	0	1	83
c	0	1	1	0	0	0	1	1	C6
d	1	0	0	0	0	1	0	1	A1
e	0	1	1	0	0	0	0	1	86
f	0	1	1	1	0	0	0	1	8E

Якщо у дисплея загальний катод, то логічну 1 потрібно замінити на логічний 0, а логічний 0 – на логічну 1. Крайній правий стовпець під назвою hex – це і є подання цифри в байтовому вигляді. Його в документації немає, потрібно розраховувати самостійно. Зробити це дуже просто: пам'ятаючи, що байт передається зі старшого біта (у нашому випадку – крайнього правого), розбити його на два числа по чотири біти і перевести їх у шістнадцяткове подання.

Зауваження: взагалі можна змусити контролер передавати біти, починаючи з молодшого, за допомогою функції SPI.setBitOrder (), проте в цій роботі дана функція не буде використовуватись.

Розберемо цифру 5. У таблиці – це 01001001. Розвертаємо і розбиваємо навпіл, отримуємо 1001 0010, бачимо, що це 9 і 2. Хто підзабув принцип переведення з двійкової системи в шістнадцяткову – дивимося таблицю:

Таблиця 14.2 – Двійковошістнадцяткові числа

Двійкове чотирирозрядне число	Шістнадцяткове число	Двійкове чотирирозрядне число	Шістнадцяткове число
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

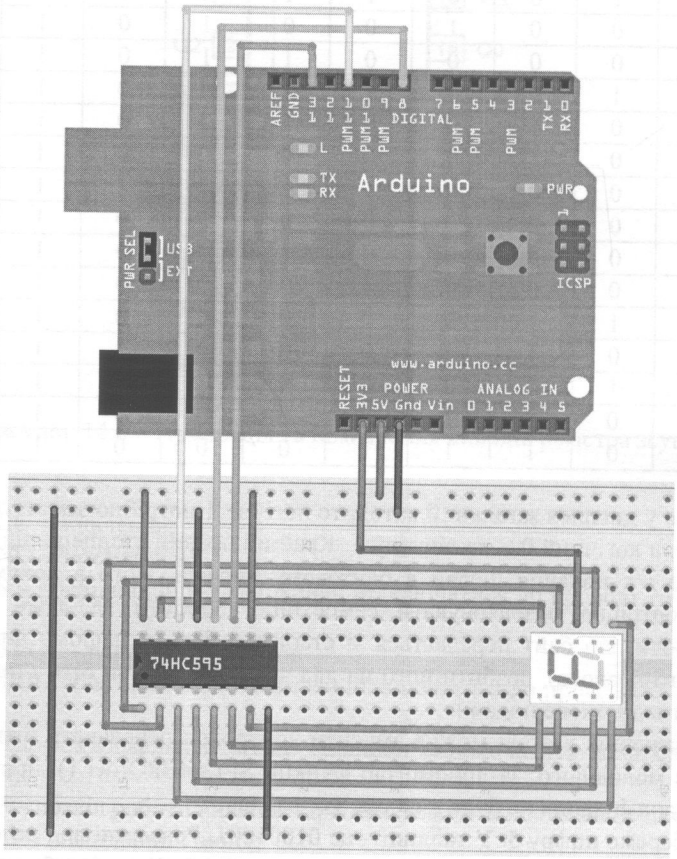


Рисунок 14.7 – Зовнішній вигляд макета

DS лічильника підключаємо до 11 виводу Arduino (жовтий провід); виходи ми завжди дозволяємо, тому підключаємо OE відразу до землі; STCP підключаємо до 8 виводу Arduino (зелений провід); SHCP (тактувальний вхід) підключаємо до 13 виводу Arduino (помаранчевий провід); MR підключаємо до живлення, оскільки не плануємо використовувати скидання; Q7S залишаємо непідключеним, в даний момент він нам не потрібен [3].

```
// Семисегментний індикатор на SPI

#include <SPI.h> // Підключаємо бібліотеку SPI

enum { reg = 8 }; // Провід CS підєднуємо до 8-го
виводу Arduino

void setup()
{
  SPI.begin();  // Ініціалізуємо SPI
  pinMode(reg, OUTPUT); // Налаштовуємо 8-й вивід
як вихід
}

void loop()
{
  // Зберігаємо в масиві всі цифри
  static uint8_t digit[16] = {0xC0,0xF9,0xA4,0xB0,
0x99,0x92,0x82,0xF8,0x80,0x90,0x88,0x83,0xC6,0xA1,
0x86,0x8E};
  // Виводимо цифри по одній
  for (int i=0;i<16;i++){
    digitalWrite(reg, LOW); // Встановлюємо лог. 0
на CS початок передавання
    SPI.transfer(digit[i]); // Передаємо байт
    digitalWrite(reg, HIGH); // Встановлюємо лог. 1
на CS - кінець передавання
    delay(1000); // Очікуємо секунду
  }
  // Очищуємо дисплей на секунду
  digitalWrite(reg, LOW);
  SPI.transfer(0xFF);
  digitalWrite(reg, HIGH);
  delay(1000);
}
```

Рисунок 14.8 – Лістинг програми

Необхідні комплектуючі:

– семисегментний індикатор;
– регістр зсуву 74HC595;
– набір провідників.

Хід лабораторної роботи

1. Безпосереднє підключення макета до семисегментного індикатора:
– скласти макет, зображений на рис. 14.1;
– підключити макет до ПК;
– запустити програму Arduino IDE;
– набрати лістинг програми, поданий на рис. 14.3;
– перевірити формування файлу, який буде записаний на мікроконтролер;
– записати файл на мікроконтролер;
– перевірити функціональність пристрою.
2. Безпосереднє підключення макета до семисегментного індикатора за індивідуальним завданням:
– скласти макет, зображений на рис. 14.1;
– набрати лістинг програми для свого варіанта;
– перевірити функціональність пристрою.
3. Підключення семисегментного індикатора за допомогою регістра зсуву:
– скласти макет, зображений на рис. 14.7;
– підключити макет до ПК;
– запустити програму Arduino IDE;
– набрати лістинг програми, поданий на рис. 14.8;
– перевірити формування файлу, який буде записаний на мікроконтролер;
– записати файл на мікроконтролер;
– перевірити функціональність пристрою.
4. Підключення семисегментного індикатора за допомогою регістра зсуву згідно з індивідуальним завданням:
– скласти макет, зображений на рис. 14.7;
– набрати лістинг програми для свого варіанта;
– перевірити функціональність пристрою.

Зміст звіту

Звіт повинен складатись з титульної сторінки, яка містить номер та тему лабораторної роботи, а також групу, прізвище та ім'я студента, що виконав дану роботу.

Також до звіту входить мета та завдання до лабораторної роботи, текст програми з коментарями, схема макета та електрична принципова схема пристрою.

У кінці роботи необхідно навести висновки.

Індивідуальне завдання до лабораторної роботи

Необхідно реалізувати пристрій для відображення символів на семисегментному індикаторі, як із використанням регістра зсуву, так і без його використання.

Номер варіанта задається викладачем, завдання необхідно взяти з табл. 14.3.

