Р. С. Белзецький, Я. В. Іванчук

КОМП’ЮТЕРНА ІНЖЕНЕРІЯ ТА ОСНОВИ РОБОТОТЕХНІКИ

Міністерство освіти і науки України
Вінницький національний технічний університет

КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ОСНОВИ РОБОТОТЕХНІКИ

Електронний лабораторний практикум
комбінованого (локального та мережного) використання

Вінниця
ВНТУ
2024

УДК 004.415 (076)
Б43

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 12 від 30.05.2024 р.)

Рецензенти:
С. Д. Штовба, доктор технічних наук, професор
А. О. Семенов, доктор технічних наук, професор
Д. Х. Штофель, кандидат технічних наук, доцент

Белзецький, Р. С.

Б43 Комп’ютерна інженерія та основи робототехніки : електронний лабораторний практикум комбінованого (локального та мережного) використання [Електронний ресурс] / Р. С. Белзецький, Я. В. Іванчук. – Вінниця : ВНТУ, 2024. – 78 с.

У виданні стисло викладено основні положення курсу «Комп’ютерна інженерія та основи робототехніки», за якими виконуються лабораторні роботи. Наведено приклади програмного коду та надано алгоритми виконання розрахункової та програмної частин лабораторних робіт. Наведено необхідні довідкові матеріали для виконання лабораторних робіт. Для самоконтролю готовності до лабораторної роботи у практикумі наприкінці кожної теми наведено контрольні запитання. Видання розроблено відповідно до плану кафедри комп’ютерних наук та програми дисципліни «Комп’ютерна інженерія та основи робототехніки» для виконання лабораторних робіт студентами за спеціальністю «Комп’ютерні науки» (освітній ступінь – бакалавр). Лабораторний практикум може бути використаний в процесі підготовки студентів інших спеціальностей.

УДК 004.415 (076)

ЗМІСТ

ВСТУП

Виконання лабораторних робіт з обов'язкової професійної навчальної дисципліни «Комп'ютерна інженерія та основи робототехніки» передбачено навчальним планом освітньої програми «Комп'ютерні науки» за спеціальністю 122 «Комп'ютерні науки».

Дисципліна «Комп'ютерна інженерія та основи робототехніки» ґрунтується на опануванні базових знань з фізичних і логічних принципів побудови електронних схем цифрових елементів і функціональних вузлів комп'ютерної техніки, а також оволодінні основними принципами і методами побудови робототехнічних систем та їх програмування. Ця дисципліна безпосередньо пов'язана і доповнює такі базові дисципліни, як «Вступ до фаху», «Фізика», «Алгоритмізація та програмування».

Лабораторний практикум з дисципліни «Комп'ютерна інженерія та основи робототехніки» призначений для студентів ІІ курсу бакалаврату спеціальності 122 «Комп'ютерні науки», але може стати у пригоді студентами інших спеціальностей.

Процес виконання кожної лабораторної роботи складається з розробки алгоритму розв'язання поставленої задачі; складання структурних блок-схем алгоритмів; їх програмної реалізації; тестування та налагодження розробленої програми; аналізу результатів; підготовки та оформлення звіту.

Звіт виконується на папері формату А4 і має мати наскрізну нумерацію (титульний лист входить до наскрізної нумерації, але не нумерується). Структура звіту:

1) титульний аркуш (обов'язково містить номер лабораторної роботи, тему, прізвище та ініціали, а також, групу виконавця, прізвище ініціали та посаду викладача);
2) мета роботи;
3) індивідуальне завдання;
4) схему електричну-комутаційну;
5) алгоритм роботи програми;
6) текст (лістинг) програми;
7) результати виконання поставленого завдання;
8) висновки за підсумками виконання лабораторної роботи.

Базове програмне забезпечення – MS Office (Word, Excel), Arduino IDE, Мова програмування – C/C++.

Лабораторна робота № 1

ЗНАЙОМСТВО ТА ПОЧАТОК РОБОТИ З ВІДКРИТОЮ ПЛАТФОРМОЮ ARDUINO. ДОСЛІДЖЕННЯ ПРИНЦИПУ РОБОТИ ПРОСТИХ ЛОГІЧНИХ ЕЛЕМЕНТІВ

Мета роботи: Здійснити апаратну та програмну реалізацію логічних елементів за допомогою апаратно-програмної платформи Arduino.

1 Основні теоретичні відомості

Arduino – вільна та відкрита апаратно-програмна платформа для створення електронних прототипів. Ця платформа основана на гнучкому, готовому для використання апаратному та програмному забезпеченні. Arduino, завдяки давачам, здатна сприймати сигнали з оточення та впливати на таке оточення, як: контроль освітлення, двигуни тощо. Мікроконтролер на платі програмується мовою програмування Arduino, яка базується на мові Wiring [1], та середовище розробки Arduino, основане на мові Processing. Проєкти Arduino можуть бути як у вигляді окремих одиниць, так і пов’язаних із ПЗ, які виконуються на ПК.

Плата може бути вільно зроблена користувачем, або замовлена у готовому для використання вигляді, програмне забезпечення може бути вільно завантажене з мережі Інтернет. Опис апаратної частини (для CAD) розповсюджується з вільною ліцензією і може без обмежень модифіковуватись для власних потреб.

1.1 Інтегроване середовище розробки – Arduino IDE

IDE (від англ. Integrated Development Environment – інтегроване середовище розробки) – це додаток або група додатків (середовище), призначене для створення, налаштування, тестування та обслуговування програмного забезпечення.

Інтегроване середовище розробки характеризується наявністю складної функціональності, охоплюючи зокрема редагування і компіляцію вихідного коду, створення програмних ресурсів, створення баз даних і т. д.

В рамках проєкту Arduino було створено програмне забезпечення, що відповідає основним вимогам типового середовища IDE [2]. Це не потужне програмне забезпечення, як, наприклад, Eclipse або NetBeans, а проста, функціональна програма, яка дозволяє нам писати, компілювати і завантажувати програму в мікроконтролер.

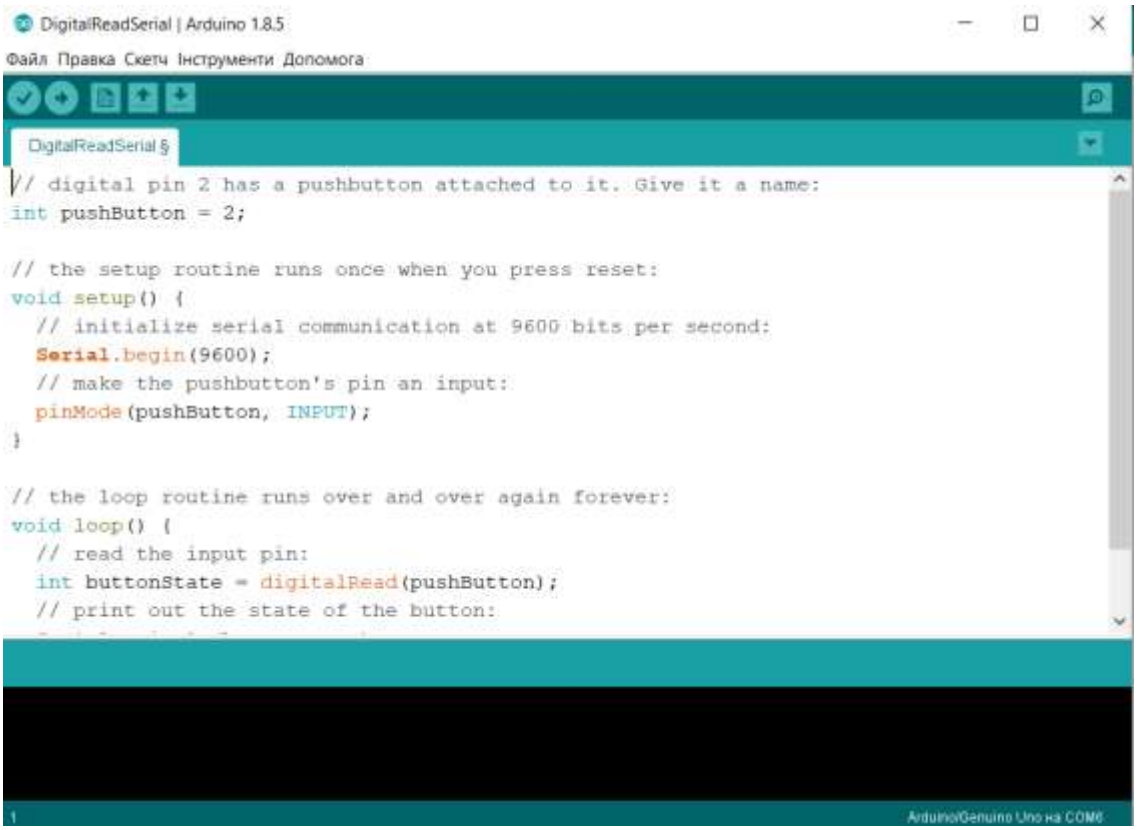


Рисунок 1.1 – Середовище розробки Arduino IDE

Проста структура Arduino IDE є перевагою, бо забезпечує швидке освоєння програми і перехід до розробки додатків для Arduino. Незважаючи на свою простоту і інтуїтивно зрозуміле управління, потрібно звернути увагу на найбільш важливі елементи програми [3].

Після запуску програми ви можете знайти чотири головних функціональних елементи (рис. 1.1):

меню програми;

панель швидкого доступу до найбільш важливих функцій;

редактор (для розміщення коду програми);

панель повідомлень і статусу програми.

Меню програми дозволяє здійснювати управління проєктом, наприклад, створити новий проєкт, зберегти поточний, роздрукувати на принтері вихідний код.

Цікавою особливістю програми є вбудований набір прикладів програм. Це дуже зручно, оскільки приклади програм можна відразу перевірити, завантаживши їх в мікроконтролер. За потреби ви можете зберегти приклад і змінити його відповідно до ваших потреб.

Меню «Файл» і «Правка» містять стандартні параметри (рис. 1.2).

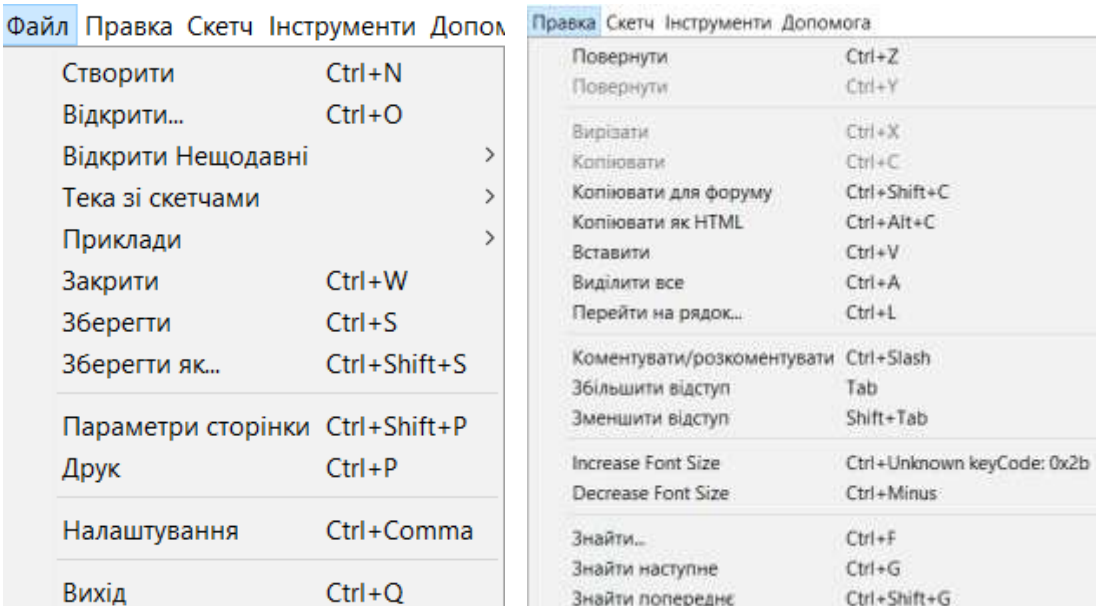


Рисунок. 1.2 – Меню «Файл» і «Правка» Arduino IDE

Меню «Скетч» (рис. 1.3) містить параметри для компіляції проєкту та імпорту необхідних бібліотек.

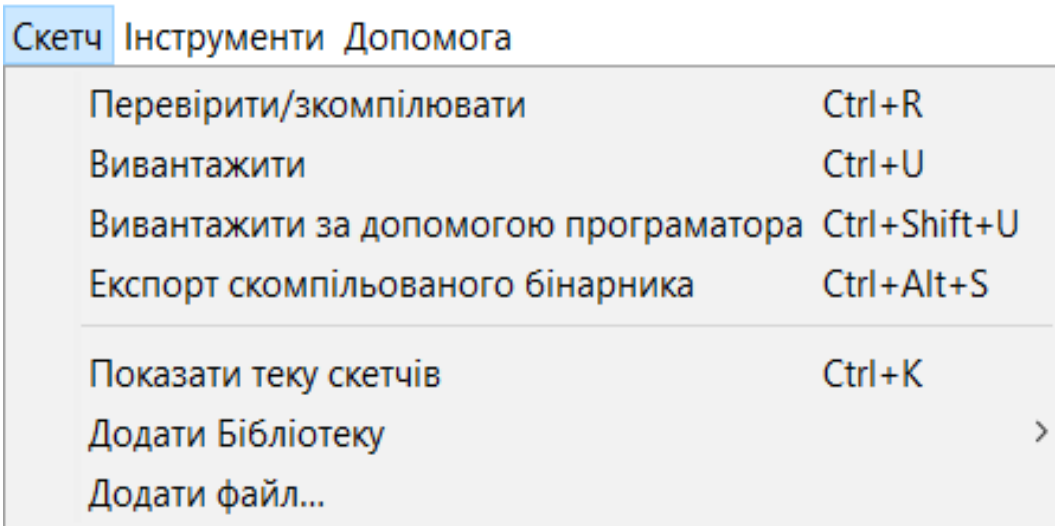


Рисунок 1.3 – Меню «Скетч» Arduino IDE

Цікавим і корисним елементом IDE є меню «Інструменти що містить функції автоматичного форматування коду, архівування проєкту, монітор послідовного порту (USB в Arduino розглядається як звичайний послідовний порт) [4].

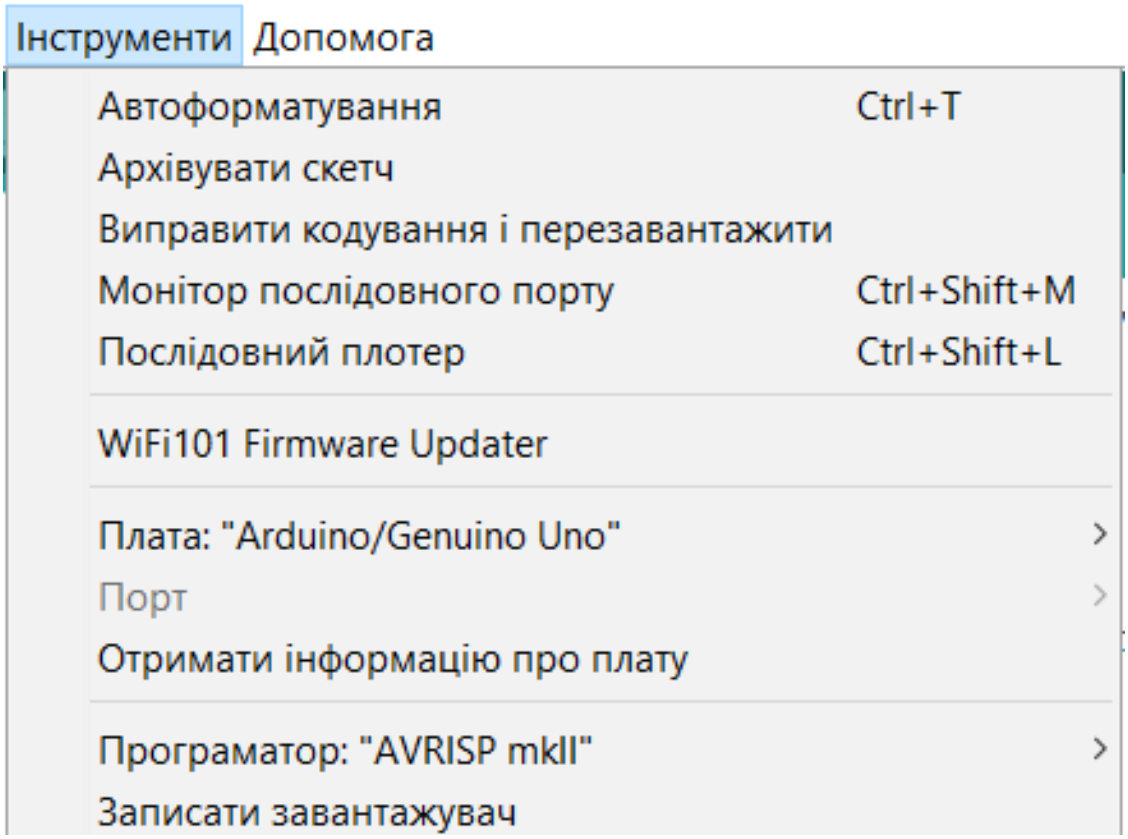


Рисунок 1.4 – Меню «Інструменти» Arduino IDE

Найбільш важливим елементом меню «Інструменти» (рис. 1.4) є можливість вибору відповідної плати, тобто вашої системи Arduino, під'єднаної до комп'ютера. У списку знаходяться всі офіційні версії Arduino. Якщо ваш тип плати відсутній в списку, то ви можемо додати його, змінивши один з файлів програми.

У меню «Інструменти» ви також можете встановити порт, до якого під'єднана плата Arduino. Пакет Arduino IDE сам визначає порт, але іноді потрібно вручну встановити номер порту в налаштуваннях.

За допомогою Arduino IDE можна також завантажити, тобто запрограмувати Bootloader (завантажувач) для нового, чистого мікроконтролера Atmega, що дозволяє клонувати чіпи або просто замінити несправний мікроконтролер в Arduino.

Для зручної роботи з Arduino IDE використовується панель швидкого доступу (рис. 1.5), яка оснащена найбільш важливими кнопками. Це рішення, що полегшує роботу з пакетом IDE, дає нам прямий доступ до практично всіх потрібних параметрів при написанні і тестуванні програми.

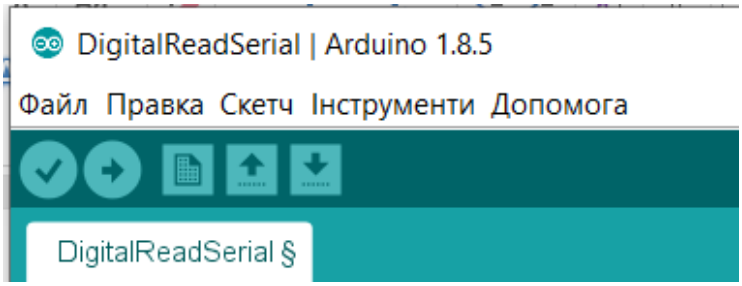


Рисунок 1.5 – Панель швидкого доступу Arduino IDE

Вони дають змогу:

– скомпілювати програму;

– завантажити програму в мікроконтролер (перед прошивкою код програми компілюється);

– почати роботу над новим проєктом;

– відкрити існуючий проєкт;

– зберегти проєкт на диск;

–ввімкнути монітор послідовного порту.

Всі опції, розташовані на панелі швидкого доступу, продубльовані в меню програми.

Додатковим корисним елементом, що знаходиться під кнопкою вмикання монітора послідовного порту, є меню для управління вкладками.

Найбільша частина вікна програми призначена для написання безпосередньо самого коду програми. Редактор в Arduino IDE не дуже просунутий, але має найважливіші елементи, що дозволяють полегшити написання простих програм. До таких елементів можна віднести підсвічування синтаксису і блоків (дужки). Це небагато, але достатньо для простих проєктів.

Останнім елементом програми є вікно повідомлень і статусу. Інформація, що там виводиться, дозволяє користувачеві знайти помилки в програмному коді і отримати підтвердження про завершення компіляції і завантаження програми в мікроконтролер.

1.2 ArduinoUnoR3. Загальні відомості [5].

ArduinoUno – це пристрій на основі мікроконтролера ATmega328. До його складу входить все необхідне для зручної роботи з мікроконтролером: 14 цифрових входів / виходів (шість з яких можна використовувати як ШІМ–виходи), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз’єм USB, роз’єм живлення, роз’єм для внутрішньосхемного програмування (ICSP) і кнопка Reset. Для початку роботи з пристроєм достатньо лише подати

живлення від AC/DC–адаптера або елемента живлення, або підключити його до комп'ютера за допомогою USB–кабелю (рис. 1.6).

Рисунок 1.6 – Зовнішній вигляд Arduino Uno R3

На відміну від всіх попередніх плат Ардуіно, Uno як перетворювач інтерфейсів USB–UART використовує мікроконтролер ATmega16U2 (ATmega8U2 до версії R2) замість мікросхеми FTDI.

Таблиця 1.1 – Характеристики ArduinoUnoR3

Мікроконтролер	ATmega328
робоча напруга	5 В
Напруга живлення (рекомендована)	7–12 В
Напруга живлення (гранична)	6–20 В
Цифрові входи / виходи	14 (з них 6 можуть використовуватися як ШІМ виходи
аналогові входи	6
Максимальний струм одного виводу	40 мА
Максимальний вихідний струм виводу 3.3V	50 мА
Flash-пам'ять	32 КБ (ATmega328) з яких 0.5 КБ використовуються завантажувачем
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактова частота	16 МГц

Живлення

ArduinoUno може живиться від USB або від зовнішнього джерела живлення – тип джерела вибирається автоматично.

Як зовнішнє джерело живлення (не USB) може використовуватися мережний AC/DC-адаптер або акумулятор / батарея. Штекер адаптера (діаметр – 2.1 мм, центральний контакт – плюс) потрібно вставити у

відповідний роз'єм живлення на платі. У разі живлення від акумулятора/батареї, її виводи потрібно під'єднати до виводів Gnd і Vin роз'єму POWER.

Напруга зовнішнього джерела живлення може бути в межах від 6 до 20 В. Однак зменшення напруги живлення нижче 7 В призводить до зменшення напруги на виводі 5V, що може стати причиною нестабільної роботи пристрою. Використання напруги більше 12 В може призводити до перегріву стабілізатора напруги і виходу плати з ладу. З огляду на це, рекомендується використовувати джерело живлення з напругою в діапазоні від 7 до 12 В.

Нижче перераховані виводи живлення, розташовані на платі:

VIN. Напруга, що надходить в Arduino безпосередньо від зовнішнього джерела живлення (не пов'язана з 5 В від USB або іншою стабілізованою напругою). Через цей вивід можна як подавати зовнішнє живлення, так і споживати струм, коли пристрій живиться від зовнішнього адаптера.

5V. На вивід надходить напруга 5 В. від стабілізатора напруги на платі, незалежно від того, як живиться пристрій: від адаптера (7–12 В), від USB (5 В) або через вивід VIN (7–12 В). Живити пристрій через вивід 5V або 3V3 не рекомендується, оскільки в цьому випадку не використовується стабілізатор напруги, що може призвести до виходу плати з ладу.

3V3. 3.3 В, що надходять від стабілізатора напруги на платі. Максимальний струм, споживаний від цього виводу, становить 50 мА.

GND. Вивід землі.

IOREF. Цей вивід надає платам розширення інформацію про робочу напругу мікроконтролера Arduino. Залежно від напруги, знятої з виводу IOREF, плата розширення може перемкнутися на відповідне джерело живлення або задіяти перетворювачі рівнів, що дозволить їй працювати як з 5 В, так і з 3,3 В пристроями.

Пам'ять. Обсяг флеш-пам'яті ATmega328 становить 32 КБ (з яких 0,5 КБ використовуються завантажувачем). Мікроконтролер також має 2 КБ пам'яті SRAM і 1 КБ EEPROM (з якої можна зчитувати або записувати інформацію за допомогою бібліотеки EEPROM).

Входи і виходи. З використанням функцій pinMode(), digitalWrite() і digitalRead() кожен з 14 цифрових вводів може працювати як вхід або вихід. Рівень напруги на виводах обмежений 5 В. Максимальний струм, який може віддавати або споживати один вивід, становить 40 мА. Всі виводи, з'єднані з внутрішніми «підтягувальними» резисторами (за замовчуванням від'єднаними) номіналом 20–50 кОм. Крім цього, деякі виводи Arduino можуть виконувати додаткові функції:

Послідовний інтерфейс: вивід 0 (RX) і 1 (TX). Використовуються для отримання (RX) і передачі (TX) даних по послідовному інтерфейсу. Ці виводи

з’єднані з відповідними виводами мікросхеми ATmega8U2, яка виконує роль перетворювача USB-UART.

Зовнішні переривання: виводи 2 і 3. Можуть служити джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих виводах. Для отримання додаткової інформації див. функцію attachInterrupt().

ШІМ: виводи 3, 5, 6, 9, 10 і 11. За допомогою функції analogWrite()можна виводити 8-бітові аналогові значення в вигляді ШІМ-сигналу.

Інтерфейс SPI: виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Із застосуванням бібліотеки SPI дані виводи можуть здійснювати зв’язок по інтерфейсу SPI.

Світлодіод 13: Вбудований світлодіод, приєднаний до піна 13. При відправленні значення HIGH світлодіод вмикається, при відправленні LOW – вимикається.

В ArduinoUno є **6 аналогових портів** (A0–A5), кожен з яких може подавати аналогову напругу у вигляді 10-бітового числа (1024 різних значень). За замовчуванням, вимірювання напруги здійснюється в діапазоні від 0 до 5 В. Однак верхню межу цього діапазону можна змінити, використовуючи вивід AREF і функцію analogReference (). Крім цього, деякі з аналогових входів мають додаткові функції:

TWI: вивід A4 або SDA і вивід A5 або SCL. З використанням бібліотеки Wire дані виводи можуть здійснювати зв’язок по інтерфейсу TWI.

Крім перерахованих на платі існує ще кілька виводів:

AREF. Опорна напруга для аналогових входів. Може бути задіяна функцією analogReference().

Reset. Формування низького рівня (LOW) на цьому виводі призведе до перезавантаження мікроконтролера. Зазвичай цей вивід служить для функціонування кнопки скидання на платах розширення.

Зв’язок. Arduino Uno надає низку можливостей для здійснення зв’язку з комп’ютером, ще одним Arduino або іншими мікроконтролерами. У ATmega 328 є приймач UART, що дозволяє здійснювати послідовний зв’язок за допомогою цифрових виводів 0 (RX) і 1 (TX). Мікроконтролер ATmega16U2 на платі забезпечує зв’язок цього приймача з USB-портом комп’ютера, і при під’єднанні до ПК дозволяє Arduino визначатися як віртуальний COM-порт. Прошивка мікросхеми 16U2 використовує стандартні драйвери USB-COM, тому встановлення зовнішніх драйверів не потрібне. На платформі Windows потрібний тільки відповідний .inf-файл. У пакет програмного забезпечення Arduino входить спеціальна програма, що дозволяє зчитувати і відправляти на Arduino прості текстові дані. При передачі даних через мікросхему-перетворювач USB-UART, під час USB-з’єднання з комп’ютером, на платі будуть блимати світлодіоди RX і TX. (При послідовній передачі даних за

допомогою виводів 0 і 1, без використання USB-перетворювача, ці світлодіоди не задіюються).

Бібліотека SoftwareSerial дозволяє реалізувати послідовний зв'язок на будь-яких цифрових пінах Arduino Uno.

У мікроконтролері ATmega328 також реалізована підтримка послідовних інтерфейсів I2C (TWI) і SPI. У програмне забезпечення Arduino входить бібліотека Wire, що дозволяє спростити роботу з шиною I2C. Для роботи з інтерфейсом SPI використовуйте бібліотеку SPI.

Програмування Arduino Uno

Arduino Uno програмується за допомогою програмного забезпечення Arduino. Для цього з меню Tools > Board потрібно вибрати "Arduino Uno" з мікроконтролером, відповідним вашій платі.

ATmega328 в Arduino Uno випускається з прошитим завантажувачем, що дозволяє завантажувати в мікроконтролер нові програми без необхідності використання зовнішнього програматора. Взаємодія з ним здійснюється за оригінальним протоколом STK500.

Однак мікроконтролер можна прошити і через роз'єм для внутрісхемного програмування ICSP (In-Circuit SerialProgramming), не звертаючи уваги на завантажувач.

Вихідний код прошивки мікроконтролера ATmega16U2 (або 8U2 на платах версії R1 і R2) знаходиться у вільному доступі. Прошивка ATmega16U2/8U2 містить DFU-завантажувач (Device Firmware Update), що дозволяє оновлювати прошивку мікроконтролера. Для активації режиму DFU потрібно: після переходу в DFU-режим для завантаження нової прошивки можна використати програмне забезпечення Atmel's FLIP (для Windows) або DFU programmer (для Mac OS X і Linux). Альтернативний варіант – прошити мікроконтролер через роз'єм для внутрісхемного програмування ISP за допомогою зовнішнього програматора, проте в цьому випадку DFU-завантажувач затреться.

Захист USB від перевантажень. В Arduino Uno є відновлювані запобіжники, що захищають USB-порт комп'ютера від коротких замикань і перевантажень. Незважаючи на те, що більшість комп'ютерів має власний захист, такі запобіжники забезпечують додатковий рівень захисту. Якщо від USB-порту споживається струм більше 500 мА, запобіжник автоматично роз'єднає з'єднання до усунення причин короткого замикання або перевантаження.

1.3 Порядок виконання роботи

1. Ознайомитися з компонуванням плати «Arduino Uno R3».
2. Завантажити інтегроване середовище розробки Arduino – IDE пройшовши за посиланням [2] та встановити на ПК.

4. Під’єднати плату «Arduino Uno R3» до ПК через USB-порт.

5. Запустити середовище Arduino IDE.

6. В середовищі програмування Arduino відкрийте заготовку, а саме: у меню програми Файл>приклади>Basics>Blink. Відкриється заготовка програми, яка змушує світлодіод на цифровому контакті 13 циклічно вмикатися на 1 секунду та вимикатися на 1 секунду.

7. Вибираємо послідовний порт, через який ми будемо працювати з платою «ArduinoUnoR3». Для цього визначаємо USB-порт, на який підключена плата у меню програми Інструмент>порт.

8. Завантажити програму у плату, для цього натиснути кнопку «Upload» у середовищі розробки. Через декілька секунд світлодіоди RX та TX почнуть блимати, вказуючи про обмін інформацією з платою.

Якщо завантаження пройшло успішно, то у рядку стану буде виведено напис «Done uploading». Через декілька секунд почне блимати вбудований у плату світлодіод. Це і є успішне завантаження та виконання програми.

9. Ввести та проаналізувати код, наведений у прикладі 1.1.

10. Відповідно до виданого викладачем завдання до лабораторної роботи здійснити програмне моделювання логічних елементів**.**

1.4 Зміст звіту

1. Звіт з лабораторної роботи потрібно виконувати на аркушах формату А4.

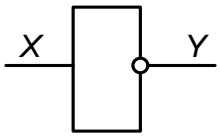
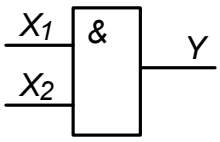
2. Звіт має містити: назву лабораторної роботи, її мету і короткі теоретичні відомості.

3. В розділі «Результати виконання лабораторної роботи» студент має навести схему електричну комутаційну, алгоритм реалізації, а також код програмної реалізації.

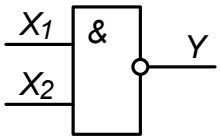
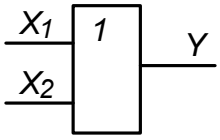
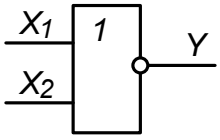
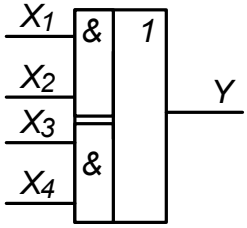
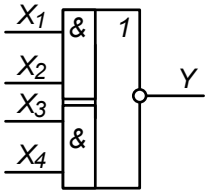
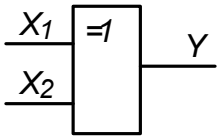
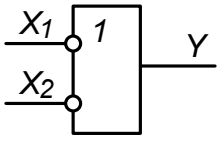
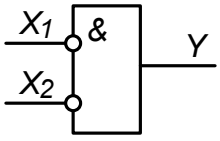
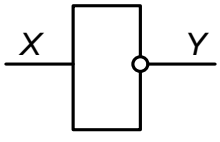
4. Написати висновки до даної лабораторної роботи.

5. Оформити звіт та подати його викладачу на захист.

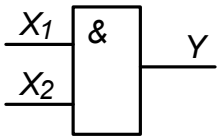
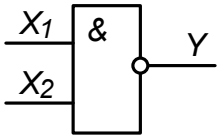
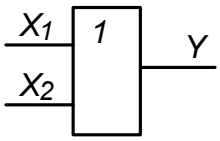
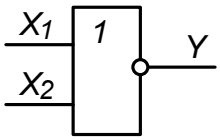
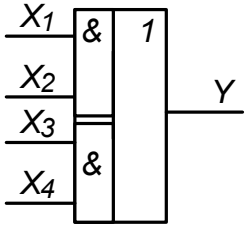
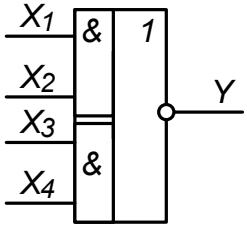
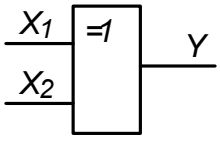
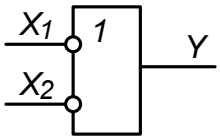
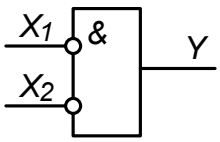
Таблиця 1.2 – Варіанти завдань

Варіант	Елемент	Виконувана функція	Схема	Задіяні порти МК
1	2 НІ	$Y = \overline{X}$		$X_1 – P_2$ $X_2 – P_8$ $У_1 – A_2$ $У_2 – P_3$
2	І	$Y = X_1 \times X_2$		$X_1 – P_{13}$ $X_2 – P_{11}$ $У – P_6$

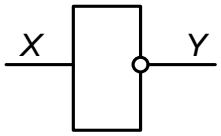
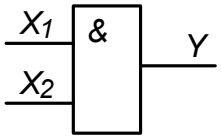
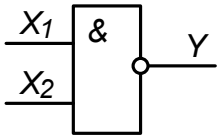
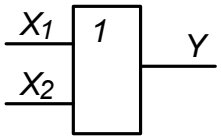
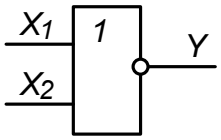
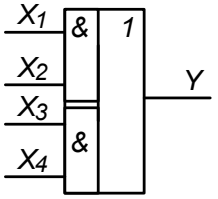
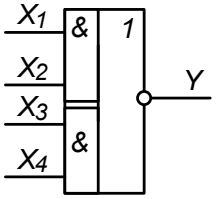
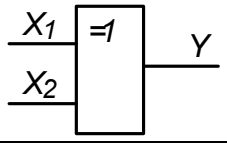
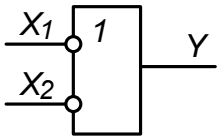
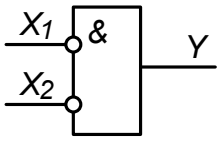
Продовження таблиці 1.2

Варіант	Елемент	Виконувана функція	Схема	Задіяні порти МК
3	І-НІ	$= \overline{X_1 \times X_2}$		X_1 – P_5 X_2 – P_7 У – P_9
4	АБО	$= X_1 + X_2$		X_1 – P_4 X_2 – P_8 У – P_{12}
5	АБО-НІ	$= \overline{X_1 + X_1}$		X_1 – P_3 X_2 – P_4 У – P_5
6	І-АБО	$= X_1 \times X_2 + X_3 \times X_4$		X_1 – P_2 X_2 – P_{12} X_3 – P_3 X_4 – P_4 У – P_5
7	І-АБО-НІ	$= \overline{X_1X_2+X_3X_4}$		X_1 – P_6 X_2 – P_7 X_3 – P_8 X_4 – P_9 У – P_3
8	Виключне АБО (XOR)	$= \overline{X_1} \times X_2 + X_1 \times \overline{X_2}$		X_1 – P_2 X_2 – P_8 У – P_{12}
9	НІ-АБО	$= \overline{X_1} + \overline{X_2}$		X_1 – P_6 X_2 – P_3 У – P_7
10	НІ-І	$= \overline{X_1} \times \overline{X_2}$		X_1 – A_2 X_2 – P_8 У – P_3
11	2 НІ	$= \overline{X}$		X_1 – P_2 X_2 – P_{11} $У_1$ – P_3 $У_2$– P_4

Продовження таблиці 1.2

Варіант	Елемент	Виконувана функція	Схема	Задіяні порти МК
12	І	$Y = X_1 \times X_2$		X_1 – P_{13} X_2 – P_{11} У – P_6
13	І-НІ	$Y = \overline{X_1 \times X_2}$		X_1 – P_5 X_2 – A_2 У – P_9
14	АБО	$Y = X_1 + X_2$		X_1 – P_4 X_2 – P_8 У – P_{12}
15	АБО-НІ	$Y = \overline{X_1 + X_1}$		X_1 – P_3 X_2 – A_3 У – P_5
16	І-АБО	$Y = X_1 \times X_2 + X_3 \times X_4$		X_1 – P_2 X_2 – A_1 X_3 – P_3 X_4 – A_2 У – P_4
17	І-АБО-НІ	$Y = \overline{X_1X_2+X_3X_4}$		X_1 – A_1 X_2 – A_2 X_3 – P_3 X_4 – A_4 У – P_5
18	Виключне АБО (XOR)	$Y = \overline{X_1} \times X_2 + X_1 \times \overline{X_2}$		X_1 – P_1 X_2 – P_8 У – P_{12}
19	НІ-АБО	$Y = \overline{X_1} + \overline{X_2}$		X_1 – P_6 X_2 – P_3 У – P_7
20	НІ-І	$Y = \overline{X_1} \times \overline{X_2}$		X_1 – A_2 X_2 – P_8 У – P_3

Продовження таблиці 1.2

Варіант	Елемент	Виконувана функція	Схема	Задіяні порти МК
21	2 НІ	$Y = \overline{X}$		$X_1 – A_2$ $X_2 – A_4$ $У_1 – P_6$ $У_2 – P_{12}$
22	І	$Y = X_1 \times X_2$		$X_1 – P_{13}$ $X_2 – A_1$ $У_1 – P_6$ $У_2 – P_3$
23	І-НІ	$Y = \overline{X_1 \times X_2}$		$X_1 – A_1$ $X_2 – A_2$ $У – P_9$
24	АБО	$Y = X_1 + X_2$		$X_1 – P_4$ $X_2 – P_3$ $У – P_{12}$
25	АБО-НІ	$Y = \overline{X_1 + X_1}$		$X_1 – P_3$ $X_2 – P_5$ $У – P_5$
26	І-АБО	$Y = X_1 \times X_2 + X_3 \times X_4$		$X_1 – A_2$ $X_2 – P_{12}$ $X_3 – A_3$ $X_4 – P_4$ $У – P_6$
*27	І-АБО-НІ	$Y = \overline{X_1X_2+X_3X_4}$		$X_1 – A_1$ $X_2 – A_2$ $X_3 – A_3$ $X_4 – A_4$ $У – P_3$
28	Виключне АБО (XOR)	$Y = \overline{X_1} \times X_2 + X_1 \times \overline{X_2}$		$X_1 – P_2$ $X_2 – P_8$ $У – P_{12}$
29	НІ-АБО	$Y = \overline{X_1} + \overline{X_2}$		$X_1 – P_6$ $X_2 – P_3$ $У – P_3$
30	НІ-І	$Y = \overline{X_1} \times \overline{X_2}$		$X_1 – A_2$ $X_2 – P_4$ $У – P_7$

Контрольні запитання

1. Для чого встановлюються пристрої «USB SerialConverter» та «USB SerialPort»?
2. Опишіть принцип під'єднання потужних виконавчих пристроїв до порту мікроконтролера.
3. Призначення методу voidsetup().
4. Призначення методу voidloop().
5. Що таке внутрісхемне програмування мікроконтролера?
6. Що таке внутрісхемне підтягування порту до шини живлення «Pull UP»?
7. Наведіть основні характеристики плати Arduino Uno R3.
8. Що таке штрих Шеффера?
9. Що таке стрілка Пірса?
10. Наведіть таблицю істинності для елемента НІ.
11. Наведіть таблицю істинності для елемента АБО.
12. Наведіть контактно-релейну схему заміщення для елемента НІ.
13. Наведіть контактно-релейну схему заміщення для елемента АБО.
14. Наведіть контактно-релейну схему заміщення для елемента І.