Таблиця 14.3 – Варіанти завдань

Номер варіанта	Вивести число за схемою на рис. 14.1	Вивести послідовно за схемою на рис. 14.5
01	2	1, 2, 3, 4, a, b, c
02	3	5, 6, 7, 8, d, e, f
03	4	9, 0, 2, 5, a, d, c, f
04	5	4, 6, 8, 0, b, d, f, c
05	6	1, 3, 5, 7, 9, e, f, a
06	7	0, 3, 7, 9, a, c, d, f

ГЛОСАРІЙ

Булеві рівняння – Boolean Equation.
Властивості виводів – Pin Properties.
Градація швидкодії – Speed grade.
Заголовок –Title Statement.
Застереження – Warning.
Інструментальні засоби — Primitive Tools.
Категорії – Categories.
Кількість виводів – Pin count.
Коефіцієнт множення (помножити на...) – Multiplied by.
Комбінаційний вузол – Combinatorial Node.
Компіляція – Compilation.
Логічна секція – Logic Section.
Мова програмування високого рівня – Hardware Description Language.
Налаштувати – Customize.
Оголошення вузла – Node Declaration.
Оголошення модуля – Instance Declaration.
Оператор вибору – Case Statement.
Оператор включення – Include Statement.
Оператор прототипу функцій – Function Prototype Statement.
Оператор таблиці відповідності – Truth Table Statement.
Повідомлення – Report.
Повноциклова САПР — Electronic design automation.
Показ типів інструментів – Show tooltips.
Порти – Ports.
Призначення – Assignment.
Причина – Cause.
Процесор повідомлень – Message Processor.
Поточне резюме – Flow Summary.
Рівень – Value.
САПР – Computer aided design.
Секція змінних – Variable Section.
Секція підпроекту – Subdesign Section.
Система числення – Radix.
Корпус – Package.
Умовний оператор – If Then Statement.
Налаштування апаратного засобу – Hardware Setup.
Файл опису схеми програмування – Chain Description File.
Файл часових діаграм – Vector Waveform File.
Часовий аналіз – Timing Analyse.
Шаблон – Template.

ЛІТЕРАТУРА

1. Кофанов В. Л. Проектування цифрових пристроїв на основі САПР Quartus II : практикум / Кофанов В. Л., Осадчук О. В., Гаврілов Д. В. – Вінниця : УНІВЕРСУМ-Вінниця, 2009. – 164 с.

2. Соммер У. Пограммирование микроконтроллерных плат Arduino/ Freeduino. – СПб : БХВ-Петербург, 2012, – 256 с.

3. Кофанов В. Л. Математичні та схемотехнічні основи цифрових пристроїв : навчальний посібник / Кофанов В. Л. – Вінниця : УНІВЕРСУМ-Вінниця, 2005. – 165 с.

4. Кофанов В. Л. Лабораторний практикум з дослідження цифрових пристроїв на основі САПР MAX+PLUS II : навчальний посібник / Кофанов В. Л., Осадчук О. В., Гаврілов Д. В. – Вінниця : УНІВЕРСУМ-Вінниця, 2007. – 196 с.

5. Зубчук В. И. Справочник по цифровой схемотехнике / Зубчук В. И., Сигорский В. П., Шкуро А. Н. – К. : Техніка, 1990. – 448 с.

6. Угрюмов Е. П. Цифровая схемотехніка : учебное пособие / Угрюмов Е. П. – СПб. : БХВ-Петербург, 2002. – 528 с.

7. Кофанов В. Л. Базовые элементы цифровых интегральных микросхем : учебное пособие / Кофанов В. Л. – К. : УМК ВО, 1988. – 116 с.

8. Ерофеев Ю. Н. Импульсные устройства : учебное пособие для вузов по спец. «Радиотехника» / Ерофеев Ю. Н. – М. : Высшая школа, 1989. – 527 с.

9. Калабеков Б. А. Цифровые устройства и микропроцессорные системы : учебник для техникумов связи / Калабеков Б. А. – М. : Горячая линия-Телеком, 2002. – 336 с.

ДОДАТКИ

Додаток А

Середовище розробки Arduino

Після копіювання програми Arduino можна запустити файл Arduino.exe. Перейдіть в каталог Arduino-xxxx. Запустіть програму, двічі клацнувши мишею файл з ім'ям Arduino.exe. Для зручності виклику програми можна створити ярлик на робочому столі.

Після запуску програми протягом кількох секунд з'являється стартове вікно, що показано на рис. А.1.

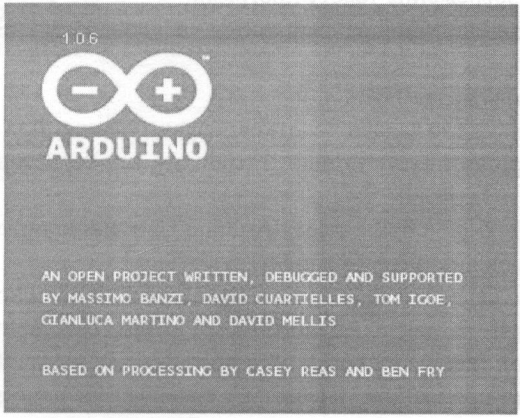


Рисунок А.1 – Вікно програми Arduino
після запуску файлу Arduino.exe

У Arduino-IDE (Integrated Development Environment) – вбудованому середовищі розробки – знаходяться різні інструменти й налаштування, які полегшують спілкування з програмою Arduino (рис. А.2)

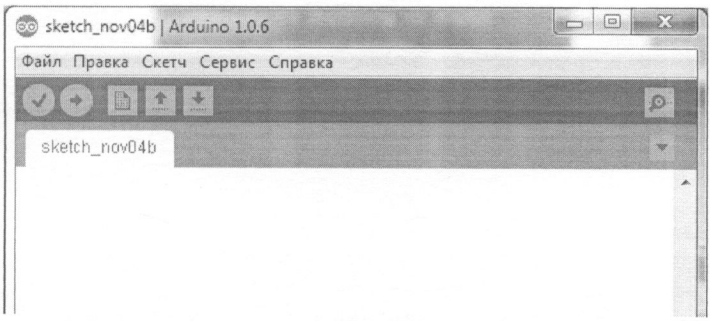


Рисунок А.2 – Середовище розробки Arduino IDE

Розглянемо докладніше середовище розробки Arduino. З меню можна викликати всі функції Arduino. Далі ми ще будемо знайомитися з окремими можливостями. Під головним меню знаходиться панель інструментів, де розташовані такі команди.

– Проверить (Verify) – перевірка та формування файлу, який буде переданий в плату мікроконтролера.

– Загрузить (Upload) – завантаження тексту програми у мікроконтролер.

– Новый (New) – створення нового файлу Arduino.

– Открыть (Open) – відкриття тексту програми.

– Сохранить (Save) – збереження тексту програми.

– Монитор порта (Serial Monitor) – відкриття вбудованого ASCII-термінала.

А.1 Налаштування в Arduino-IDE

До початку роботи необхідно задати початкові налаштування. Для цього потрібно вибрати потрібну плату Arduino (Freeduino) і використовуваний інтерфейс.

У даному випадку необхідно обрати плату Arduino Uno. Якщо потрібна інша, то слід вибрати відповідну плату зі списку.