Приклад 1.1

```
int X1 = 8;
int X2 = 9;
int Y = 12;
bool a = 0;
bool b = 0;
bool c = 0;

void setup() {
  pinMode(Y, OUTPUT);
  pinMode(X1, INPUT);
  pinMode(X2, INPUT);
}
 loop() {
  a = digitalRead(X1);
  b = digitalRead(X2);
  if ((a == 1) and (b == 1)) {
    c = 1;
  }
  else {
    c = 0;
  }
  digitalWrite(Y, !c);
  delay(200);
}
```

Лабораторна робота № 2

ДОСЛІДЖЕННЯ ПРИНЦИПУ РОБОТИ ПРОСТИХ СХЕМОТЕХНІЧНИХ ВУЗЛІВ ЗА ДОПОМОГОЮ ПЛАТФОРМИ ARDUINO

Мета роботи: Дослідити принцип роботи та методи під'єднання тактової кнопки до цифрового порту. Здійснити керування світлодіодами за допомогою мікроконтролера. Дослідити принцип виведення інформації на одноразрядний семисегментний індикатор.

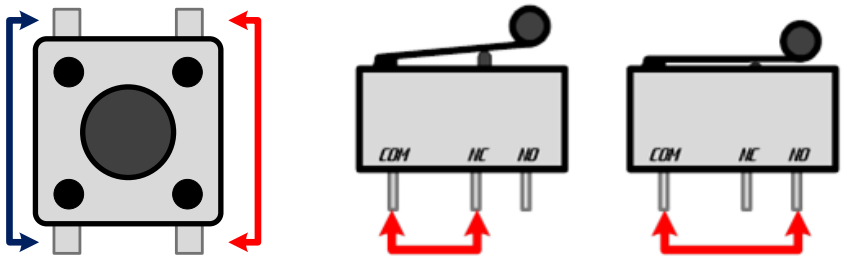
2 Основні теоретичні відомості

2.1 Робота з тактовою кнопкою

Кнопка є найпростішим пристроєм, за допомогою якого можна керувати виконанням алгоритму роботи програми, але фізично виконує дуже просту функцію: замикає і розмикає контакт [6]. Кнопки бувають кількох типів:

- з фіксацією – кнопка залишається натиснутою після відпускання;
- без фіксації – повертається в попереднє положення;
- нормально розімкнена (Normal Open, NO) – при натисканні замикає контакти;
- нормально замкнена (Normal Closed, NC) – при натисканні розмикає контакти.

Тактові кнопки – замикають чи розмикають контакт. У звичайних тактових кнопках виводи з'єднані вздовж через корпус (рис. 2.1, а.) [7]. Перемикачі – зазвичай мають три контакти: загальний COM, нормально розімкнений NO і нормально замкнений NC. При відпущеній кнопці замкнений ланцюг COM-NC, при натиснутій замикається COM-NO (рис. 2.1, б).



а) б)

Рисунок 2.1 – Тактова кнопка. Нормально розімкнена (NO) та нормально замкнена (NC) кнопки

Під’єднання та підтягування тактової кнопки

Мікроконтролер може зчитувати напругу зі своїх пінів. Відповідно, кнопка може подати на пін той рівень, до якого під’єднано її другий вивід. Нікуди не під’єднаний цифровий пін приймає наведення з повітря, і зчитане з нього значення буде практично випадковим. Тобто, під’єднавши до піну 5V (сигнал високого рівня) через кнопку, ми нічого не досягнемо: при натиснутій кнопці на піні зчитуватиметься чіткий сигнал високого рівня, а при відпущеній – випадкове значення. Для вирішення цієї проблеми існує таке поняття, як підтягування (pull) піна. Підтягування виконується до землі (pull down) або живлення (pull up) мікроконтролера за допомогою резистора. Підтягування виконується до протилежного сигналу, тобто, якщо потрібно високий сигнал при натисканні на кнопку, підтягування виконується до землі, якщо потрібно низький сигнал землі – підтягування виконується до живлення. Ось два варіанти під’єднання кнопки, з підтягування до VCC та GND відповідно (рис. 2.2).

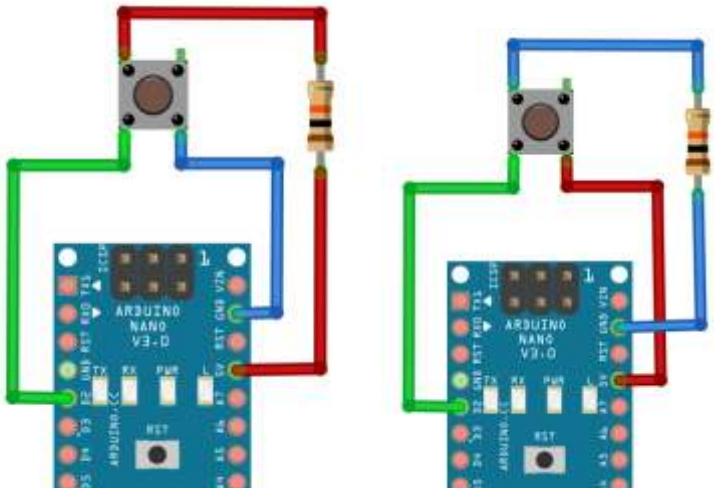


Рисунок 2.2 – Підключення тактової кнопки: а) з підтягуванням до землі; б) з підтягуванням до V_{CC}

При натисканні на кнопку через резистор потече струм, тому що у будь-якому випадку замикається коло живлення-земля. Чим вищий струм, тим більші втрати енергії та нагрівання резистора, тому опір резистора підтягування зазвичай вибирається в діапазоні 5–50 кОм [8]. Якщо ставити опір більше 50 кОм – підтягування може не забезпечити стабільний рівень сигналу на піні, а якщо ставити менше 5 кОм – будуть більші втрати енергії на нагрівання резистора: при опорі в 1 кОм через нього потече струм величиною 5 В/1000 Ом = 5 мА, для порівняння плата Arduino з МК в активному режимі

споживає 20–22 мА. Найчастіше для підтягування використовується резистор на 10 кОм. Також МК має вбудовані резистори для всіх GPIO, ці резистори під'єднані до живлення (V_{CC}), тобто, буквально дублюють першу схему і дозволяють не використовувати зовнішній резистор. У мікроконтролерів іншої архітектури буває підтягування до GND, або може не бути внутрішньої підтягування. Під час використання підтягування до живлення ми отримаємо інвертований сигнал – функція digitalRead () поверне 1 (HIGH) при відпущеній кнопці та 0 (LOW) при натиснутій (при використанні нормально розімкненої кнопки).

На рисунку 2.3 наведено схему під'єднання тактової кнопки до піна D6 плати Arduino з використання внутрісхемного підтягувального резистора.

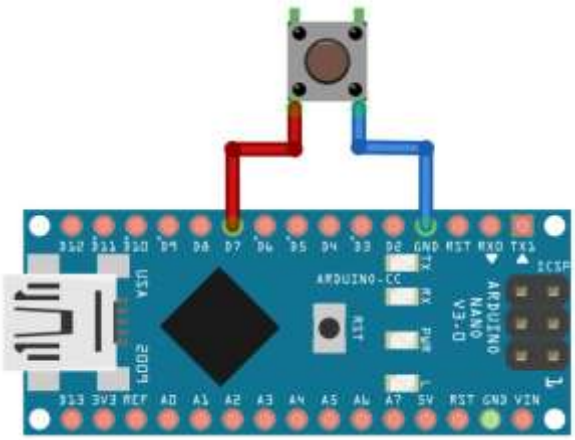


Рисунок 2.3 –Під'єднання тактової кнопки з використанням внутрісхемного вбудованого резистора підтягування

Код реалізації тактової кнопки з використанням внутрісхемного вбудованого резистора підтягування.

```
void setup () {
  Serial. begin (9600) ;
  pinMode (6, INPUT_PULLUP) ;
}
void loop () {
  // виведе 0, якщо кнопка натиснута та 1, якщо ні
  Serial. println ( digitalRead ( 6 )) ;
  delay ( 10 ) ;
}
```

Алгоритми опрацювання натискання тактової кнопки

У більшості реальних проєктів працювати з поточним станом кнопки дуже незручно, наприклад, коли дія має бути виконана один раз за «кліком». Трохи

ускладнимо конструкцію коду, наведеного вище, додавши один прапорець, який запам’ятовуватиме стан кнопки. Така конструкція дозволяє відстежувати натискання та відпускання кнопки та реагувати на них одноразово [9]:

```
   void setup () {
     Serial. begin ( 9600 ) ;
     pinMode (6, INPUT_PULLUP);
   }
   bool flag = false;
   void loop () {
     // Читаємо інвертоване значення для зручності
     bool button_State = ! digitalRead (6);
     if (button_State && !flag ) {              // обробник
натискання
       flag = true;
       Serial. println ( "натиснуто" ) ;
     }
     if ( !btnState && flag ) {                 // обробник
відпускання
       flag = false ;
       Serial.println("відпущено");
     }
   }
```

Дана конструкція поверне «1» (HIGH) при відпущеній кнопці та «0» (LOW) при натиснутій (при використанні кнопки з нормально розімкнутими контактами).

Брязкіт контактів

Брязкіт контактів – явище, що виникає в електромеханічних пристроях. Пов’язане з тим, що контакти при замиканні короткий час багаторазово відскакують один від одного. На рисунку 2.4 наведено графічне подання брязкоту контактів [10].

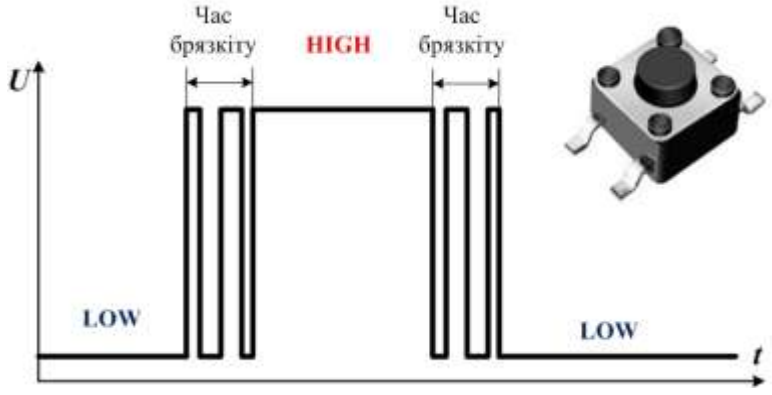


Рисунок 2.4 – Брязкіт контактів

Позбутися брязкоту контактів можна як апаратно, так і програмно: апаратно завдання вирішується за допомогою RC-кола, тобто резистора (~1–10 кОм) і конденсатора (~100 nF). Виглядає це так, як показано на рисунку 2.5.

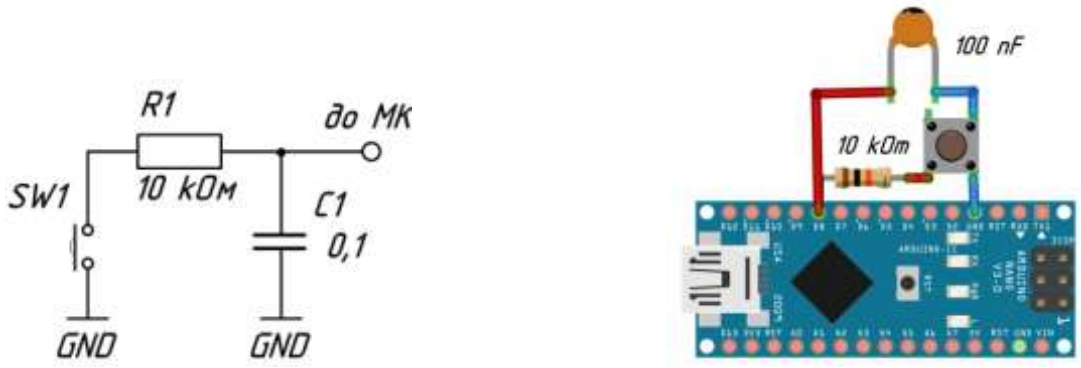


Рисунок 2.5 – Апаратна реалізація RC-кола на тактовій кнопці для антибрязкоту

Програмно можна ввести найпростіший таймер натискання, заснований на millis (), час гасіння брязкоту приймемо 100 мілісекунд. Ось так виглядатиме код:

```
void setup () {
  Serial. begin ( 9600 ) ;
  pinMode ( 3, INPUT_PULLUP ) ;
}
bool flag = false ;
uint32_t btnTimer = 0;
void loop () {
  // Читаємо інвертоване значення для зручності
  bool btnState = ! digitalRead (3) ;
  if ( btnState && !flag && millis () - btnTimer > 100 ) {
    flag = true ;
    btnTimer = millis () ;
    Serial. println ( "натиснуто" ) ;
  }
  if ( !btnState && flag && millis () - btnTimer > 100 ) {
    flag = false;
    btnTimer = millis () ;
    Serial.println("відпущено");
  }
}
```

Зазвичай рекомендується використовувати апаратний спосіб, оскільки він не навантажує ядро зайвими розрахунками. У більшості випадків буде достатньо програмного антибрязкоту.

Утримання кнопки

У пристроях з керуванням кнопкою дуже часто буває потрібна можливість зміни значення як одноразово «кліком» по кнопці, так і автоматично з тим же кроком – при утриманні, наприклад пролистування меню або збільшення числового значення деякого параметра. Такий варіант реалізується дуже просто, додаванням ще однієї умови до нашого попереднього алгоритму, а саме: якщо кнопка була натиснута, але ще не відпущена, і пройшло часу більше, ніж задано, – умова поверне true. У прикладі нижче періодичність «натискань» при утриманні налаштована на 500 мілісекунд (2 рази на секунду).

```
void setup () {
  Serial. begin ( 9600 ) ;
  pinMode ( 3, INPUT_PULLUP ) ;
}
bool flag = false ;
uint32_t btnTimer = 0;
void loop () {
  // Читаємо інвертоване значення для зручності
  bool btnState = ! digitalRead ( 3 ) ;
  if ( btnState && !flag && millis () - btnTimer > 100 ) {
    flag = true ;
    btnTimer = millis () ;
    Serial. println ( "press" ) ;
  }
  if ( btnState && flag && millis () - btnTimer > 500 ) {
    btnTimer = millis () ;
    Serial. println ( "Натиснути і утримувати" ) ;
  }
  if ( !btnState && flag && millis () - btnTimer > 500 ) {
    flag = false ;
    btnTimer = millis () ;
  }
}
```

2.2 Робота зі світлодіодами

Під’єднання світлодіода

Світлодіоди – напівпровідникові прилади з електронно-дірковим переходом, що створює оптичне випромінювання при пропусканні через нього електричного струму в прямому напрямку.

Світлодіод – один із найпоширеніших електронних компонентів, які застосовуються в електроніці. Існує безліч світлодіодів різних варіантів і моделей в різних корпусах і з різними характеристиками [11].

Випромінене світлодіодом світло лежить у вузькому діапазоні спектра. Іншими словами, його кристал випромінює конкретний колір на відміну від лампи, що випромінює ширший спектр, де потрібний колір можна отримати лише застосуванням зовнішнього світлофільтра. Діапазон випромінювання світлодіода великою мірою залежить від хімічного складу використаних напівпровідників[12].

Світлодіод складається з кількох частин (рис. 2.6):

- анод, яким подається позитивна півхвиля на кристал;
- катод, яким подається негативна півхвиля на кристал;
- відбивач;
- кристал напівпровідника;
- розсіювач.

Ці елементи є у будь-якому світлодіоді, незалежно від його моделі.

Світлодіод є низьковольтним приладом. Для індикаторних видів напруга живлення має становити 2–4 В та струм до 50 мА. Діоди для освітлення споживають таку ж напругу, але їх струм вищий і може досягати декількох Ампер.

При під’єднанні світлодіодів важливо знати два основні правила.

У світлодіода є позитивний і негативний контакти, тому важливо дотримуватись полярності при під’єднанні.

Першого правила легко дотримуватись, якщо знати, де у світлодіода мінус, а де плюс. Тут на допомогу приходять правила маркування. Ми дивимося на ніжки та бачимо, що вони різного розміру. Довша ніжка означає плюс. Якщо немає можливості порівняти довжину або хтось вже до вас відрізав частину ніжок, ми обмацуємо корпус (візуально визначити буде складно) – з однієї зі сторін корпус злегка обрізаний (скошений), з цього боку мінус.

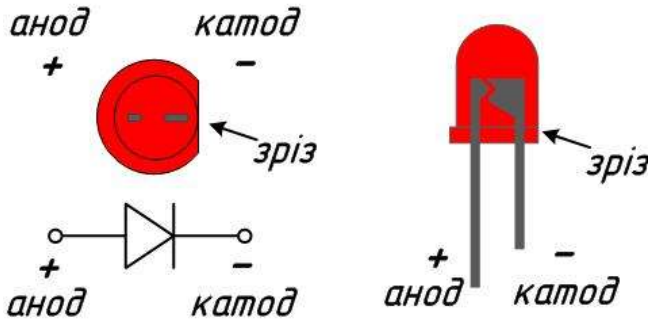


Рисунок 2.6 – Будова світлодіода

У світлодіодів є обмеження щодо струму, що протікає через них, тому потрібно забезпечувати правильний режим ввімкнення світлодіода в електричну схему.

Для того, щоб розрахувати, який опір резистора потрібно під'єднати в електричне коло, скористаємось формулою

$$\frac{U_{\text{ж}} - U_{LED}}{I_{LED}}, \text{Ом}.$$

Потужність резистора

$$P \quad (U_{\text{ж}} - U_{LED}) \cdot I_{LED}, \text{ Вт}.$$

де $U_{\text{ж}}$ – напруга джерела живлення;

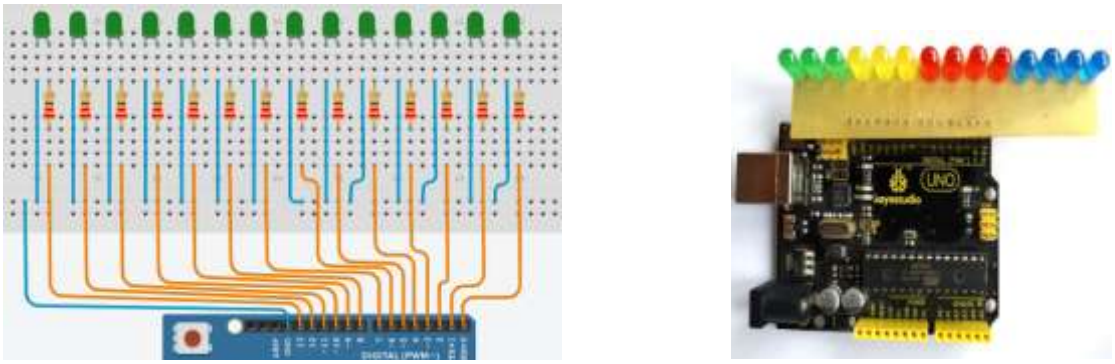
U_{LED} – прямий спад напруги світлодіода;

I_{LED} – робочий струм світлодіода.

Якщо отриманого результату в списку номіналів резисторів немає – використовуйте найближчий до нього в бік збільшення або змінюйте додатковими опорами. Також не забувайте і про потужність резистора. Недостатньо потужний буде «грітися» або перегорить. Потрібно використовувати опір з запасом потужності 20–30%.

Керування світлодіодною шкалою

Для під'єднання світлодіодної шкали до плати Arduino потрібно задіяти 14 цифрових виводів, кожен з анодів світлодіодів під'єднується до виводу Arduino через обмежувальний резистор 220…470 Ом. Усі катоди світлодіодів під'єднані до шини землі на бредборді, або використовується плата розширення, як показано на рисунку 2.7.



а) б)

Рисунок 2.7 – Схема під'єднання світлодіодної шкали до плати Arduino:
а – в середовищі Tinkercad; б – за допомогою плати розширення

Приклад 2.1

```
int a = 0;
int i = 0;
void setup() {
  for (int i = -1; i < 14; i++) {
    pinMode(i, OUTPUT);
  }
}
void loop() {

  a = analogRead(A0);
  a = map(a, 0, 1023, 5, 90);
  for (int i = -3; i < 16; i++) {
    digitalWrite(i, HIGH);
    digitalWrite(i - 3, LOW);
    digitalWrite(i + 3, LOW);
    delay(a);
  }
  for (int i = 16; i > -3; i--) {
    digitalWrite(i, HIGH);
    digitalWrite(i - 3, LOW);
    digitalWrite(i + 3, LOW);
    delay(a);
  }
}
```

Керування яскравістю світлодіода

В більшості мікроконтролерів широтно-імпульсна модуляція (ШІМ) робиться на основі таймера, в якому налаштовують параметри, щоб формувати ШІМ імпульси [13].

ШІМ є цифровим сигналом прямокутної форми, в якому рівень сигналу перемикається між логічним нулем й одиницею. Логічна одиниця відповідає напрузі між живленням плати (наприклад, 5 В на Arduino UNO) та нульовою напругою. ШІМ блок визначає довжину цієї логічної одиниці в часі. Ця ширина називається шириною імпульсу. А тривалість високого і низького рівнів називається періодом (рис. 2.8) [14].

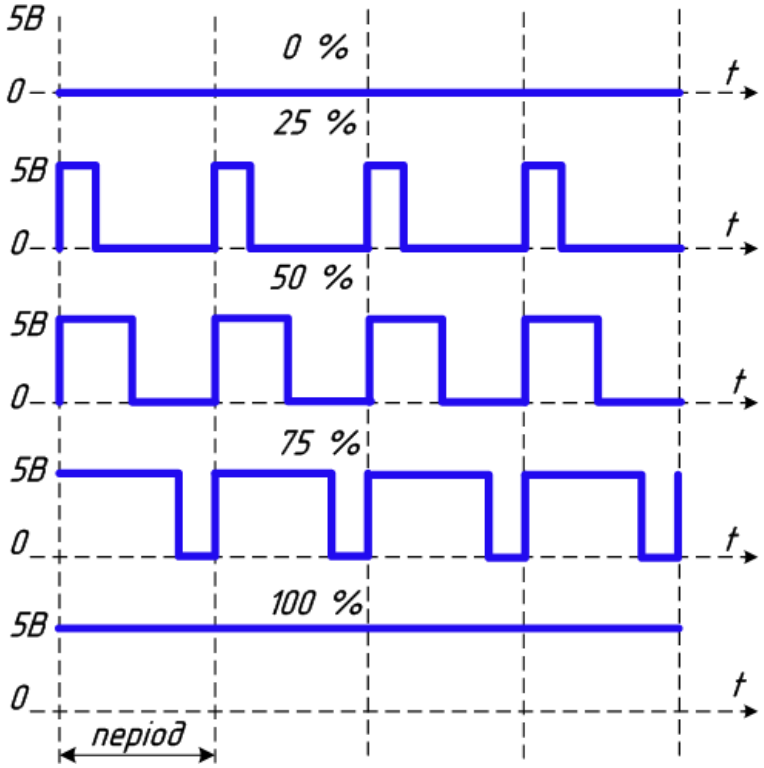


Рисунок 2.8 – Графічне подання сигналу широтно-імпульсної модуляції

Якщо частота імпульсів буде досить високою, то для під'єднаного світлодіода результат буде такий, ніби сигнал є постійною напругою від 0 до Vcc. Таким чином ми можемо керувати його яскравістю.

В Arduino ШІМ керується за допомогою спеціальних регістрів. Мікросхема ATmega168P/328P має три ШІМ-таймери, що контролюють 6 ШІМ-виходів. На платі ці виходи позначені символом тильди (~): ~3, ~5, ~6, ~9, ~10, ~11.

В Arduino IDE є спеціальна функція для керування ШІМ – analogWrite(pin, dutyCycle), де dutyCycle – це значення від 0 до 255, а pin – це один із контактів ШІМ (3, 5, 6, 9, 10 або 11). Наприклад, analogWrite(255, 5) запитує 100% робочий цикл (завжди ввімкнено) на виході 5, а analogWrite(127, 9) – 50% робочий цикл (у половині часу) на виході 9 [15].

Недоліком цієї функції є те, що вона не забезпечує контролю над частотою (періодом) ШІМ сигналу.

Розглянемо приклад, які демонструє роботу ШІМ на платі Arduino UNO. Для демонстрації вам потрібно скласти схему (рис. 2.9) або використати плату розширення, яка зображена на рисунку 2.7, б.

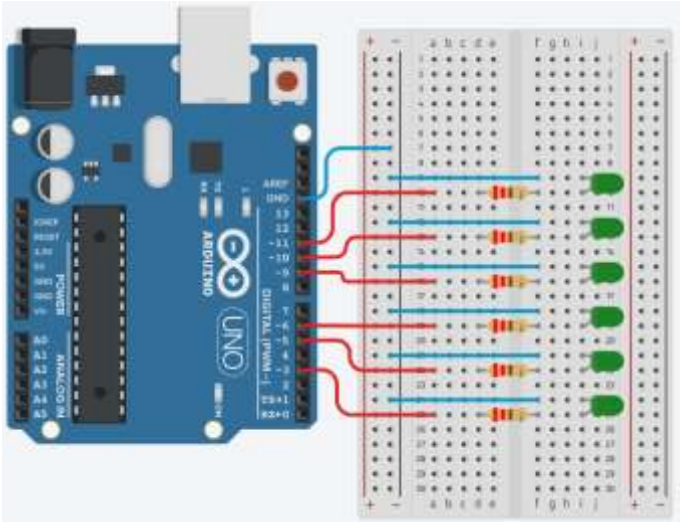


Рисунок 2.9 – Схема під’єднання світлодіодів до ШІМ портів плати Arduino

В Arduino IDE є готовий приклад, який демонструє повільне згасання світлодіода на контакті 9. Цей скетч знаходиться в меню Файл → Приклади → 01.Basics → Fade.

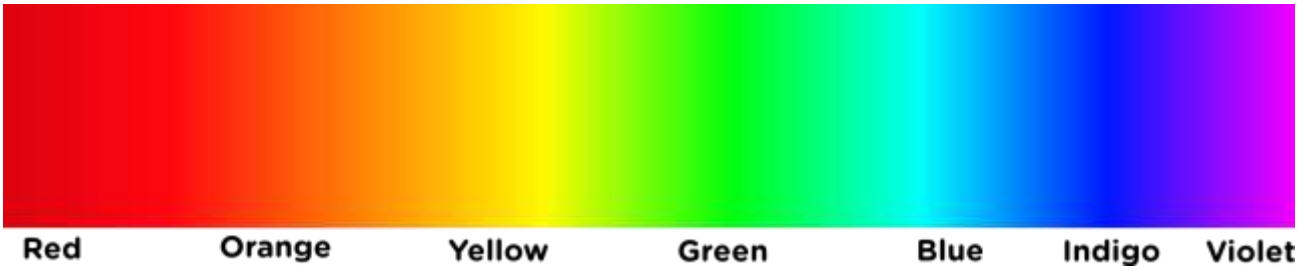
2.3 Керування RGB світлодіодом

Якщо ми подивимося на відтінки, які спостерігає наше око, то первинними кольорами є – червоний, зелений і синій.

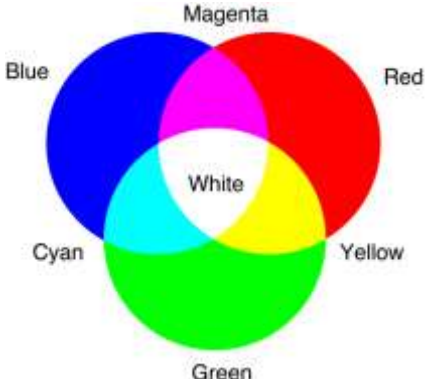
Це пояснюється тим, що людське око має три типи світлочутливих клітин, які сприймають світлові хвилі від первинних кольорів червоного, зеленого і синього, дозволяючи нам бачити весь спектр інших кольорів.

Таким чином, червоний, зелений і синій кольори були визначені як основні кольори, відомі як RGB (за їхніми англійськими назвами read – червоний, qreen – зелений, dark blue – синій) [16].

Світло – це потік фотонів, але водночас воно є електромагнітною хвилею, випромінюванням. Людське око сприймає дуже вузький діапазон цього випромінювання: приблизно від 390 до 790 ТГц (терагерц), так зване видиме випромінювання або видиме світло. «Орієнтуватися» в цьому діапазоні електромагнітного випромінювання прийнято у оберненій величині – довжині хвилі, що вимірюється в даному випадку в нанометрах (нм): людське око бачить випромінювання в діапазоні від ~ 400 нм (фіолетовий) до ~ 800 нм (червоний). Між синім і червоним є ще один важливий колір – зелений:



Червоний (Red, **R**), зелений (Green, **G**) і синій (Blue, **B**) є основними кольорами: змішуючи ці три кольори в різних пропорціях можна отримати, плюс-мінус, всі інші кольори[16].



RGB-світлодіоди

RGB-світлодіод – це, насправді, три світлодіоди в одному корпусі. Щоб не реалізовувати зайві виводи, всі аноди або катоди світлодіодів об'єднуються і виходить 4 контакти: R, G, B та загальний. Загальним може бути як мінус-катод (Common Cathode), так і плюс-анод (Common Anode).

На рисунку 2.10 зображена «розпіновка» типового RGB світлодіода: найдовша ніжка – загальний вивід, ліворуч – вивід червоного світлодіода, праворуч – зелений, а далі – синій.

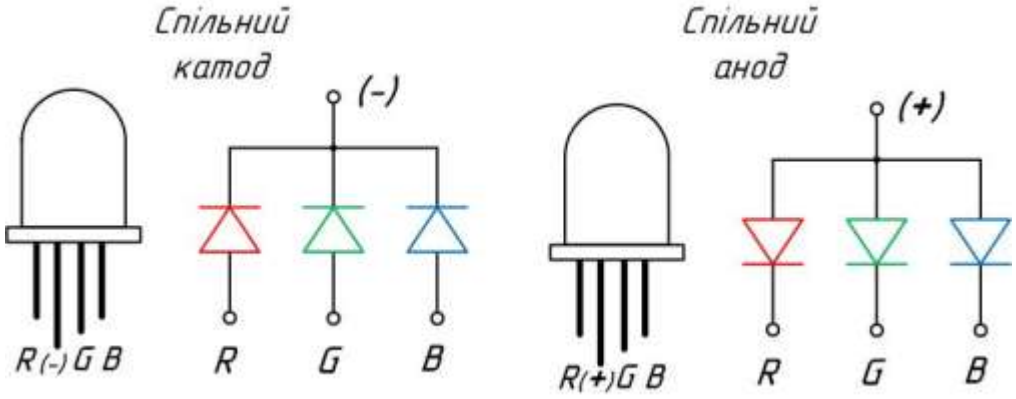


Рисунок 2.10 – «Розпіновка» типових RGB світлодіодів

До Arduino такий світлодіод під’єднується так само, як якщо б ми під’єднували три окремі світлодіоди: на кожен колір потрібний струмообмежувальний резистор, а спільний вивід потрібно під’єднувати залежно від того, анод це чи катод (рис. 2.11).

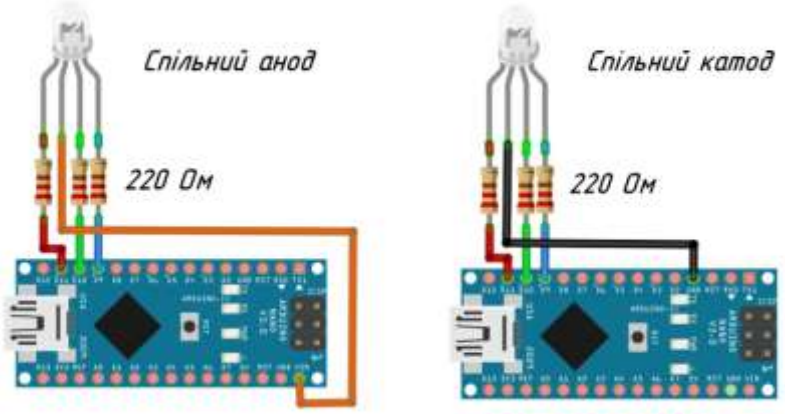


Рисунок 2.11 – Під’єднання RGB-світлодіодів до плати Arduino

Можна керувати кожним кольором так само, якби це були окремі світлодіоди. Також не забуваємо про під’єднання: якщо світлодіод має загальний катод, то високий сигнал (digitalWrite (pin, HIGH);) з пінів керування вмикатиме вибраний колір, а якщо загальний анод – то вимикати. Відповідно, плавне управління яскравістю за допомогою ШІМ працює за тією ж логікою: у загального катода, analogWrite (pin, value); увімкне колір майже на повну яскравість, а у загального анода – майже повністю погасить.

Під’єднайте RGB світлодіодом до таких пінів: Мінус – GND, B – Pin 9, G – Pin10, R – Pin 11 (рис. 2.11). Після під’єднання світлодіода та складання схеми на Arduino завантажте скетч у плату.

Приклад 2.2

```
                  // Плавне миготіння RGB світлодіода
#define RED 11    // присвоюємо ім’я RED для піна 11
#define GRN 10    // присвоюємо ім’я GRN для піна 10
#define BLU 9     // присвоюємо ім’я BLU для піна 9

void  setup () {
  pinMode (RED, OUTPUT );  // Pin11 для виведення
  pinMode (GRN, OUTPUT );  // Pin10 для виведення
  pinMode (BLU, OUTPUT );  // Pin9 для виведення
}

void  loop () {
  // плавне вмикання/вимикання червоного кольору
  for ( int i = 0; i <= 255; i++) {
```

```
    analogWrite (RED, i);
    delay (2);
  }
  for ( int i = 255; i >= 0; i--) {
    analogWrite (RED, i);
    delay (2);
  }

  // плавне вмикання/вимикання зеленого кольору
  for ( int i = 0; i <= 255; i++) {
    analogWrite (GRN, i);
    delay (2);
  }
  for ( int i = 255; i >= 0; i--) {
    analogWrite (GRN, i);
    delay (2);
  }

  // плавне вмикання/вимикання синього кольору
  for ( int i = 0; i <= 255; i++) {
    analogWrite (BLU, i);
    delay (2);
  }
  for ( int i = 255; i >= 0; i--) {
    analogWrite (BLU, i);
    delay (2);
  }
}
```

2.3 Однорозрядний семисегментний індикатор та платформа Arduino

Семисегментний індикатор має вісім світлодіодів, які розташовані у формі вісімки. Він простий у використанні та економічний щодо вартості. На рисунку 2.12 показано типовий семисегментний індикатор [17].

Рисунок 2.12 – Семисегментний індикатор

Семисегментні індикатори використовуються для відображення цифр (0–9) та деяких літер алфавіту (наприклад, C, A, H, P тощо).

7 цих світлодіодних сегментів мають форму лінії, тоді як 1 сегмент має круглу форму і використовується для десяткової точки.

Кожен з 8-ми елементів з'єднаний з контактом, через який відбувається керування станом ВИСОКИЙ (HIGH) «1» або НИЗЬКИЙ (LOW) «0» рівень напруги.

Щоб відобразити число або літеру, потрібно ввімкнути певні світлодіодні сегменти дисплея. Приклад шрифту для відображення повідомлення наведено на рисунку 2.13 [18].

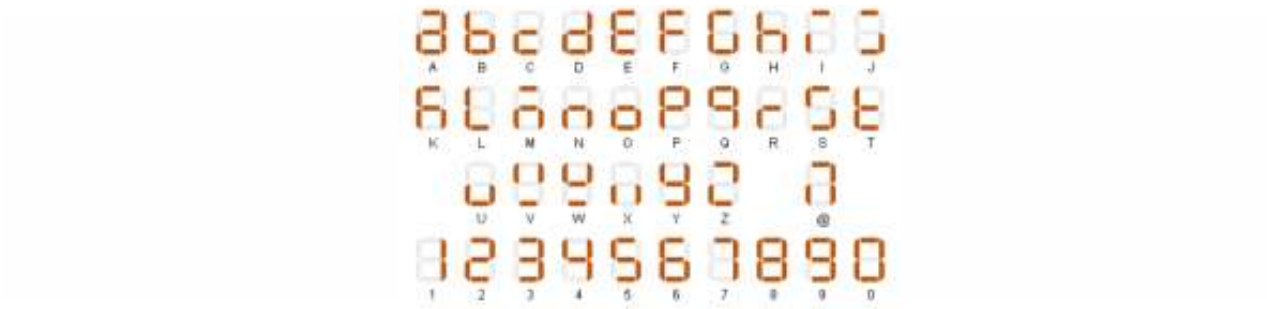


Рисунок 2.13 – Шрифт семисегментного індикатора

Семисегментні індикатори бувають двох типів: із спільним анодом (СА) та спільним катодом (СС). Внутрішня структура обох типів майже однакова. Різниця полягає в полярності світлодіодів і спільних виводів. У семисегментному індикаторі зі спільним катодом (такий ми використовуємо в роботі) катоди всіх семи світлодіодів та світлодіода для десяткової точки під'єднані до виводів 3 та 8. Щоб використовувати такий індикатор, нам потрібно під'єднати корпус до виводів 3 та 8 та подати +5 В на інші виводи, щоб окремі світлодіоди засвітились. На рисунку 2.14 показано внутрішню структуру семисегментного індикатора зі спільним катодом.