Перед тим як відкрити середовище розробки (IDE), необхідно приєднати плату мікроконтролера до персонального комп'ютера і встановити драйвери належним чином. Інакше в списку не з'явиться COM-порт. Якщо є декілька альтернативних пропозицій, упевніться в менеджері пристроїв, який COM-порт призначений платі мікроконтролера. У крайньому випадку від'єднайте плату і знову приєднайте її. Потім подивіться, який COM-порт зник з менеджера пристроїв і який був знову зареєстрований після приєднання.

Насамкінець ми встановлюємо ще швидкість у бодах у терміналі. Для цього потрібно натиснути піктограму Монітор порту (Serial Monitor) і задати швидкість 9600 бод. Цю швидкість використовуємо для більшості проектів.

Додаток Б

Індивідуальні завдання

ВАРІАНТ ЗАВДАННЯ 1

Розробити й реалізувати проект на Arduino з використанням фоторезистора.

ВАРІАНТ ЗАВДАННЯ 2

Розробити й реалізувати проект на Arduino для підключення й регулювання швидкості синхронного двигуна.

Додаток В

Опис комплектуючих

Тепер, коли ми маємо початкові відомості про мікроконтролер, платформу Arduino, її програмування та застосування, настав час розпочати експерименти з Arduino. Розглянемо необхідні комплектуючі.

В.1 Список основних комплектуючих

- Плата мікроконтролера Arduino / Freeduino Uno.
- Панель з контактними гніздами Tiny.
- Дві кнопки для друкованих плат RM 2,54.
- Датчик освітленості фоторезистора LDR A9060-13 (R100 = 5 кОм).
- n-p-n-транзистор BC548C.
- Кремнієвий діод 1N4148.
- Акустичний п'єзоперетворювач.
- Світлодіоди червоного кольору.
- Світлодіоди зеленого кольору.
- Світлодіоди жовтого кольору.
- Резистори 1,5 кОм.
- Резистори 4,7 кОм.
- Резистори 47 кОм.
- Резистори 10 кОм.
- Резистори 68 кОм.
- Підлаштовувальний резистора 10 кОм.
- Конденсатор 1 мкФ.
- Гнучкі з'єднувачі.
- Моток монтажного проводу.

В.2 Експериментальна плата Freeduino

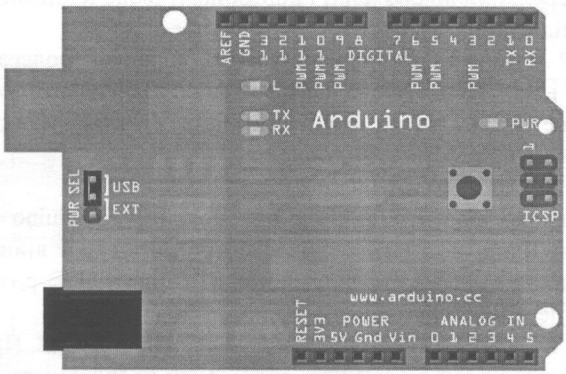
Малогабаритна експериментальна плата Freeduino функціонально відповідає платі Arduino Uno. Платформа Freeduino являє собою версію безкоштовного програмного забезпечення, повністю сумісного з Arduino Uno. Далі ми будемо коротко називати їх «мікроконтролер» і «плата мікроконтролера».

Підключення між персональним комп'ютером і мікроконтролером AVR здійснюється через порт USB. У персональний комп'ютер передаються дані з нашої Arduino-програми. Плата служить як база для експериментів з вже наявним апаратним обладнанням. Ще для експериментів потрібна панель з контактними гніздами (макетна плата), у яку вставляються комплектуючі вироби і за допомогою яких можна виконувати з'єднання провідниками.

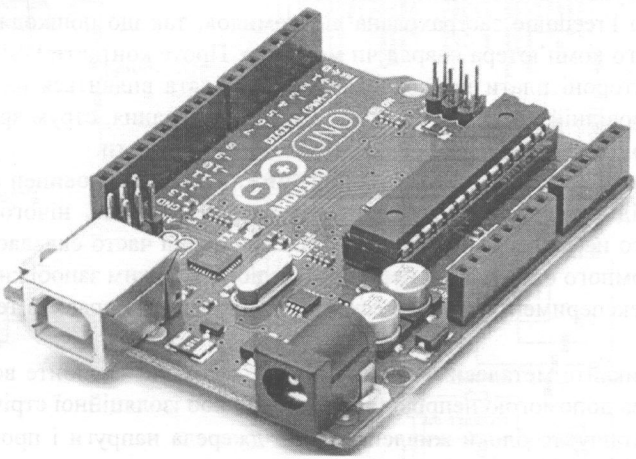
У процесі експериментів з'єднують вибрані комплектуючі і через USB-порт завантажують підготовлені програми в мікроконтролер. Таким чином можна легко освоїти призначення і властивості електронних компонентів та одночасно вивчити основи метрології і техніки управління, а також програмування з Visual Basic Dot.Net і спілкування з платформою Arduino.

В.3 Експериментальна плата мікроконтролера Freeduino

Усі підключення плати Freeduino (рис. В.1) можна здійснити через роз'єми. Світлодіод «PWR» включення живлення сигналізує, що на плату мікроконтролера подається електроживлення. Світлодіод на платі Freeduino, який постійно підключений до цифрового виводу 13 плати Arduino, позначений символом «L». Провідники та комплектуючі вироби можна безпосередньо вставляти в роз'єми. Світлодіоди з позначенням «TX» і «RX» показують активність послідовного інтерфейсу. У процесі обміну даними через UART-інтерфейс світлодіоди ритмічно блимають.



а)



б)

Рисунок В.1 – Експериментальна плата Aduino / Freeduino:
а) – схематичний, б) – зовнішній вигляд

В.4 Живлення макета

Електроживлення може надходити через USB-роз'єм або штекер блока живлення (розрахований на струм приблизно 500 мА). Вибір джерела живлення здійснюється установленням перемички «PWR SEL» у положення «USB» або «EXT». При більших навантаженнях плати мікроконтролера радимо не використовувати живлення від USB, оскільки можливе пошкодження порту USB.

В.5 Кнопка Reset

Кнопка Reset (Скидання) виконує перезавантаження системи. Те ж саме відбувається при виключенні і повторному включенні живлення.

В.6 ISP-підключення

Порт ISP служить для програмування мікроконтролера через ISP-програматор і для початкового завантаження. Для наших експериментів цей порт і не знадобиться. Початковий завантажувач (Bootloader) вже встановлений підприємством-виробником.

Порада:

При живленні плати від USB підключайте Freeduino через USB-розгалужувач. Якщо помилково при експериментуванні виникає коротке замикання, вийде з ладу в більшості випадків тільки USB-розгалужувач, а не порт USB персонального комп'ютера.

Детальна принципова схема плати наведена на рис. В.2. При проведенні експериментів по ній ви зможете перевірити правильність усіх підключень.

В.7 Зауваження з техніки безпеки

Плата Freeduino застрахована від помилок, так що пошкодження персонального комп'ютера навряд чи можливо. Проте контакти USB-гнізда на нижній стороні плати не ізольовані. Якщо плата виявиться на металевій струмопровідній поверхні, то може статися замикання, струм зросте і станеться пошкодження персонального комп'ютера і плати.