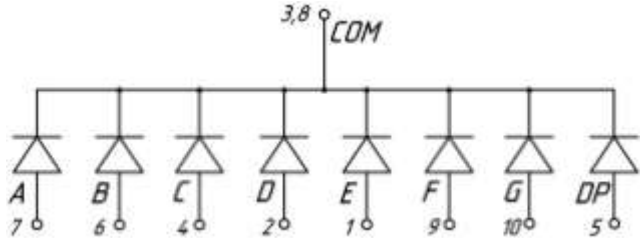


Рисунок 2.14 – Семисегментний індикатор із спільним катодом

Індикатор із спільним анодом є повною протилежністю індикатора зі спільним катодом. В індикаторі зі спільним анодом плюсові виводи всіх восьми світлодіодів з'єднані разом і під'єднані до виводів 3 і 8. Щоб засвітити окремий сегмент, потрібно під'єднати інший вивід до «землі». На рисунку 2.15 показано внутрішню структуру семисегментного індикатора зі спільним анодом.

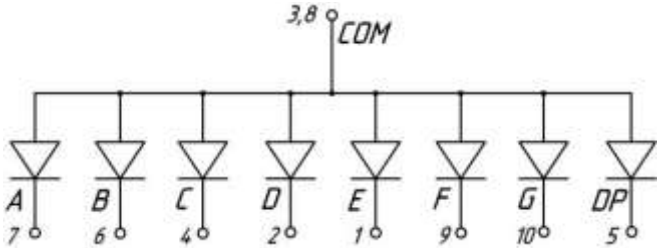


Рисунок 2.15 – Семисегментного індикатора зі спільним анодом

Щоб відобразити конкретну цифру, потрібно ввімкнути окремі сегменти, як показано нижче (табл. 2.1).

Таблиця 2.1 – Ввімкнення сегментів індикатора для відображення цифр

Цифра	***A***	***b***	***c***	***d***	***e***	***F***	***g***
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	0
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Для під’єднання однорозрядного семисегментного індикатора до Arduino потрібно задіяти 7 цифрових виводів, кожен з контактів a–g індикатора під’єднується до виводу Arduino через обмежувальний резистор 220…470 Ом. У лабораторній роботі ми використовуємо семисегментний індикатор із загальним катодом ОК, загальний провід приєднуємо до землі.

На рисунку 2.16 показано схему під’єднання однорозрядного семисегментного індикатора до плати Arduino.

а) б)

Рисунок 2.16 – Схема під’єднання однорозрядного семисегментного індикатора до плати Arduino: а – в середовищі Tinkercad; б – за допомогою плати розширення

Приклад 2.3

```
int i;
void setup() {
  pinMode(14, OUTPUT);
  pinMode(16, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(2, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(12, OUTPUT);

  Serial.begin (9600);
}
//            A   B   C  D  E  F   G  DP
int M[6] = { 14, 16, 10, 6, 8, 2};
int N[8] = { 14, 16, 10, 6, 8, 2,  4, 12};
        // A  B  C  D  E  F  G   DP
int q[] = {0, 1, 1, 0, 0, 0, 0,  0};   // сегменти "1"
int w[] = {1, 1, 0, 1, 1, 0, 1,  0};   // сегменти "2"
int r[] = {1, 1, 1, 1, 0, 0, 1,  0};   // сегменти "3"
int t[] = {0, 1, 1, 0, 0, 1, 1,  0};   // сегменти "4"
int y[] = {1, 0, 1, 1, 0, 1, 1,  0};   // сегменти "5"
int u[] = {1, 0, 1, 1, 1, 1, 1,  0};   // сегменти "6"
int o[] = {1, 1, 1, 0, 0, 0, 0,  0};   // сегменти "7"
int p[] = {1, 1, 1, 1, 1, 1, 1,  0};   // сегменти "8"
int s[] = {1, 1, 1, 1, 0, 1, 1,  0};   // сегменти "9"
int k[] = {1, 1, 1, 1, 1, 1, 0,  0};   // сегменти "0"
int l[] = {0, 0, 0, 0, 0, 0, 0,  0};   //сегменти вимкн.
int c[] = {0, 0, 0, 0, 0, 0, 0,  1};   // сегмент крапка

void loop() {
  function_KRUG();
  function_1();
  delay(1000);
  function_KRUG();
  function_2();
  delay(1000);
  function_off();
  function_KRUG();
  function_3();
  delay(1000);
  function_off();
  function_KRUG();
  function_4();
  delay(1000);
  function_off();
```

```
  function_KRUG();
  function_5();
  delay(1000);
  function_off();
  function_KRUG();
  function_6();
  delay(1000);
  function_off();
  function_KRUG();
  function_7();
  delay(1000);
  function_KRUG();
  function_8();
  delay(1000);
  function_off();
  function_KRUG();
  function_9();
  delay(1000);
  function_off();
  function_KRUG();
  function_0();
  delay(1000);
  function_off();
  delay(1000);
  function_ptn();
  Serial.println(i);
}

void function_KRUG() {
  for (int j = 0; j < 2; j++) {
    for (int i = 0; i < 7; i++) {
      digitalWrite((M[i]), HIGH);
      delay(40);
      digitalWrite((M[i - 1]), LOW);
      delay(40);
    }
  }
}

void function_1() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (q[i]));
  }
}

void function_2() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (w[i]));
  }
```

```
}
void function_3() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (r[i]));
  }
}
void function_4() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (t[i]));
  }
}
void function_5() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (y[i]));
  }
}
void function_6() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (u[i]));
  }
}
void function_7() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (o[i]));
  }
}
void function_8() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (p[i]));
  }
}
void function_9() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (s[i]));
  }
}
void function_0() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (k[i]));
  }
}
void function_off() {
  for (int i = 0; i < 8; i++) {
    digitalWrite((N[i]), (l[i]));
  }
}
void function_ptn() {
  for (int j = 0; j < 3; j++) {
```

```
      for (int i = 0; i < 8; i++) {
        digitalWrite((N[i]), (c[i]));
      }
      delay(500);
      for (int i = 0; i < 8; i++) {
        digitalWrite((N[i]), (l[i]));
      }
      delay(500);
    }
  }
```

2.3 Порядок виконання роботи

1. Розглянути основні теоретичні відомості лабораторної роботи. Проаналізувати схеми та методи під'єднання одноколірних світлодіодів, RGB світлодіодів та семисегментних LED-індикаторів до плати Arduino Uno R3.

2. Скласти схему відповідно до схеми, наведеної на рисунку 2.7, а в середовищі Tinkercad, або використовуючи плату розширення (див. рис. 2.7, б) та наведений у прикладі 2.1 код, дослідити роботу складеної схеми.

3. Скласти схему відповідно до схеми, наведеної на рисунку 2.9, в середовищі Tinkercad або використовуючи плату розширення рисунка 2.7 (б), згідно з варіантом лабораторної роботи, здійснити керування яскравістю світлодіодів за допомогою ШІМ.

4. Скласти схему за рис. 2.1 та з урахуванням варіанта лабораторної роботи (див. табл. 2.1), вивести на RGB світлодіод задані кольори. За основу взяти код, наведений у прикладі 2.2.

5. Використовуючи плату розширення (див. рис. 2.16, б) або у середовищі Tinkercad скласти схему (див. рис. 2.16, а), почергово вивести літери свого прізвища, враховуючи запропонований шрифт сегментного індикатора (див. рис. 2.13) та код, наведений у прикладі 2.3.

2.4 Зміст звіту

1. Звіт з лабораторної роботи має бути виконаний на аркушах формату А4.

2. Звіт має містити: назву лабораторної роботи, її мету і короткі теоретичні відомості.

3. В розділі «Результати виконання лабораторної роботи» студент має навести необхідні розрахунки, побудувати відповідні графіки, нарисувати: схему електричну принципову, схему електричну комутаційну, показати алгоритм реалізації, а також навести код програмної реалізації.

4. Написати висновки до цієї лабораторної роботи.

5. Оформити звіт та подати його викладачеві на захист.

Таблиця 2.2 – Варіанти завдань

Варіант	Керувати яскравістю світлодіода на портах:	Тип RGB світлодіода	Колір світіння RGB світлодіода					
			перший колір			другий колір		
			R	**G**	**B**	**R**	**G**	**B**
1	P3, P5	CA	219	219	25	64	134	148
2	P6, P9	CK	48	5	190	34	32	186
3	P10, P11	CA	135	135	30	245	250	250
4	P3, P6,	CK	219	219	219	135	135	135
5	P5, P9	CA	134	9	217	31	219	219
6	P6, P10	CK	135	85	135	134	9	217
7	P9, P11	CA	219	219	46	34	135	135
8	P3, P10	CK	250	253	30	134	9	217
9	P5, P11	CA	135	135	200	135	135	30
10	P10, P11	CK	134	9	217	48	5	190
11	P3, P6,	CA	46	34	157	246	32	186
12	P5, P9	CK	30	245	135	250	250	253
13	P6, P10	CA	219	135	31	48	55	210
14	P9, P11	CK	253	31	246	219	219	46
15	P3, P10	CA	135	246	25	253	253	30
16	P3, P5	CK	134	9	217	135	165	219
17	P6, P9	CA	31	135	125	219	120	253
18	P10, P11	CK	246	219	146	34	204	135
19	P3, P6,	CA	25	253	253	31	219	134
20	P5, P9	CK	64	134	135	246	250	32
21	P6, P10	CA	48	5	190	34	135	250
22	P9, P11	CK	253	31	219	250	250	135
23	P3, P6,	CA	135	246	250	135	135	219
24	P5, P9	CK	46	34	135	32	186	250
25	P6, P3	CA	30	250	125	48	5	190
26	P9, P11	CK	200	55	135	253	31	219
27	P3, P5	CA	134	9	217	135	246	250
28	P5, P11	CK	48	5	190	46	34	135
29	P10, P5	CA	135	135	135	30	150	250
30	P3, P6,	CK	204	7	38	200	35	135

Контрольні запитання

1. Як трактувати поняття нормально розімкнена (NO) та нормально замкнена (NC) кнопка?
2. Наведіть схему під'єднання кнопки з підтягування до «землі».
3. Наведіть схему під'єднання кнопки з підтягування до V_{CC}.
4. Що таке внутрісхемне підтягування кнопки до шини живлення?
5. Що таке брязкіт контактів?
6. Яким чином можна уникнути наслідків брязкоту контактів?
7. Що таке світлодіод, яка його будова?
8. Як розрахувати струмообмежувальний опір для світлодіода?
9. Наведіть схему ввімкнення світлодіода з СА.
10. Наведіть схему ввімкнення світлодіода з СК.
11. Що таке широтно-імпульсна модуляція (ШІМ)?
12. За допомогою чого здійснюється керування яскравістю світлодіодів?
13. Які порти платформи Arduino UNO підтримують ШІМ?
14. Що таке RGB світлодіод?
15. Яку будову має семисегментний індикатор?

Лабораторна робота № 3

СХЕМОТЕХНІЧНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ПІДКЛЮЧЕННЯ ДЕКІЛЬКОХ КНОПОК ДО ОДНОГО АНАЛОГОВОГО ВХОДУ МІКРОКОНТРОЛЕРА

Мета роботи: Здійснити схемотехнічну та програмну реалізацію під'єднання заданої кількості тактових кнопок з використанням лише одного порту мікроконтролера.

3 Основні теоретичні відомості

3.1 Схема подільника напруги на резисторах

Схема подільника напруги містить вхідне джерело напруги і два резистори. Нижче ви можете побачити кілька схематичних варіантів зображення подільника, але всі вони несуть один і той самий функціонал [19].

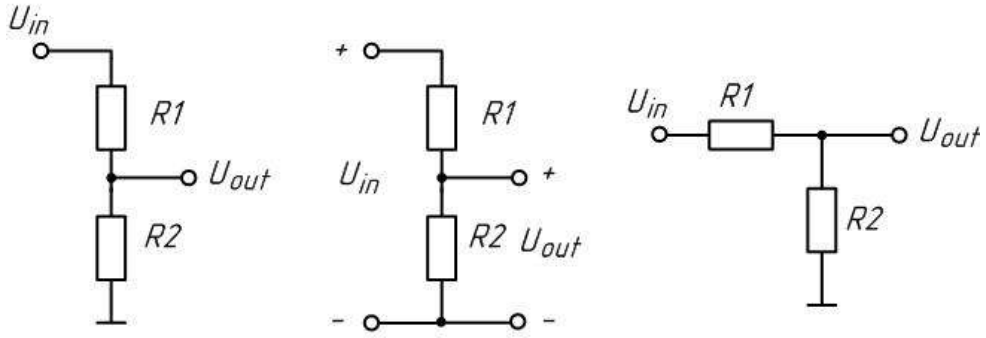


Рисунок 3.1 – Подільник напруги

Позначимо резистор, який знаходиться ближче до плюса вхідної напруги (U_{in}), як R1, а резистор, що знаходиться ближче до мінуса, як R_2. Спад напруги (U_{out}) на резисторі R_2 – це знижена напруга, отримана в результаті застосування резисторного подільника напруги.

3.2 Під'єднання кількох кнопок до одного аналогового входу

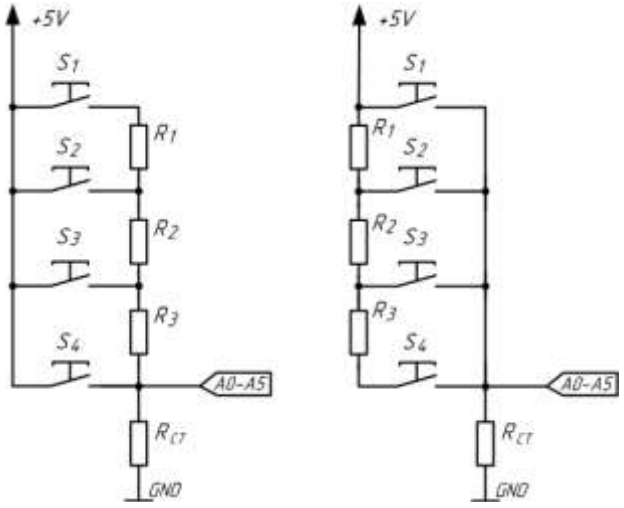
Розглянемо дві різні схеми під'єднання великої кількості кнопок до одного аналогового входу мікроконтролера. Принцип роботи схем заснований на зчитуванні та інтерпретації аналого-цифровим перетворювачем

мікроконтролера індивідуальної напруги, що формується різними комбінаціями окремих ділянок схеми.

Обидві схеми, незалежно від кількості кнопок (в розумних межах), в кожний активний момент (натискання) часу є ключовим резистивним подільником напруги. Але важливі відмінності та особливості роботи кожної з них мають різні алгоритми розрахунку елементів та кінцеву поведінку.

3.2.1 Резистивно-послідовна схема під'єднання

Принципово схема може виконуватися в 2 варіантах і виглядає так [20]:



варіант А варіант Б

Рисунок 3.2 – Резистивно-послідовна схема під'єднання

Відмінності схем полягають у тому, що при натисканні кнопки S_4 (рис. 3.2, варіант А) на аналоговий вхід надійде максимальна напруга 5 В, а кнопки S_1 – мінімальна напруга, знижена всіма резисторами кола. У схемі (рис. 3.2, варіант Б), навпаки, максимальна напруга буде, натискаючи кнопку S_1.

Натискаючи будь-яку кнопку у схемі ми отримуємо класичний резистивний подільник напруги (рис. 3.3).

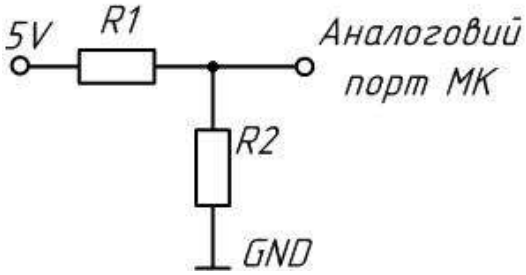


Рисунок 3.3 – Приклад подільника напруги

Залежність значення напруги і опорів виражається формулою

$$\text{вих} = \text{вх} \times \frac{R_2}{R_1+R_2}.$$

З формули видно, що залежність вихідної напруги $U_{вих}$ від опорів R_1 і R_2 не лінійна, а екпоненційна. Графік це підтверджує. Єдиний нюанс полягає в тому, що на осі абсцис номінали резистора R_1 наведені в номіналах «стягувального» резистора R_2, $X = R_2$.

Наприклад, якщо $R_2 = 10$ *кОм*, то при $R_1 = 3 \times R_2 = 30$ *кОм*, $U_{вих} = 1{,}25$ *В*.

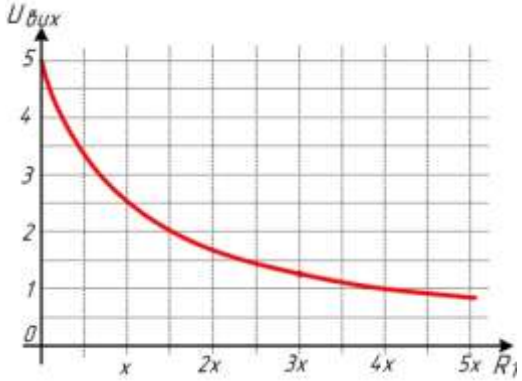
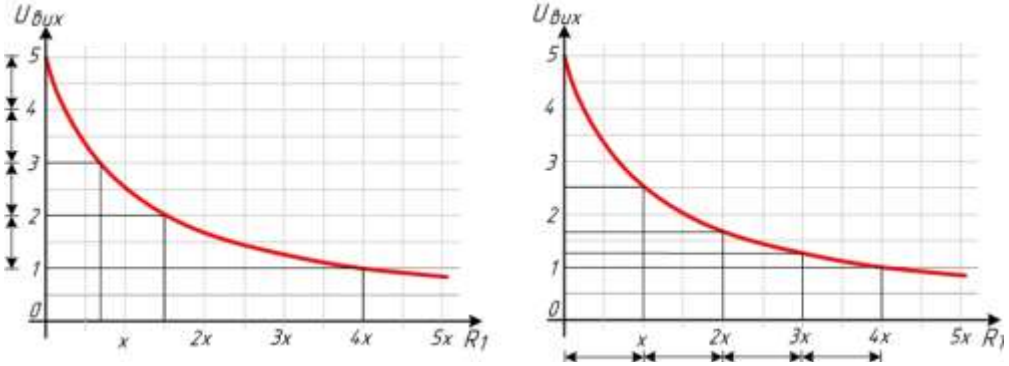


Рисунок 3.4 – Залежність вихідної напруги $U_{вих}$ від опорів R_1 і R_2

Виходячи з графіка можна наочно уявити, яким чином розрахувати номінали резисторів залежно від кількості кнопок. І тут, як завжди, можливі 2 варіанти: або підбирати номінали так, щоб розбити інтервал на рівні проміжки напруги, або використовувати в схемі резистори тільки одного номіналу.



а) б)

Рисунок 3.5– Приклад використання 4 резисторів для 5 кнопок:
а) резистори підібрані під рівномірну дискретність напруги;
б) резистори одного номіналу

У першому випадку (див. рис. 3.5, а) номінали резисторів підібрані таким чином, щоб забезпечити рівні інтервали при визначенні значень напруги АЦП. Такий підхід дуже зручний для кількості кнопок 2^n (n⩾2), тобто 4, 8, 16, ... Зручність полягає в тому, що результат роботи АЦП за допомогою бітового зсуву можна округлити, при цьому максимально нівелювати похибки АЦП і номіналів резисторів.

Недолік такого способу полягає в обмеженні кількості кнопок, коли їх число збільшується, номінали резисторів кнопок, що відповідають за низьку напругу (менше 1 В), зростають у геометричній прогресії.

Формула розрахунку номіналів резисторів для встановлення рівних проміжків напруг для кількості кнопок k та прив'язаних до номіналу стягувального резистора $R_{\text{ст}}$ (якщо прийняти $R_{\text{ст}}$ за одиницю, то отримані коефіцієнти можна використовувати для розрахунку опорів при іншому заданому підтягувальному опорі) виглядає так:

$$R_n = \frac{U_{\text{вх}} \times R_{CT}}{u \times (k-n)} - R_{CT} - \sum_{i=1}^{n-1} R_i,$$

де $u = \frac{U_{\text{вх}}}{k}$ – крок дискретності вихідної напруги

Оскільки перша кнопка замикається без резистора, то резистор нульового опору опускаємо, звідси $n \leq k-1$. Ця кнопка дасть результат рівний $_{\text{вх}} - 5$ В. Розглянемо розрахунок номіналів на прикладі наведених 5 кнопок і 4 резисторів, стягувальний резистор $R_{\text{ст}} = 10$ кОм (отриманий результат буде в кОм):

$$u = \frac{\text{вх}}{k} = \frac{5}{5} = 1 \text{ В}.$$

$$R_1 = \frac{5 \text{ В} \times 10 \text{ кОм}}{1\text{В} \times (5-1)} - 10 \text{ кОм} - 0 \text{ кОм} = 2{,}5 \text{ кОм};$$

$$R_2 = \frac{5 \text{ В} \times 10 \text{ кОм}}{1\text{В} \times (5-2)} - 10 \text{ кОм} - (2{,}5 \text{ кОм} + 0 \text{ кОм}) = 4{,}167 \text{ кОм};$$

$$R_3 = \frac{5 \text{ В} \times 10 \text{ кОм}}{1\text{В} \times (5-3)} - 10 \text{ кОм} - (4{,}167 \text{ кОм} + 2{,}5 \text{ кОм} + 0 \text{ кОм}) = 8{,}333 \text{ кОм};$$

$$R_4 = \frac{5 \text{ В} \times 10 \text{ кОм}}{1\text{В} \times (5-4)} - 10 \text{ кОм} - (8{,}333 \text{ кОм} + 4{,}167 \text{ кОм} + 2{,}5 \text{ кОм} + 0 \text{ кОм}) = 25 \text{ кОм}.$$

Результат: нам знадобляться резистори таких номіналів: 2.5 кОм, 4.167 кОм, 8.333 кОм, 25 кОм.

Для другого випадку (див. рис. 3.5, б) все набагато простіше – встановлені резистори одного номіналу. Однак краще, якщо цей номінал буде менше номіналу резистора, що стягує *X*. У цьому випадку перші кнопки потраплять у діапазон, де відмінності між напругами максимальні.

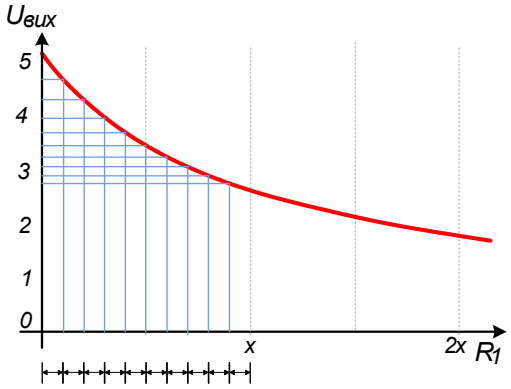


Рисунок 3.6 – 10 резисторів, номіналом *0.1×X* (при номіналі стягувального резистора 10 кОм, номінал резисторів кнопок – 1 кОм)

Діапазон напруг знаходиться за формулою

$$\text{вих}\, n = \text{вх} \times \frac{R_{CT}}{R \times n + R_{CT})}.$$

Але, як видно з графіка, дискретність напруг неминуче знижується. В такому випадку можна піти на хитрість – там, де різниця між напругами стає критично мінімальною, почати встановлювати резистори більшого номіналу, збільшивши дискретність.

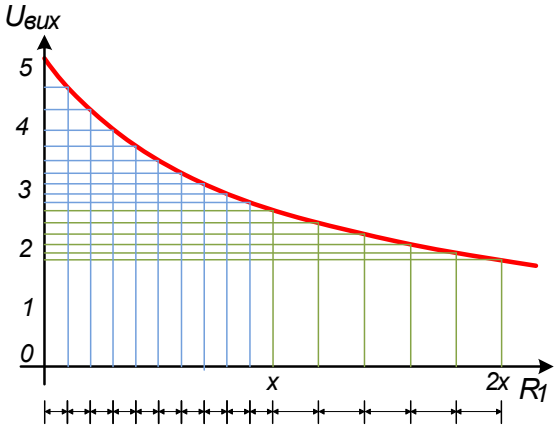


Рисунок 3.7 – Зниження дискретності вихідної напруги $U_{вих}$

Основною відмінністю резистивно-послідовної схеми є ігнорування натискання кількох кнопок – результат завжди буде вказувати на натискання єдиної кнопки з меншим, відносно інших натиснутих кнопок, сумарним опором.

Переваги резистивно-послідовної схеми:

- передбачувана поведінка;
- порівняно легкий розрахунок;
- натискання кількох кнопок одночасно приводить до передбачуваного результату;
- варіанти реалізації – однаковий крок зміни вихідної напруги на виході або однакові номінали резисторів, що використовуються.

Недоліки резистивно-послідовної схеми:

- накопичувана похибка сумарного опору використовуваних резисторів;
- для першого випадку (рівномірна дискретність значень напруги) – складність розрахунків;
- обмеження кількості кнопок в діапазоні напруги до 1 В;
- важливий порядок розташування резисторів на схемі.

3.2.2 Резистивно-паралельна схема під'єднання

Принципова схема може виглядає так:

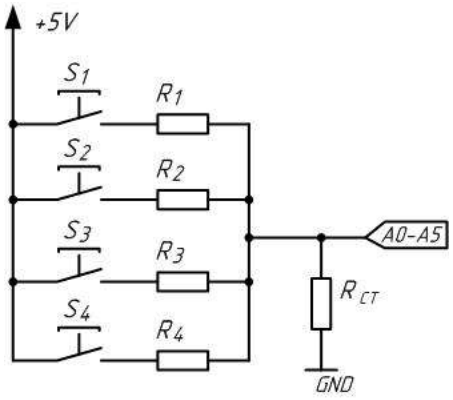


Рисунок 3.8 – Резистивно-паралельна схема під'єднання

Особливість даної схеми полягає в тому, що резистор кожної кнопки задає потрібну напругу на виході – його номінал розраховується так, як і в першій схемі. Схема не допускає використання резисторів одного номіналу. У разі натискання двох або декількох кнопок генерується індивідуальна напруга, яка визначається, виходячи з паралельного під'єднання опорів.

Ця особливість (розпізнавання одночасно натиснутих кнопок) є перевагою, при їх малій кількості, і в той же час, робить поведінку схеми складно передбачуваною при великій кількості цих кнопок.

$$R_n = \frac{\text{вх} \times R_{CT}}{\text{вих}\, n} - R_{CT}.$$

Наприклад, у нас є схема з 3 кнопок, які мають генерувати, відповідно, 1, 2.5 і 4 В. Номінал стягувального резистора 10 кОм, вхідна напруга 5 В. Розрахунок номіналів резисторів буде таким:

$$R_1 = \frac{5\text{В} \times 10\ \text{кОм}}{1\text{В}} - 10\ \text{кОм} = 40\ \text{кОм}$$

$$R_2 = \frac{5\text{В} \times 10\ \text{кОм}}{2{,}5\ \text{В}} - 10\ \text{кОм} = 10\ \text{кОм}$$

$$R_3 = \frac{5\text{В} \times 10\ \text{кОм}}{4\text{В}} - 10\ \text{кОм} = 2{,}5\ \text{кОм}$$

Переваги резистивно-паралельної схеми:

- простота підбору номіналів,
- не важливий порядок встановлення резисторів – кожен номінал під'єднується до схеми з відповідною кнопкою,
- можливе нарощування додаткового функціонала за рахунок інтерпретацій комбінацій натискань.

Недоліки резистивно-паралельної схеми:

- підходить тільки для малої кількості кнопок (до 5) –якщо в схему під'єднується більше 5 кнопок, поведінка схеми при одночасному натисканні двох і більше кнопок стає непередбачуваною.

Приклад 3.1

```
int analogPin = A0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  int val = analogRead(analogPin);
  if ((val > 50) && (val < 150))
  {
    Serial.println("Button 8 pressed");
    delay(100);
  }
  else if ((val > 150) && (val < 300))
  {
    Serial.println("Button 7 pressed");
    delay(100);
  }
  else if ((val > 300) && (val < 420))
  {
    Serial.println("Button 6 pressed");
```

```
    delay(100);
  }
  else if ((val > 420) && (val < 550))
  {
    Serial.println("Button 5 pressed");
    delay(100);
  }
  else if ((val > 550) && (val < 720))
  {
    Serial.println("Button 4 pressed");
    delay(100);
  }
  else if ((val > 720) && (val < 850))
  {
    Serial.println("Button 3 pressed");
    delay(100);
  }
  else if ((val > 850) && (val < 920))
  {
    Serial.println("Button 2 pressed");
    delay(100);
  }
  else if (val > 920)
  {
    Serial.println("Button 1 pressed");
    delay(100);
  }
  else
  {
    Serial.println("button not pressed");
    delay(100);
  }
}
```

3.3 Порядок виконання роботи

1. Розглянути основні теоретичні відомості резистивних подільників напруги та схеми реалізації резистивно-послідовних та резистивно-паралельних схем під'єднання тактових кнопок на аналоговий порт мікроконтролера.
2. Згідно з варіантом лабораторної роботи здійснити потрібні розрахунки елементів схеми та реалізувати задану схему й алгоритм її роботи, використовуючи макетну плату та Arduino Uno R3 (за потреби використовувати емулятор Arduino-Tinkercad).
3. Реалізувати розроблений в пункті 2 алгоритм з урахуванням прикладу реалізації (див. приклад 3.1), а допомого інтегрованого середовища розробки

Arduino IDE для плати Arduino UNO R3 (За потреби використовувати емулятор Arduino-Tinkercad).

3.4 Зміст звіту

1. Звіт з лабораторної роботи має бути виконаний на аркушах формату А4.
2. Звіт має містити: назву лабораторної роботи, її мету і короткі теоретичні відомості.
3. В розділі «Результати виконання лабораторної роботи» студент має навести потрібні розрахунки, побудувати відповідні графіки, нарисувати схему електричну принципову, схему електричну комутаційну, навести алгоритм реалізації, а також код програмної реалізації.
4. Написати висновки до даної лабораторної роботи.
5. Оформити звіт та подати його викладачу на захист.

Таблиця 3.1 – Варіанти завдань

Варіант	Кількість кнопок	Тип схеми	Забезпечити	порт МК
1	12	Резистивно-паралельна	Відклик на комбінації натискання декількох кнопок	A0
2	4	Резистивно-паралельна		A1
3	5	Резистивно-паралельна		A2
4	4	Резистивно-паралельна		A3
5	9	Резистивно-паралельна		A4
6	7	Резистивно-паралельна		A5
7	5	Резистивно-паралельна		A0
8	9	Резистивно-паралельна		A1
9	9	Резистивно-паралельна		A2
10	10	Резистивно-паралельна		A3
11	5	Резистивно-послідовна Б	дискретність напруги	A4
12	6	Резистивно-послідовна Б	Опір одного номіналу	A5
13	5	Резистивно-послідовна Б	дискретність напруги	A0
14	6	Резистивно-послідовна Б	Опір одного номіналу	A1
15	7	Резистивно-послідовна Б	дискретність напруги	A2
16	8	Резистивно-послідовна Б	Опір одного номіналу	A3
17	11	Резистивно-послідовна Б	дискретність напруги	A4
18	6	Резистивно-послідовна Б	Опір одного номіналу	A5

Продовження таблиці 3.1

19	7	Резистивно-послідовна Б	дискретність напруги	A0
20	8	Резистивно-послідовна Б	Опір одного номіналу	A1
21	10	Резистивно-послідовна А	дискретність напруги	A2
22	9	Резистивно-послідовна А	Опір одного номіналу	A3
23	6	Резистивно-послідовна А	дискретність напруги	A4
24	7	Резистивно-послідовна А	Опір одного номіналу	A5
25	8	Резистивно-послідовна А	дискретність напруги	A0
26	10	Резистивно-послідовна А	Опір одного номіналу	A1
27	6	Резистивно-послідовна А	дискретність напруги	A2
28	12	Резистивно-послідовна А	Опір одного номіналу	A3
29	6	Резистивно-послідовна А	дискретність напруги	A4
30	7	Резистивно-послідовна А	Опір одного номіналу	A5

Контрольні запитання

1. Що таке дискретність напруги?
2. Опишіть принцип роботи аналогового порту.
3. Що таке подільник напруги?
4. Як обрати потужність резисторів резистивного подільника напруги?
5. Що таке внутрісхемне «підтягування» порту до шини живлення «pull UP»?
6. Наведіть резистивно-паралельну схему під’єднання кнопок.
7. Наведіть резистивно-послідовну схему під’єднання кнопок.
8. Які переваги резистивно-паралельної схеми під’єднання?
9. Які переваги резистивно-послідовної схеми під’єднання?
10. Які недоліки резистивно-паралельної схеми під’єднання?
11. Які недоліки резистивно-послідовної схеми під’єднання»?
12. Що таке дискретність напруги на резистивному подільнику?
13. Як ви розумієте поняття «підтягувальний резистор»?

Лабораторна робота № 4

ДОСЛІДЖЕННЯ ПРИНЦИПУ РОБОТИ КОМПАРАТОРІВ

Мета роботи: дослідити принцип роботи цифрового компаратора. реалізувати схему сутінкового автомата на основі компаратора аналогових сигналів.

4 Основні теоретичні відомості

4.1 Компаратор аналогових сигналі

Компаратор аналогових сигналів (від лат. *comparare* «порівнювати») – порівнювальний пристрій: електронна схема, що приймає на свої входи два аналогових сигнали і видає сигнал високого рівня, якщо сигнал на неінвертувальному вході («+») більший, ніж на інвертувальному (інверсному) вході («-»), і сигнал низького рівня, якщо сигнал неінвертувальному вході, менший, ніж на інвертувальному вході. Значення вихідного сигналу компаратора за рівності вхідної напруги у такому випадку не визначено. Зазвичай у логічних схемах сигналу високого рівня приписується значення логічного 1, а низькому – логічного 0 [22].

Через компаратори здійснюється зв’язок між неперервними сигналами, наприклад, напруги та логічними змінними цифрових пристроїв.

Застосовуються у різних електронних пристроях, АЦП та ЦАП, пристроях сигналізації, контролю доступу тощо.

Одну з напруг (сигналів), що подається на один із входів компаратора, зазвичай називають опорною або пороговою напругою. Порогова напруга поділяє весь діапазон вхідної напруги, що подається на інший вхід компаратора, на два піддіапазони. Стан виходу компаратора, високий або низький рівень, вказує, в якому з двох піддіапазонів знаходиться вхідна напруга. Компаратор з однією вхідною пороговою напругою прийнято називати однопороговим компаратором, існують компаратори з двома або декількома пороговими напругами, які, відповідно, ділять діапазон вхідної напруги на число піддіапазонів на 1 більше числа порогів.

Порівнюваний сигнал може подаватися як на інвертувальний, так і на неінвертувальний входи компаратора. Відповідно, залежно від цього, компаратор називають інвертувальним або неінвертувальним.

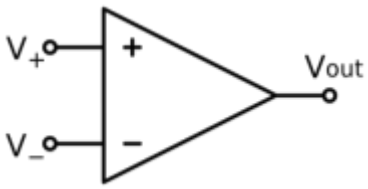


Рисунок 4.1 – Символічне зображення аналогового компаратора на електричних та структурних схемах

Математичний опис компаратора

В аналітичному вигляді ідеальний однопороговий неінвертувальний компаратор задається нижченаведеною системою нерівностей:

$$_{out} = \begin{cases} 0, & \text{if} \quad U_{in} < {}_{ref,} \\ \text{не визнач.} & \text{if} \quad U_{in} = {}_{ref}, \\ 1, & \text{if} \quad U_{in} > {}_{ref,} \end{cases}$$

де $_{ref}$ – напруга порога порівняння,

$_{out}$ – вихідна напруга компаратора,

U_{in} – вхідна напруга на сигнальному вході компаратора.

Третьому, невизначеному значенню, у разі бінарного стану виходу можна:

- присвоїти U_0 або U_1;
- присвоїти U_0 або U_1 випадковим чином динамічно;
- враховувати попередній стан виходу та вважати рівність недостатньою для перемикання,
- враховувати першу похідну за часом вихідного сигналу та її рівність нулю вважати недостатньою для перемикання.

У разі використання багатозначної логіки, наприклад, потрійної для реєстрації третього стану (рівність), застосувати відповідну потрійну функцію з чіткої потрійної логіки з точним третім значенням.

Схемотехнічно найпростіший компаратор є диференційним підсилювачем з високим коефіцієнтом підсилення (в ідеалі – нескінченним). Зазвичай як компаратори напруги в сучасній електроніці застосовують мікросхеми операційних підсилювачів (ОП). Але існують і випускаються спеціалізовані мікросхеми для використання їх як компараторів.

Мікросхема компаратора відрізняється від звичайного лінійного (ОП) схемою і вхідного, і вихідного каскадів.

Вхідний каскад компаратора має витримувати широкий діапазон диференційних вхідних напруг (між інвертувальним і неінвертувальним

входами), аж до значень напруги живлення, а також повний діапазон синфазних напруг.

Вихідний каскад компаратора зазвичай конструюють сумісним за логічними рівнями та струмами з поширеним типом входів логічних схем (технологій ТТЛ, ЕПЛ (емітерно пов'язана логіка). Можливі виконання вихідного каскаду компаратора на окремому транзисторі з відкритим колектором, що забезпечує одночасну сумісність з ТТЛ і з КМОП логічними мікросхемами.

Мікросхеми компараторів не розраховані для роботи з негативним зворотним зв'язком як ОУ, та при їх застосуванні негативний зворотний зв'язок не використовується. І навпаки, для формування гістерезисної передавальної характеристики компаратори часто використовують позитивний зворотний зв'язок. Цей захід дозволяє уникнути швидких небажаних перемикань стану виходу, зумовленому шумами у вхідному сигнал в разі вхідного сигналу, що повільно змінюється.

При проєктуванні мікросхем компараторів приділяється особлива увага швидкому відновленню вхідного каскаду після перевантаження та зміни знака різниці вхідної напруги. У швидкодійних компараторах підвищення швидкодії схемотехнічно не допускає заходу біполярних транзисторів у вихідному каскаді в режим насичення.

Компаратори з позитивним зворотним зв'язком мають гістерезис і, насправді, є двопороговими компараторами. Часто такий компаратор називають тригером Шмітта.

При рівності вхідних напруг реальні компаратори і ОУ, під'єднані за схемою компараторів, дають вихідний сигнал, що хаотично змінюється через власні шуми і шуми вхідних сигналів. Звичайна міра пригнічення такої хаотичності – запровадження позитивного зворотного зв'язку для отримання гістерезисної передавальної характеристики.

У разі програмного моделювання компаратора виникає проблема вихідної напруги компаратора при однакових напругах на обох входах компаратора. При однакових напругах на входах, компаратор перебуває у стані нестійкої рівноваги.

4.2 Компаратор цифрових сигналів

Компаратором цифрових сигналів називається пристрій для порівняння двох однорідних величин, зокрема чисел A і B, та формування ознаки відношення між ними. Не маючи інформації про самі числа, за сигналами компаратора можна встановити: $A > B$, $A < B$ або $A = B$.

Цифрові компаратори призначені для порівняння чисел, поданих у двійковому коді [23].

Однорозрядний компаратор

Схема однорозрядного цифрового компаратора наведена на рисунку 4.2.