Згідно зі специфікацією USB-порт нижнього рівня повинен бути захищений від перевантажень за струмом так, що, власне, нічого особливо страшного не станеться. Зрозуміло, захисна схема часто складається лише з низькоомного опору, який перегорає подібно плавким запобіжникам.

При експериментах, будь ласка, зверніть увагу на правила техніки безпеки.

– Уникайте металевих предметів під платою чи ізолюйте всю нижню сторону за допомогою непровідної пластини або ізоляційної стрічки.

– Розташуйте блоки живлення, інші джерела напруги і проводи з напругою більше 5 В подалі від макетної плати.

– Приєднуйте плату, за змоги, не безпосередньо до персонального комп'ютера, а через розгалужувач, який зазвичай забезпечений додатковою системою захисту.

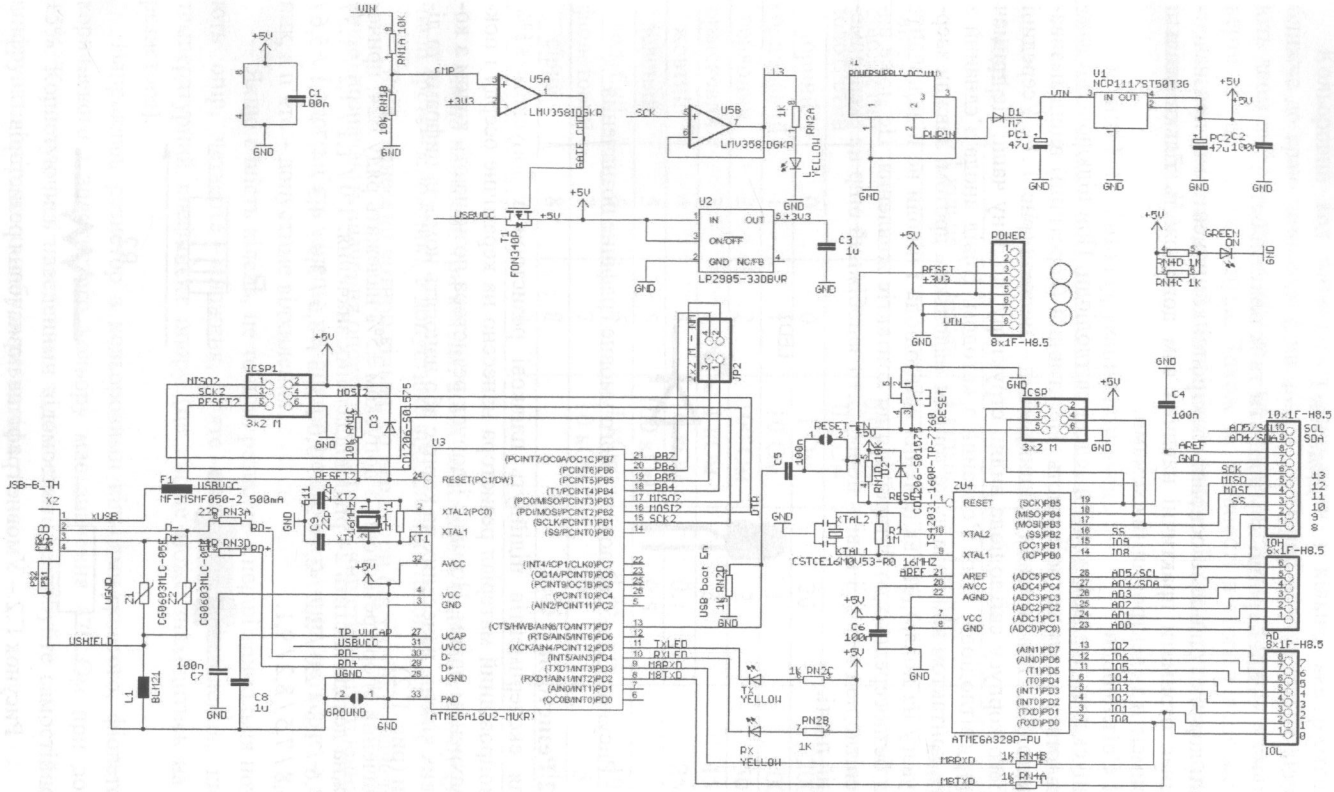


Рисунок В.2 – Принципова схема експериментальної плати Arduino Uno

Додаток Г

Електронні компоненти та їх властивості

Далі коротко опишемо основні електронні компоненти і їх призначення. Однак отримати практичні навички вам допоможуть тільки реальні експерименти з електронними схемами.

Г.1 Світлодіоди

Випускають червоні, жовті й зелені світлодіоди. При підключенні важливо звертати увагу на полярність. Негативний (коротший) вивід називається катодом. Позитивний (довший) вивід – це анод (рис. Г.1). Усередині прозорого корпусу світлодіода видно структуру, подібну чаші. Це тримач кристала світлодіода, який з'єднаний з катодом. Вивід анода з'єднаний з верхнім контактом кристала дуже тонким маленьким дротом. Завжди звертайте увагу на полярність включення світлодіода і ніколи не підключайте прилад безпосередньо до батареї або до контактів живлення. Це може вивести світлодіод з ладу і навіть пошкодити запобіжний опір на експериментальній платі.

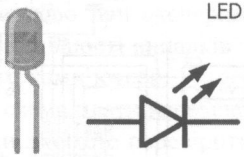


Рисунок Г.1 – Світлодіод і його умовне графічне позначення

Г.2 Резистори

Для експериментів підійдуть плівкові резистори з допуском ±5%. Струмопровідний матеріал резистора нанесено на керамічне осердя і покрито захисним лаком. Номінал і допуск резистора позначають кодом з кольорових кілець, нанесених на корпус або наносять номінал цифрами та літерами (рис. Г.2).

Значення опору резисторів з допуском ±5% належать ряду E24, причому кожна декада містить 24 рівновіддалених значення: 1,0 / 1,1 / 1,2 / 1,3 / 1,5 / 1,6 / 1,8 / 2,0 / 2,2 / 2,4 / 2,7 / 3,0 / 3, 3 / 3,6 / 3,9 / 4,3 / 4,7 / 5,1 / 5,6 / 6,2 / 6,8 / 7,5 / 8,2 / 9,1.

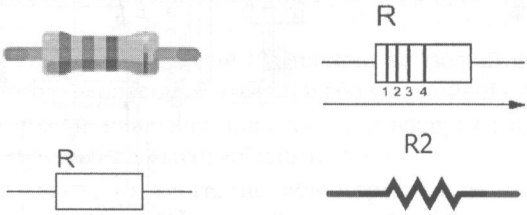


Рисунок Г.2 – Умовне графічне позначення резисторів

Кольоровий код читається, починаючи з кільця, яке розташоване ближче до краю резистора (див. рис. Г.2). Перші обидва кільця відповідають двом цифрам, третє кільце – це множник значення опору в омах. Четверте кільце вказує допуск. Також за наявності п'ятого кільця можливо визначити температурний коефіцієнт опору. Розшифрування позначень наведено в табл. Г.1.