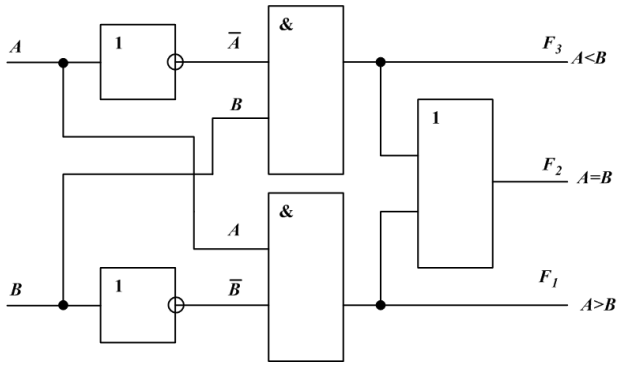


Рисунок 4.2 – Схема однорозрядного компаратора

Роботу найпростішого компаратора однорозрядних чисел можна описати нижченаведеною таблицею.

Таблиця 4.1 – таблиця істинності одно розрядного компаратора

A	B	F_1	F_2	F_3
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Такий пристрій має формувати три функції: F_1, F_2, F_3, що приймають одиничне значення за відповідних співвідношеннях між числами A і B.

Алгебраїчна форма запису цих функцій матиме вигляд, наведений нижче. F_2 є функцією логічної рівнозначності, оскільки її значення інверсні щодо функції логічної нерівнозначності, можливі дві форми запису F_2.

$$F_1 = \mathrm{A} \cdot \overline{\mathrm{B}};$$
$$F_2 = \overline{\mathrm{A}} \cdot \overline{\mathrm{B}} + \mathrm{A} \cdot \mathrm{B} = \overline{\mathrm{A} \cdot \overline{\mathrm{B}} + \overline{\mathrm{A}} \cdot \mathrm{B}};$$
$$F_3 = \overline{\mathrm{A}} \cdot \mathrm{B}.$$

Чотирирозрядний компаратор

Для порівняння сигналів використовується суматор, який виконує операцію віднімання (алгебраїчного додавання) вхідних чисел, і за цими результатами формуються сигнали, що показують співвідношення між ними. Для виконання операції віднімання одне з вхідних чисел подається у доповнювальному коді на відповідні входи суматора, тому до складу цифрового компаратора входить схема, яка формує доповнювальний код

одного з операндів. За результатами виконання операції $A - B$ можна зробити такі висновки:

- якщо результат дорівнює 0, то це є ознакою того, що значення A і B збігаються;
- якщо результат не дорівнює 0, а вихідне переповнення дорівнює 1, то $A < B$;
- якщо вихідне переповнення дорівнює 0, то це є ознакою, що $A > B$.

Сигнали, що показують співвідношення між вхідними числами, формуються за допомогою логічних схем.

Функцію дешифратора нуля тут виконує логічний елемент 4АБО. Тільки при рівності всіх розрядів суми $S_0 = S_1 = S_2 = S_3 = 0$ на виході 4АБО з'явиться 0, а після інвертора 1, яка підтверджує рівність А= В. У інших випадках на виході 4АБО є одиниця, яка стає дозволом для схеми збігу 2І.

Функціональна схема цифрового компаратора для чотирирозрядних чисел наведена на рисунку 4.3.

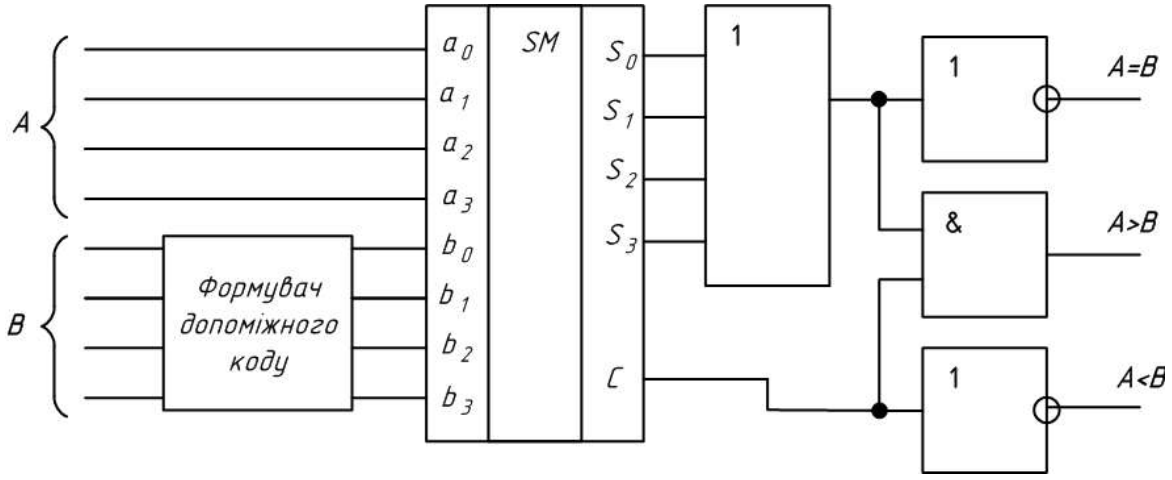


Рисунок 4.3 – Функціональна схема чотирирозрядного цифрового компаратора

Для нарощування розрядності вхідних даних цифрові компаратори дозволяється каскадувати.

4.3 Порядок виконання роботи

1. Розглянути основні теоретичні відомості лабораторної роботи.
2. Скласти схему відповідно до схеми на рисунку 4.4.
3. Змоделювати умови, за яких $_{in} <$ $_{ref}$, $_{in} >$ $_{ref}$ та $_{in} =$ $_{ref}$ й дослідити роботу схеми.

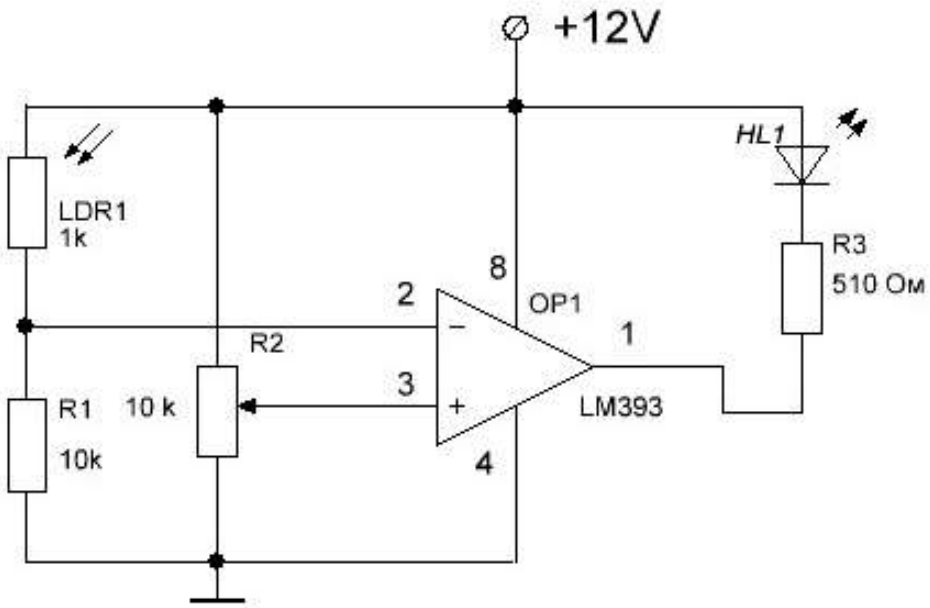


Рисунок 4.4 – Схема електрична принципова

4. Доповнити схему відповідно до схеми, наведеної на рисунку 4.5, та повторити п. 3.

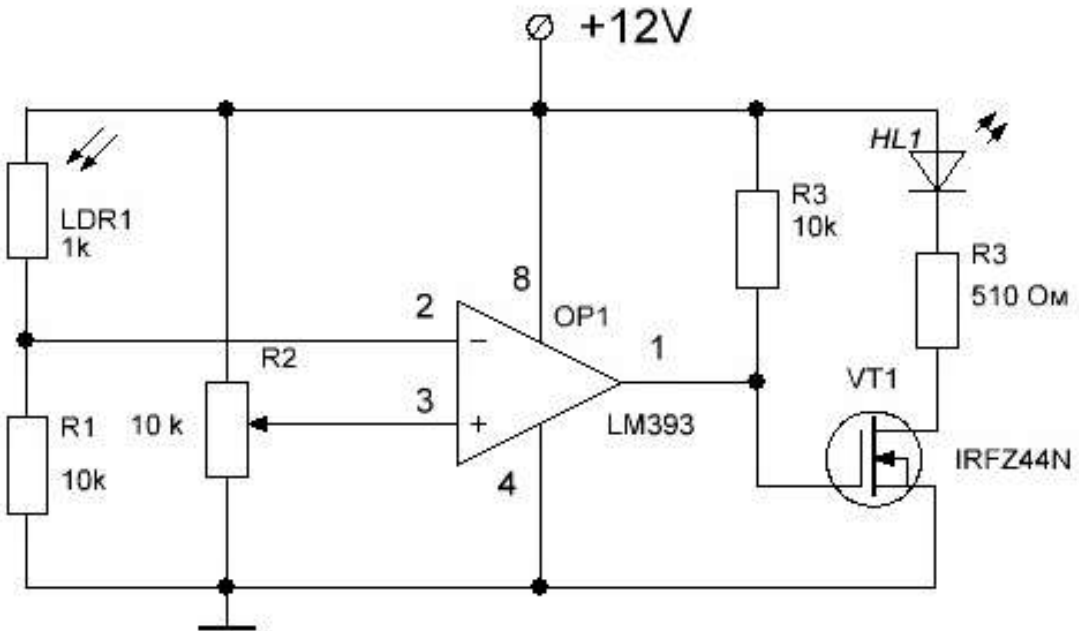


Рисунок 4.5 – Схема електрична принципова

5. Доповнити схему відповідно до схеми, наведеної на рисунку 4.6, та повторити п. 3.

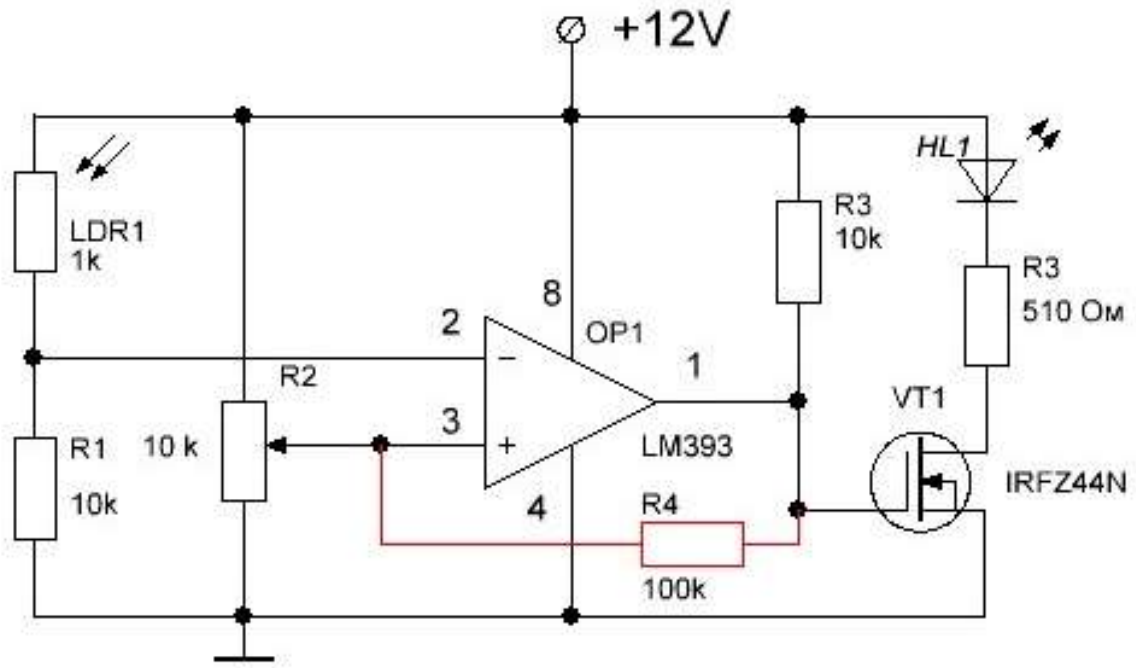


Рисунок 4.6 – Схема електрична принципова

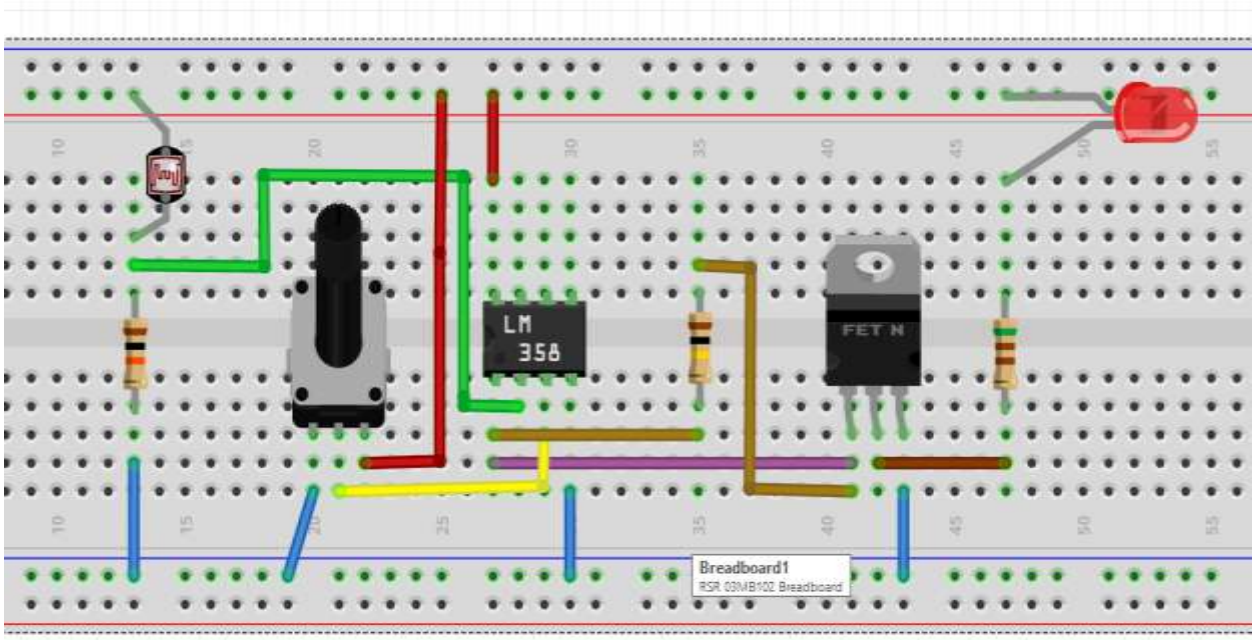


Рисунок 4.7 – Схема електрична комутаційна

Зовнішній вигляд монтажної плати, складеної відповідно до вимог п. 2 та п.4, наведено на рисунках 4.8–4.9.

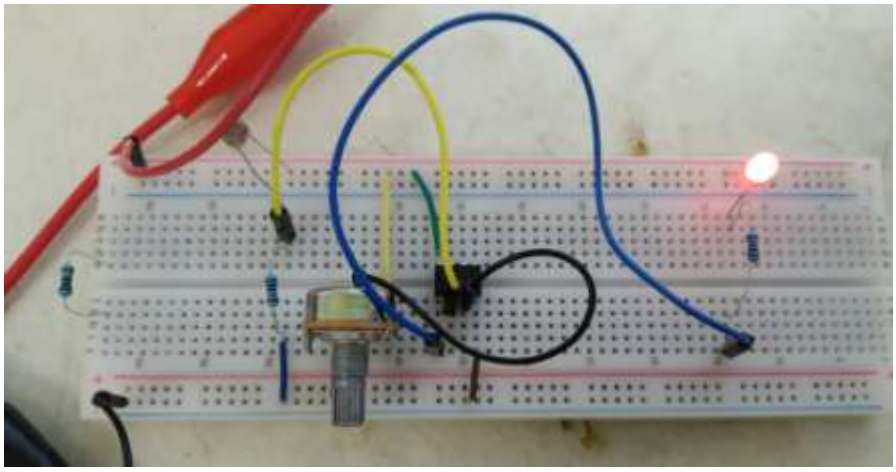


Рисунок 4.8 – Зовнішній вигляд монтажної плати

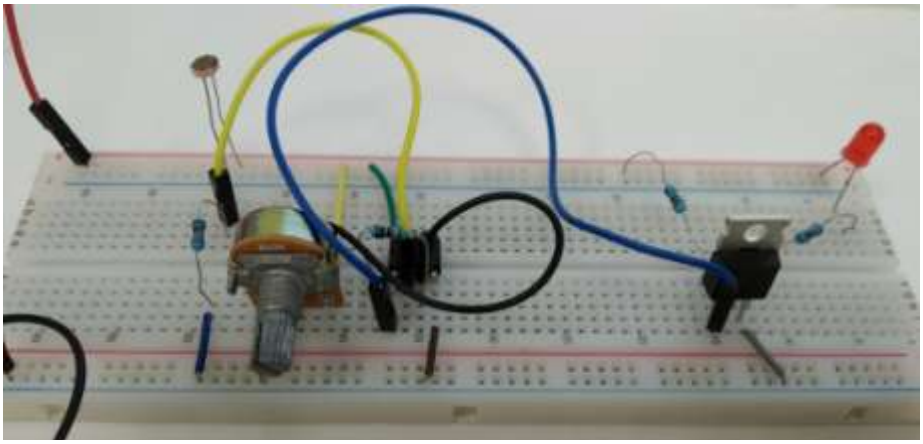


Рисунок 4.9 – Зовнішній вигляд монтажної плати

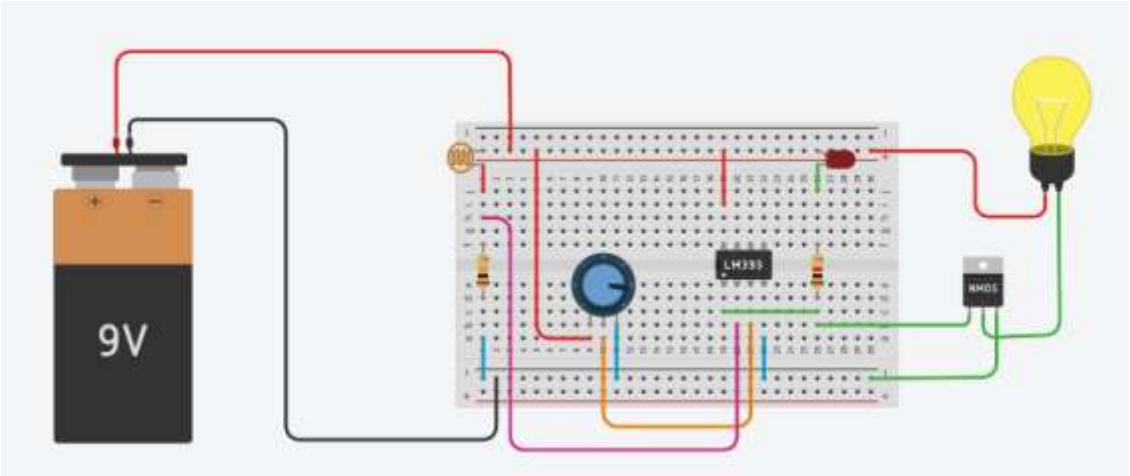


Рисунок 4.10 – Зовнішній вигляд монтажної плати у середовищі Tinkercad

У разі дистанційного навчання або навчання за індивідуальним графіком, дане виконання лабораторної роботи студент здійснює в он-лайн програмі для моделювання електронних схем – Tinkercad, як це показано на рисунках 4.7, 4.10.