Таблиця Г.1 – Розшифрування позначень

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

– Якщо нанесено три кільця, вони позначають величину опору (у тому числі третє – множник), а допустиме відхилення становить ±20%.

– Якщо нанесено чотири кільця, то перші три (як у пункті, наведеному вище) позначають значення опору, а четверте – допустиме відхилення.

– Якщо є п'ять кілець, перші три позначають опір, четверте – множник, а п'яте – допустиме відхилення.

– Якщо є шість кілець, це точний резистор і перші три кільця позначають опір, четверте – множник, п'яте – допустиме відхилення, шосте – температурний коефіцієнт опору (це кільце може знаходитись на краю резистора).

Наприклад, резистор з кольоровими кільцями жовтого, фіолетового, коричневого і золотистого кольору має значення 470 Ом при допуску ±5%. Користуючись наведеними відомостями, спробуйте самостійно розшифрувати опір якого-небудь резистора.

Г.3 Конденсатори

Конденсатор складається з двох металевих пластин, що знаходяться одна навпроти одної, й ізолювального шару між ними. Якщо подається електрична напруга, то між пластинами конденсатора утворюється електричне силове поле, у якому зберігається енергія. Чим більша площа пластин, тим більше енергії може зберігати конденсатор. Ємність конденсатора вказується і вимірюється у фарадах (Ф). У прикладах, що наведені в цьому посібнику та й взагалі в електроніці, ємність конденсаторів зазвичай знаходиться в діапазоні від 10 нФ (0,00000001 Ф) до 1 000 мкФ (0,001 Ф). Ізоляційний матеріал (діелектрик) збільшує ємність порівняно з повітряною ізоляцією. У керамічних дискових конденсаторах застосовується спеціальний матеріал, який забезпечує велику ємність при маленьких габаритах.

На рис. Г.4 зображений електролітичний конденсатор, який також зустрічається в роботах, що розглядаються в даному посібнику. При підключенні таких конденсаторів потрібно звертати увагу на полярність, оскільки помилка в полярності може призвести до вибуху. На корпусі електролітичного конденсатора поруч з негативним полюсом нанесена біла смужка.

Керамічні конденсатори не мають полярності. На принциповій схемі ці конденсатори зображуються двома паралельними чорними смужками без вказання полярності (рис. Г.5).

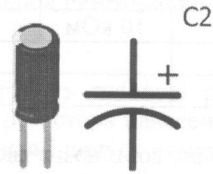


Рисунок Г.4 – Електролітичний конденсатор

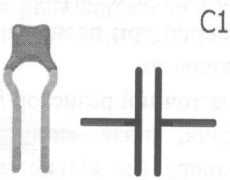


Рисунок Г.5 – Керамічний конденсатор

Примітка. Для наших експериментів керамічні конденсатори не будуть потрібні. Інформація про них наведена тільки для відомості.

Г.4 Транзистори

Транзистори (рис. Г.6) призначені для посилення струмів. Подача невеликого струму в базу призводить до протікання великого струму колектора. Ці прилади мають три виводи: база (Basis, B), колектор (Collector, C) й емітер (Emitter, E).

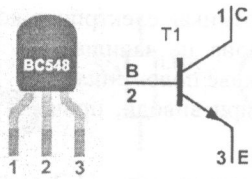


Рисунок Г.6 – Транзистор структури n-p-n і його умовне позначення

У наших експериментах використовується транзистор BC548C. Це універсальний малопотужний транзистор.

Г.5 Діоди

Діод – це електронний вентиль, який пропускає струм тільки в одному напрямку. Діоди виготовляють з германію (Ge) або кремнію (Si). У наших експериментах застосовуються кремнієві діоди типу 1N4148. Це популярні кремнієві діоди, які витримують струм до 100 мА. Включаючи діод, потрібно звертати увагу на полярність. Негативний полюс (катод) позначений маленьким кільцем на краю корпусу (рис. Г.7).

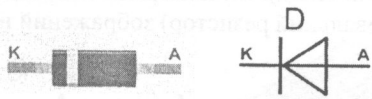


Рисунок Г.7 – Діод і його умовне графічне позначення

Г.6 Акустичний п'єзоперетворювач

Акустичний п'єзоперетворювач (рис. Г.8) може працювати як простий маленький динамік, датчик або мікрофон. Структура п'єзоперетворювача схожа на дисковий керамічний конденсатор, але діелектрик додатково електрично заряджений. Внаслідок цього виникає взаємозв'язок між механічною і електричною напругою. П'єзоелектричний ефект мають кристали кварцу. Ще приклад – електрична запальничка, однак у ній генерується напруга значно вища, ніж в нашому п'єзоакустичному перетворювачі.

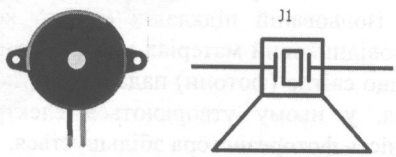


Рисунок Г.8 – П'єзоакустичний перетворювач і його умовне позначення

Г.7 Монтажний провід

За допомогою проводу виконуються всі підключення на монтажній платі. Наріжте маленькими кусачками-бокорізами або ножицями відрізки необхідної довжини і зніміть ізоляцію з обох кінців. Не викидайте використані дроти, вони ще знадобляться.

Г.8 Кнопка

Кнопка замикає або розмикає електричне коло. На відміну від перемикача після відпускання вона не залишається в кінцевому положенні, а повертається в своє початкове положення.

Наша кнопка має чотири виводи, причому два з них завжди з'єднані один з одним (рис. Г.9).

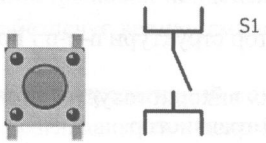


Рисунок Г.9 – Кнопка

Г.9 Потенціометр

Більшість резисторів, що дозволяють регулювати опір за допомогою ручки, називають потенціометрами. Потенціометр має, як правило, три виводи: кінцеві контакти резистивного шару і ковзний контакт повзунка (зазвичай – це середній вивід). Мініатюрний потенціометр з регулюванням «під шліц» (підстроювальний резистор) зображений на рис. Г.10.

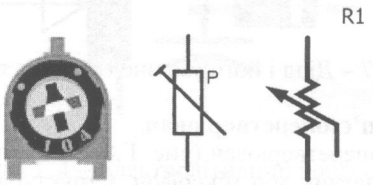


Рисунок Г.10 – Підстроювальний резистор

Г.10 Фоторезистор

Резистор, який реагує на світло, називається фоторезистором (LDR – Light Dependent Resistor). Фоторезистор складається з двох мідних доріжок, нанесених на ізольованій підкладці (білого кольору). Між ними знаходиться напівпровідниковий матеріал у формі звивистої стрічки (червоного кольору). Якщо світло (фотони) падає на світлочутливий напівпровідниковий матеріал, у ньому утворюються електронно-діркові пари. У результаті провідність фоторезистора збільшується, а його опір зменшується. Чим більше світла потрапляє на фоторезистор, тим менший його

опір і тим більший буде електричний струм через нього. Однак, швидкодія фоторезистора невелика, перехідний процес продовжується кілька мілісекунд. На рис. Г.11 наведено зовнішній вигляд і умовне позначення фоторезистора.

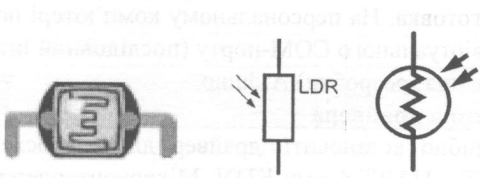


Рисунок Г.11 – Фоторезистор

В.11 Монтажна панель з контактними гніздами

На монтажній панелі з контактними гніздами можна швидко скласти схеми без паяння. Контакти позначені літерами від A до E і від F до J (рис. Г.12). Панель з контактними гніздами складається із 60 стовпців і 10 рядів (від A до J). Корисно трохи обрізати виводи проводів і компонентів (3 мм) навскіс так, щоб отримати щось на зразок клина. Тоді комплектуючі вироби легше вставити в контактні гнізда панелі. Якщо все ж виникають складнощі зі вставлянням виводів, то для установлення компонента найкраще скористатися маленькими плоскогубцями.

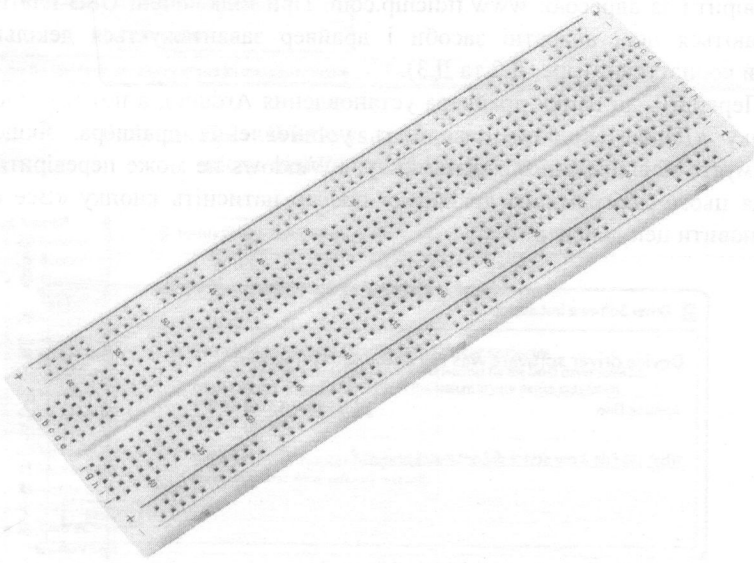


Рисунок Г.12 – Монтажна панель

Додаток Д

Підготовка обладнання перед початком роботи

Перш ніж почати експериментувати і програмувати, буде потрібна деяка попередня підготовка. На персональному комп'ютері потрібно встановити драйвер для віртуального COM-порту (послідовний інтерфейс) і середовище програмування (розробки) Arduino.

Д.1 Установлення драйвера

Спочатку потрібно встановити драйвер для мікросхеми FT232RL – перетворювача USB / UART фірми FTDI. Мікроконтролер на експериментальній платі постачається із заводу з встановленим початковим завантажувачем, необхідне з'єднання між інтерфейсами USB і UART плати Freeduino виконує мікросхема FT232RL. Тому плата Freeduino в менеджері пристроїв позначена як віртуальний COM-порт (віртуальний послідовний інтерфейс).

Можливо, драйвер для FT232RL вже встановлений на вашому персональному комп'ютері, однак може виявитися, що версія драйвера застаріла і не підтримує всі функції мікропроцесора. Необхідна версія драйвера встановлюється автоматично.

Програма видаляє попередній FTDI-драйвер і інсталює новий (рис. Д.1). За необхідності наявність самої останньої версії драйвера можна перевірити за адресою: www.ftdichip.com. При підключені USB-плати визначаються нові апаратні засоби і драйвер завантажується декількома діями користувача (рис. Д.2 та Д.3).

Перейдіть до папки драйвера установлення Arduino, а потім натисніть кнопку «Далі». Windows завершить установлення драйвера. Якщо ви отримуєте повідомлення про помилку: «Windows не може перевірити видавця цього програмного драйвера» просто натисніть кнопку «Все одно встановити цей драйвер».

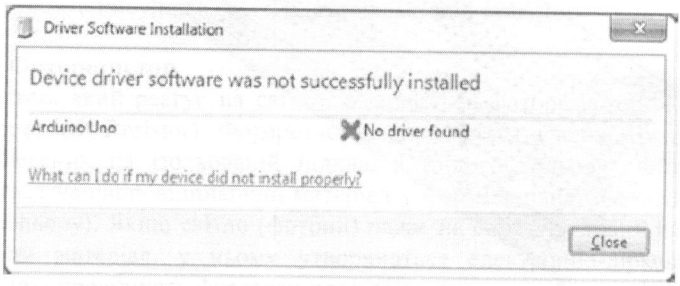


Рисунок Д.1 – Вікно після підключення USB-кабелю до плати

Тепер на персональному комп'ютері існує новий послідовний інтерфейс, наприклад, COM2, COM3 і т. д. Дізнатися номер можна в менеджері апаратних засобів. При повторному установленні адаптера USB ОС Windows надасть максимальний номер COM-порту. У цьому випадку новий пристрій може називатися, наприклад, COM35.

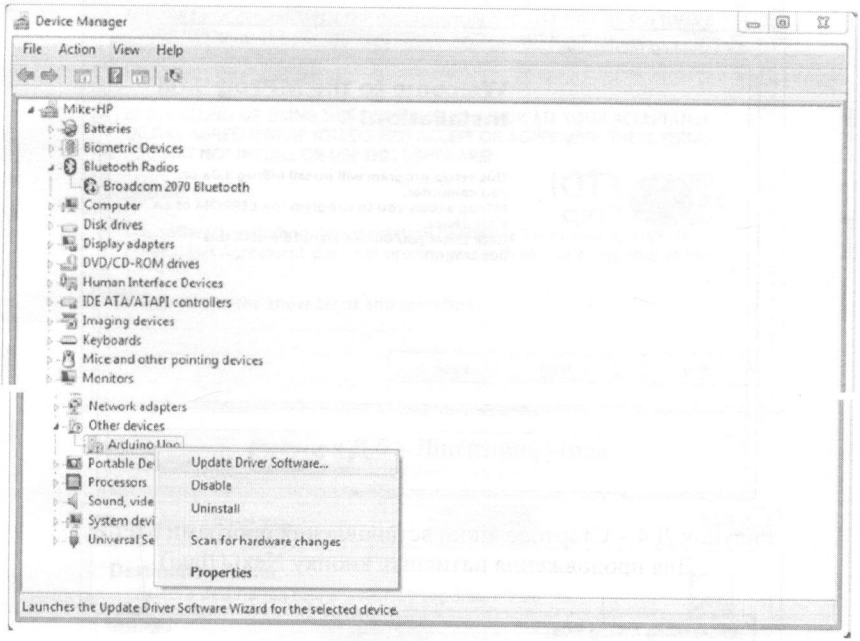


Рисунок Д.2 – Windows знайшов новий пристрій після підключення USB-плати

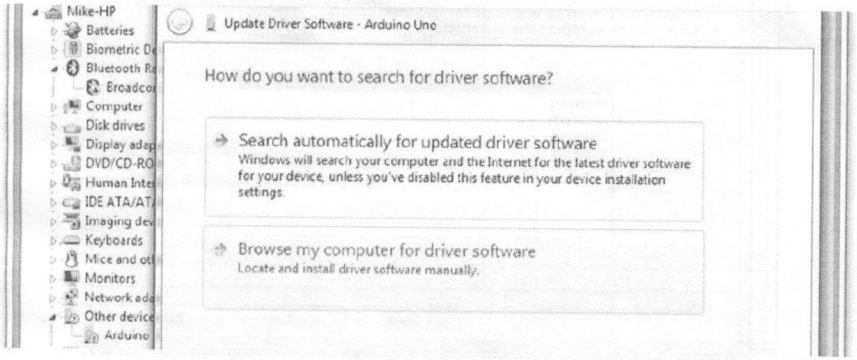


Рисунок Д.3 – Вибір встановлення драйвера в ручному режимі

Д.2 Допоміжна програма MProg для FT232RL

USB-чіп має численні настройки, які користувач може змінити. Для цього призначена програма MProg, яку можна завантажити за адресою http://www.ftdichip.com/Support/Utilities.htm. Установлення програми MProg на комп'ютер буде супроводжуватись рядом діалогових вікон, зображених на рис. Д.4–Д.9.

Рисунок Д.4 – Стартове вікно встановлення програми MProg. Для продовження натисніть кнопку Next (Далі)

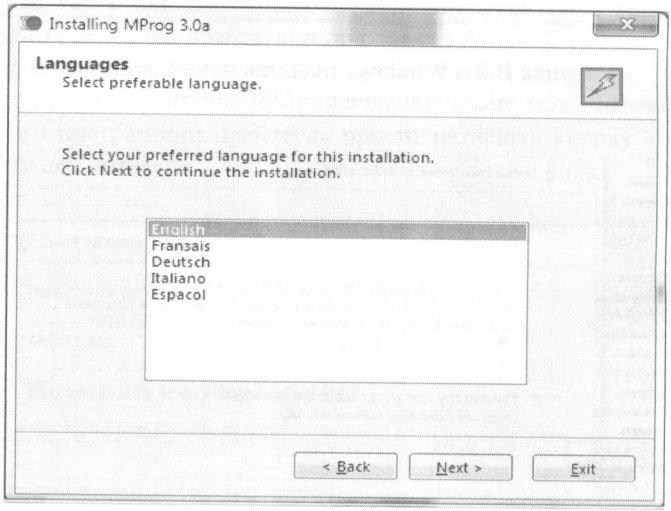


Рисунок Д.5 – Тепер можна вибрати мову інтерфейсу програми

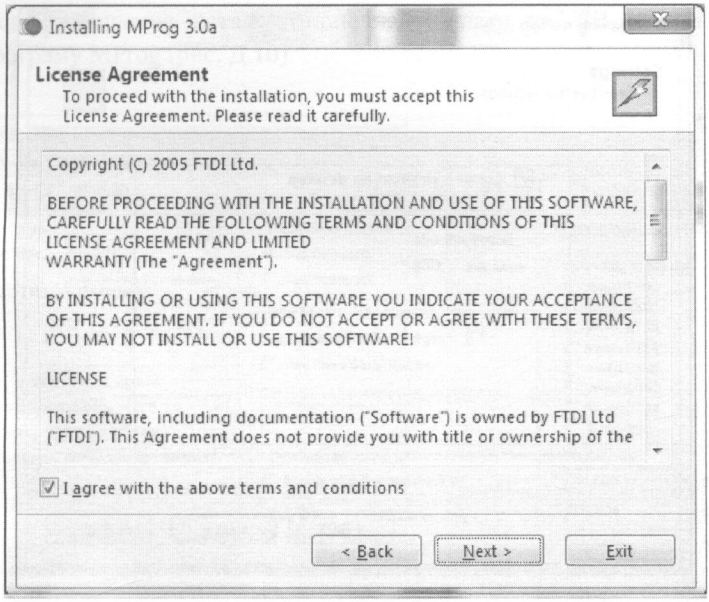


Рисунок Д.6 – Ліцензійна угода

Рисунок Д.7 – У цьому діалоговому вікні можна вибрати цільову папку

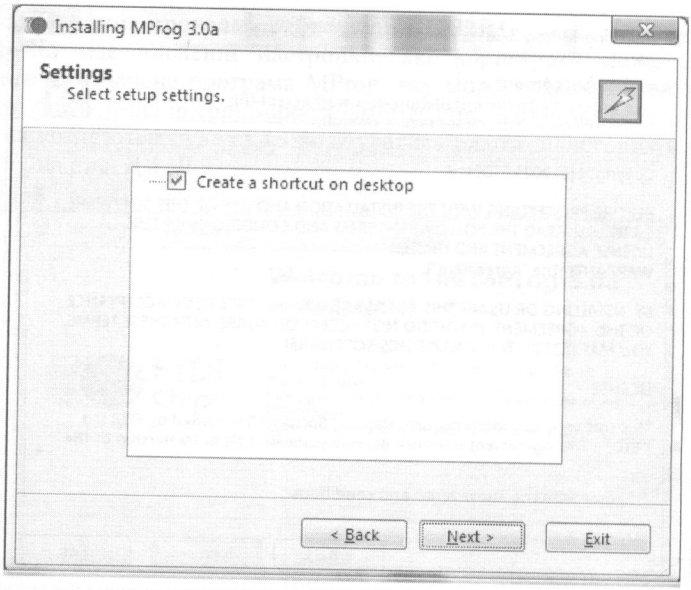


Рисунок Д.8 – Якщо потрібен ярлик на робочому столі, установіть прапорець і натисніть кнопку Next (Далі)

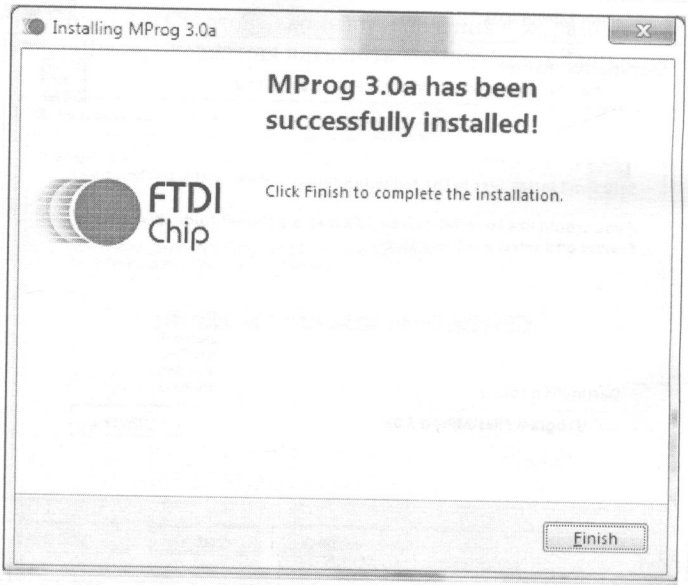


Рисунок Д.9 – Програма успішно встановлена на вашому комп'ютері

Після завершення інсталяції приєднайте плату до USB-порту і запустіть програму MProg (рис. Д.10).

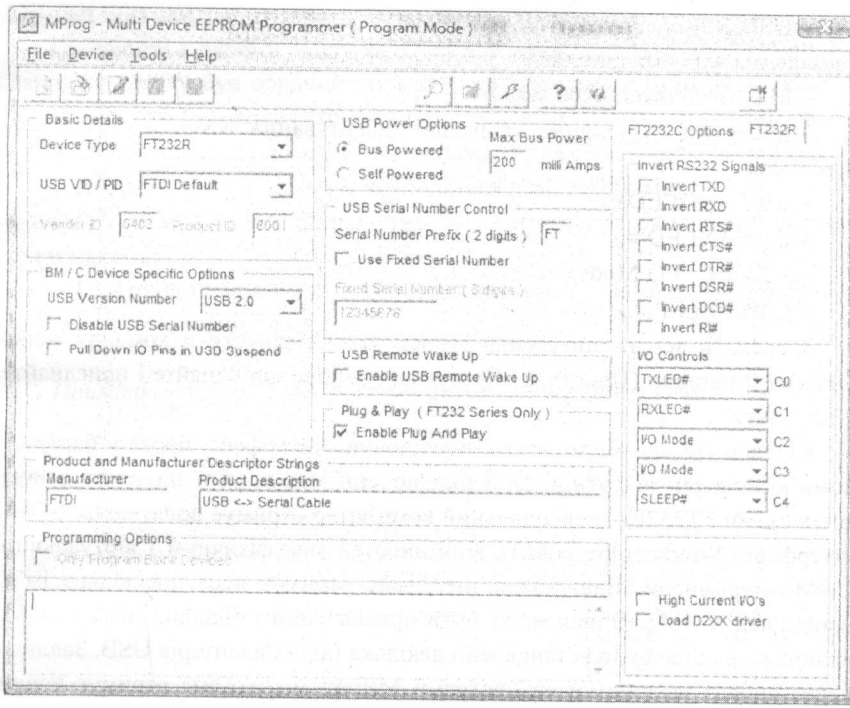


Рисунок Д.10 – Налаштування в програмі MProg

Натисніть кнопку File | New (Файл | Новий) і після цього Tools | Scan (Інструменти | Перегляд). Далі виберіть опцію FT232R в списку Device Type (Тип пристрою). Відкрийте файл Freeduino.ept, який знаходиться в папці MProg Files. Далі виконайте команду меню Device | Programm (Пристрій | Програма).

Переконайтеся в тому, що ніякі сторонні пристрої не підключені до USB-порту персонального комп'ютера.

Д.3 Програмування мікросхеми FT232R за допомогою MProg

Після програмування плата з'являється знову як новий пристрій USB. Вийміть і знову вставте USB-штекер. ОС Windows визначає новий пристрій і автоматично встановлює наявний драйвер. Мікросхема FT232RL отримує в цьому випадку новий номер COM-порту. В інших операційних системах установлення відбувається аналогічно.

Виберіть у меню File (Файл) пункт New (Новий). Ось типові налаштування MProg:

– Device Type (Тип пристрою) – FT232R;

– USB Power Option (Вибір живлення від USB-шини) – Bus Powered (Живлення від USB-шини);

– Max Bus Power – 200 мА.

– Invert RS-232 Signals (Сигнали інвертування RS-232) – прапорець відсутній;

– C0 – TXDLED #;

– C1 – TXLED #;

– C2 і C3 – I/O Mode;

– C4 – SLEEP #.

Збережіть всі налаштування. Тепер можна запустити команду меню Device | Programm (Пристрій | Програма). Заново від'єднайте і приєднайте USB-плату.

Сучасні ПК ще часто мають послідовний інтерфейс, проте у багатьох комп'ютерів він відсутній. При підключенні зовнішньої плати на основі мікросхеми FT232R – персональний комп'ютер отримує додатковий COM-інтерфейс. Windows не робить відмінностей між фізичним і віртуальним COM-інтерфейсом. Ваш новий інтерфейс отримує наступне вільне ім'я, наприклад, COM2. Однак може бути призначений і більший номер COM, якщо вже раніше було встановлено декілька інших адаптерів USB. Завдяки перестановці за допомогою програми MProg, мікросхема отримує новий внутрішній номер пристрою і визнається як новий пристрій з новим умовним номером COM, наприклад, COM3.

Слід зазначити, що не кожне програмне забезпечення може працювати з COM-портами вище COM9. З іншого боку, номери менші COM 10, можливо, раніше були зайняті встановленими пристроями, які в даний час не використовуються. Тому має сенс примусово визначити USB-плату, наприклад, як COM2.

Д.4 Встановлення програмного забезпечення Arduino

Тепер встановимо середовище програмування Arduino, яке базується на програмі Processing і його можна знайти у вільному доступі в мережі. Processing – це проста підмова програмування C. Вона розроблена спеціально для користувачів, які не глибоко володіють програмуванням, але потребують написання власних програм. Додаткову інформацію з цього питання можна знайти за такими посиланнями: http://processing.org/ і www.arduino.cc.

Скопіюйте папку Arduino, яку можна завантажити за адресою http://arduino.cc/en/Main/Software на ваш комп'ютер, наприклад, D:\Arduino. У цій папці можна створювати інші папки для збереження в них пізніше власних програм.

Скопіюйте в окрему папку на жорсткий диск приклади до посібника. У даному випадку був обраний диск D:\, хоча він може бути будь-яким іншим. У нашому випадку готовий каталог буде виглядати приблизно так:

– D:\Arduino\Arduino-z.z.z\ – тут знаходиться середовище розробки;

– D:\Arduino\Projects\ – тут знаходяться ваші програми;

– D:\Arduino\Fritzing-z.z.z\ – тут знаходиться програма для створення схем макетів;

– D:\Arduino\Eagle-z.z.z\ – тут знаходиться програма для створення та перегляду друкованих плат.

Примітка. Через z.z.z позначено актуальні версії програм.

Навчальне видання

Гаврілов Дмитро Володимирович
Осадчук Олександр Володимирович
Звягін Олександр Сергійович

ОСНОВИ КОМП'ЮТЕРНОГО ПРОЕКТУВАННЯ ТА МОДЕЛЮВАННЯ РЕА. ЧАСТИНА 1

Лабораторний практикум

Редактор Т. Старічек
Оригінал-макет підготовлено О. С. Звягіним

Підписано до друку 20.04.2016 р.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 6,5.
Наклад 75 пр. Зам. № 2016-064.

Вінницький національний технічний університет,
навчально-методичний відділ ВНТУ.
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, к. 2201.
Тел. (0432) 59-87-36.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.

Віддруковано у Вінницькому національному технічному університеті
в комп'ютерному інформаційно-видавничому центрі
21021, м. Вінниця, Хмельницьке шосе, 95,
ВНТУ, ГНК, к. 114.
Тел. (0432) 59-87-38.
publish.vntu.edu.ua; email: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.