6. Згідно з варіантом лабораторної роботи здійснити програмне моделювання компаратора й реалізувати задану схему та алгоритм її роботи (за необхідності використовувати емулятор Arduino у Tinkercad).

LM393. Опис [24]

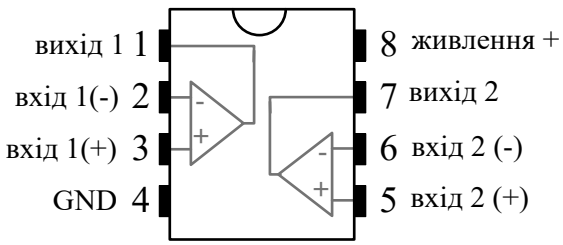
Мікросхема LM393 має у своєму корпусі два незалежні компаратори напруги. Компаратор LM393 може працювати як від однополярного джерела живлення в широкому діапазоні напруги, так і від двополярного джерела. У разі використання двополярного різниця між потенціалами має становити від 2 В до 36 В.

Струм споживання компаратора не залежить від напруги живлення. Потрібно звернути увагу, що цей компаратор має вихід із відкритим колектором.

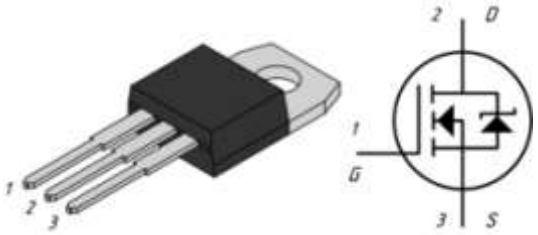
Технічні характеристики:

- одно- або двополярне живлення;
- широкий діапазон напруги живлення (від 2 В до 36 В);
- невеликий споживаний струм, що не залежить від напруги живлення 0,4 мА;
- низький вхідний струм зміщення: 25 нА;
- низький вхідний струм зміщення нуля: 25 нА;
- низька вхідна напруга зміщення нуля: 2 мВ;
- TTL-, DTL-, ECL-, MOS-, CMOS-сумісні виходи;
- Компаратор LM393 доступний у корпусі: DIP8, DFN8, MiniSO8, TSSOP8 та SO8.

Призначення виводів мікросхеми LM393.



Призначення виводів MOSFET транзистора IRFZ44N.



4.4 Зміст звіту

1. Звіт з лабораторної роботи має бути виконаний на аркушах формату А4.
2. Звіт має містити: назву лабораторної роботи, її мету і короткі теоретичні відомості.
3. В розділі «Результати виконання лабораторної роботи» студент має подати розрахунки, побудувати відповідні графіки, схему електричну принципову, схему електричну комутаційну, алгоритм реалізації, а також навести код програмної реалізації.
4. Написати висновки до даної лабораторної роботи.
5. Оформити звіт та подати його викладачу на захист.

Таблиця 4.1 – Варіанти завдань

Вар.	Порти МК	Третій стан компаратора «Х» приписати:
1	Вх. А2; А3; Вих. Р0	до «0» постійно
2	Вх. А3; А4; Вих. Р1	до «1» постійно
3	Вх. А0; А5; Вих. Р2	до «0» або «1» випадковим чином
4	Вх. А0; А5; Вих. Р3	до «1» постійно
5	Вх. А3; А0; Вих. Р4	враховувати попереднє значення
6	Вх. А1; А2; Вих. Р5	до «0» постійно
7	Вх. А5; А4; Вих. Р6	до «1» постійно
8	Вх. А0; А5; Вих. Р7	до «0» або «1» випадковим чином
9	Вх. А1; А4; Вих. Р8	до «1» постійно
10	Вх. А2; А3; Вих. Р9	до «0» постійно
11	Вх. А0; А3; Вих. Р10	до «1» постійно
12	Вх. А2; А0; Вих. Р11	до «0» або «1» випадковим чином
13	Вх. А3; А4; Вих. Р12	до «1» постійно
14	Вх. А4; А5; Вих. Р13	враховувати попереднє значення
15	Вх. А5; А0; Вих. А4	враховувати попереднє значення
16	Вх. А4; А5; Вих. А0	до «0» постійно
17	Вх. А1; А3; Вих. А2	до «1» постійно

Продовження таблиці 4.1

Вар.	Порти МК	Третій стан компаратора «Х» приписати:
18	Вх. А0; А4; Вих. А3	до «0» або «1» випадковим чином
19	Вх. А1; А2; Вих. Р12	до «1» постійно
20	Вх. А2; А3; Вих. Р0	враховувати попереднє значення
21	Вх. А3; А4; Вих. Р1	до «1» постійно
22	Вх. А0; А5; Вих. Р2	до «0» або «1» випадковим чином
23	Вх. А0; А5; Вих. Р3	до «1» постійно
24	Вх. А3; А0; Вих. Р4	до «1» постійно
25	Вх. А1; А2; Вих. Р5	до «0» або «1» випадковим чином
26	Вх. А5; А4; Вих. Р6	до «1» постійно
27	Вх. А0; А5; Вих. Р7	враховувати попереднє значення
28	Вх. А1; А4; Вих. Р8	враховувати попереднє значення

Контрольні запитання

1. Що таке подільник напруги?
2. Що таке петля гістерезису у компараторі?
3. Які види компараторів Ви знаєте?
4. Які види входів у компараторі Ви знаєте?
5. Що таке стан нестійкої рівноваги компаратора?
6. Які потенціали на виході компаратора LM393 існують?
7. Яку роль відіграє резистор R3 на схемі, зображеній на рисунку 4.5?
8. Як виконується порівняння двох двійкових чисел у цифровому компараторі?
9. Яку роль відіграє резистор R3 на схемі, зображеній на рисунку 4.4?
10. Поясніть призначення формувача доповнювального коду?
11. На підставі якої ознаки приймається рішення, що A = B?
12. На підставі яких ознак приймається рішення, що A > B?

Лабораторна робота № 5

СХЕМОТЕХНІЧНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СХЕМИ ВИМІРЮВАННЯ ТЕМПЕРАТУРИ ЗА ДОПОМОГОЮ АНАЛОГОВОГО ТА ЦИФРОВОГО ДАВАЧІВ

Мета роботи: Здійснити схемотехнічну та програмну реалізацію вимірювання температури навколишнього середовища за допомогою NTC терморезистора та цифрового давача DS18B20.

5 Основні теоретичні відомості

5.1 Вимірювання температури за допомогою терморезистора

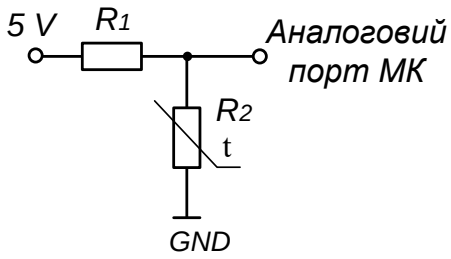
Терморезистор (або термістор) – це такий резистор, який змінює свій електричний опір залежно від температури [25].

Існує два види термісторів: PTC – з позитивним температурним коефіцієнтом, та NTC – з негативним. Позитивний коефіцієнт означає, що з підвищенням температури опір термістора зростає. NTC-термістор навпаки, зі збільшенням температури опір зменшується.

Також термістори відрізняються номінальним опором, який відповідає кімнатній температурі – 25 °C. Наприклад, популярними є термістори з номіналом 100 кОм та 10 кОм.

Рисунок 5.1 – Зовнішній вигляд терморезистора

Щоб виміряти опір термістора, під’єднаємо його як нижнє плече подільника напруги. Середню точку подільника під’єднують до аналогового входу мікроконтролера (рис. 5.2, 5.3) [26]. Подібний спосіб використовувався на лабораторній роботі дослідження компаратора з фоторезистором.



R1=R2=10 кОм

Рисунок 5.2 – Схема під’єднання терморезистора

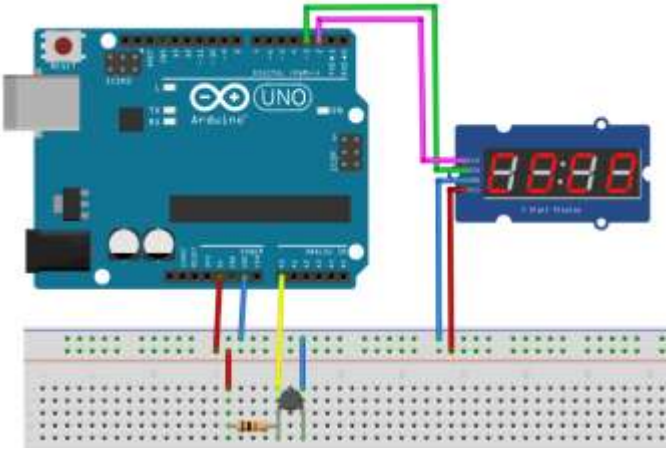


Рисунок 5.3 – Схема електрична комутаційна під’єднання терморезистора та дисплея TM 1637 до Arduino UNO

LED-індикатор на базі драйвера TM1637 [27]
Даний дисплей має такі характеристики:
напруга живлення – 3.3–5.5 В;
струм споживання – 0.2–80 мА;
кількість сегментів – 4;
інтерфейс – двопровідний послідовний;
регульована яскравість – 7 рівнів.

Рисунок 5.4 – Зовнішній вигляд TM1637

Для під'єднання дисплея до мікроконтролера використовують два контакти:

DIO – вхідні / вихідні дані;

CLK – синхронізація.

Під'єднувати дисплей до мікроконтролера Arduino будемо так: DIO – 2 пін, CLK – 3 пін.

Щоб обчислити значення температури, використовують рівняння Стейнхарта-Харта [28]

$$\frac{1}{T} = \mathrm{A} + \mathrm{B} \cdot \ln(\mathrm{R}) + \mathrm{C} \cdot (\ln(\mathrm{R}))^3$$

Рівняння має параметри A, B та C, які потрібно брати зі специфікації до давача. Оскільки нам не потрібна велика точність, можна скористатися модифікованим рівнянням (B-рівнянням)

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \cdot \ln\left(\frac{\mathrm{R}}{R_0}\right).$$

У цьому рівнянні невідомим залишається лише параметр *B*, який для NTC термістора дорівнює 3950. Інші параметри вже відомі:

- T_0 – кімнатна температура в кельвінах, на яку вказується номінал термістора; $T_0 = 25 + 273.15$;
- T – потрібна температура, в кельвінах;
- R – виміряний опір термістора в омах;
- R_0 – номінальний опір термістора в омах.

Приклад 5.1

```
#define B 3950                    // B-коефіцієнт
#define SERIAL_R 10000                    // опір послідовного
резистора,
  //10 кОм
#define THERMISTOR_R 10000              // номінальний опір
термістора,
                      //10 кОм
#define NOMINAL_T 25              // номінальна температура (при
  //якій TR = 10 кОм)
const byte tempPin = A0;
float steinhart;

void setup() {
  pinMode(tempPin, INPUT );
  Serial.begin(9600);
}

void loop() {
```

```
  int t = analogRead( tempPin );
  float tr = 1023.0 / t - 1;
  tr = SERIAL_R / tr;
  Serial.print("R=");
  Serial.print(tr);
  Serial.print(", t=");

  steinhart = tr / THERMISTOR_R; // (R/Ro)
  steinhart = log(steinhart); // ln(R/Ro)
  steinhart /= B; // 1/B * ln(R/Ro)
  steinhart += 1.0 / (NOMINAL_T + 273.15); // + (1/To)
  steinhart = 1.0 / steinhart; // Invert
  steinhart -= 273.15;
  Serial.println(steinhart);
  delay(100);
}
```

5.2 Вимірювання температури за допомогою цифрового давача DS18B20

Давач DS18B20 від компанії DALLAS високоточний цифровий давач температури, один з найпопулярніших цифрових давачів, з діапазоном вимірювання від -55 до +125 °C [29].

Особливість таких давачів – не лише широкий діапазон вимірювання, а й висока точність, абсолютна похибка менше 0.5 °C в діапазоні від -10 до +85 °C. Роздільна здатність АЦП може бути змінена програмно користувачем від 9 до 12 bit.

Характеристики:

- напруга живлення від 3 до 5.5 V;
- струм в очікуванні 750 nA;
- струм в роботі 1 мА;
- похибка в діапазоні від -10 до +85 ±0.5 °C;
- похибка в діапазоні від -55 до +125 ±2 °C;
- роздільна здатність (таблиця 5.1)

Таблиця 5.1 – Роздільна здатність

Розрядність bit	Температура °C
9	0.5
10	0.25
11	0.125
12	0.0625

Під’єднання

Для під’єднання використовується шина 1-Wire. Перевагою цієї шини є можливість під’єднати теоретично необмежену кількість пристроїв на одну лінію. Сам давач має унікальний 64-розрядний ідентифікатор, що в теорії дає можливість під’єднати на одну лінію 2^{64} = 18 446 744 073 709 551 616 давачів DS18B20.

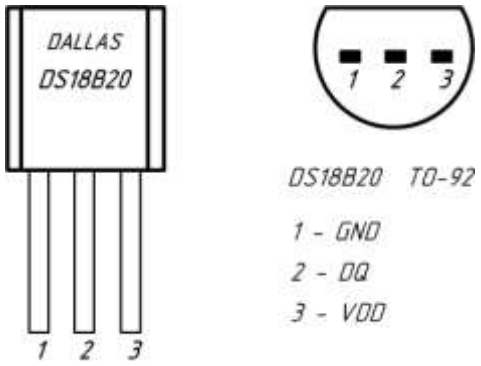


Рисунок 5.5 – Схема виводів давача

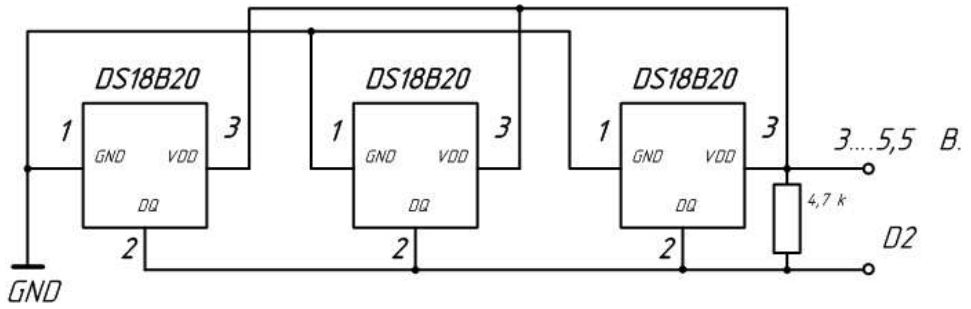


Рисунок 5.6 – Схема під’єднання давачів DS18B20 до тридротової лінії

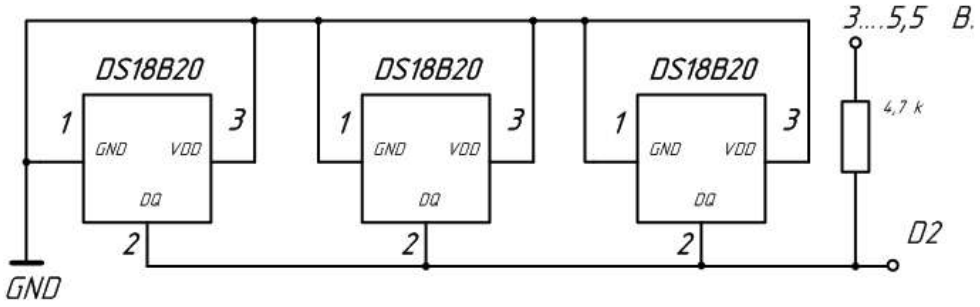


Рисунок 5.7 – Схема під’єднання давачів DS18B20до дводротової лінії (паразитне живлення)

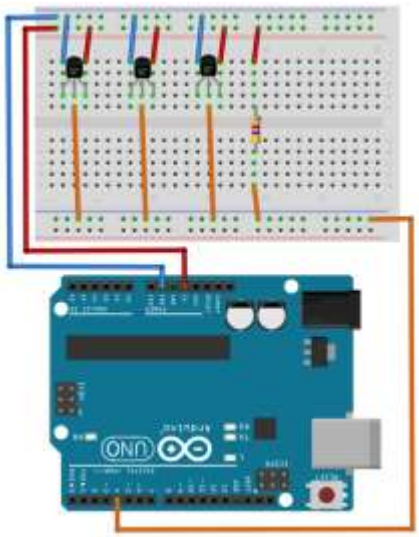


Рисунок 5.8 – Схема електрична комутаційна під’єднання
трьох давачів DS18B20

Робота з DS18B20 без адресації

У цьому режимі на один вивід МК під’єднується один давач, для роботи з ним не потрібно попередньо читати адресу і записати його в програму. Можна під’єднати кілька давачів, кожному вказати свій вивід.

```
MicroDS18B20 < № піна 1 > sensor 1;
MicroDS18B20 < № піна 2 > sensor 2;
// ... і так далі
```

Робота з DS18B20 з адресацією

У цьому режимі можна під’єднати скільки завгодно давачів на один вивід МК, але для роботи з ними потрібно занести в програму унікальні адреси давачів. У момент читання адреси до піну має бути під’єднаний лише один давач! Приклад address_read. Для подальшої роботи адреси зберігаються в масивах на стороні програми і передаються давачам при ініціалізації, вивід вказується той самий [30]:

```
   uint8_t  addr1[]  =  {0x28,  0xFF,  0xCD,  0x59,  0x51,
0x17, 0x4, 0xFE};
   uint8_t  addr2[]  =  {0x28,  0xFF,  0x36,  0x94,  0x65,
0x15, 0x2, 0x80};

   MicroDS18B20< пін, addr1> sensor1;
   MicroDS18B20< пін, addr2> sensor2;
   // ... і так далі
```

Також можна змінити адресу під час роботи програми.

Приклад 5.2

```
#include <microDS18B20.h>
MicroDS18B20 <2> sensor;     // Підєднуємо давач на пін D2
uint8_t address[8];          // Створюємо масив для адреси

void setup() {
  Serial.begin(9600);
}

void loop() {
                      // зчитуємо адресу давача у вказаний масив
  if (sensor.readAddress(address)) {     //  Якщо  успішно,
виводимо
    Serial.print('{');
    for (uint8_t i = 0; i < 8; i++) {
      Serial.print("0x");
      Serial.print(address[i], HEX);   // Виводимо адресу
      if (i < 7) Serial.print(", ");
    }
    Serial.println('}');

  } else Serial.println("Not connected");
  delay(1000);
}
```

Зчитування температури

Читання температури ділиться на два етапи – запит та отримання даних. Запит здійснюється функцією: `requestTemp()`.

Після отримання запиту давач починає вимірювання температури, що триває від 90 до 750 мс залежно від налаштованої точності (за замовчуванням максимальна точність, перетворення триває 750 мс) [31]. Якщо прочитати температуру до закінчення перетворення, – давач поверне результат попереднього вимірювання, тому приклади використовують затримку або опитування за таймером на 1 секунду. Отримати значення температури можна за допомогою `getTemp()  [float]` або `getTempInt()[int]`.

Якщо отримані дані пошкоджені або давач відсутній на лінії, – функція поверне попереднє прочитане значення температури.

При повторних викликах `getTemp()` не запитує з давача нову температуру, натомість вона просто повертає попередній результат доти, доки не буде зроблено новий запит `requestTemp()`.

У версії бібліотеки 3.5 з'явилася можливість окремо запитати температуру та визначити коректність отриманих даних, щоб тільки після цього їх прочитати та застосувати у програмі – функція `readTemp ()`.

Також це дозволяє визначити стан під'єднання і чи все гаразд із давачем. Для читання температури рекомендується використовувати конструкцію виду

```
if ( sensor. readTemp ()) value = sensor. getTemp () ;
 else          //відпрацювання помилки
```

де readTemp() запитує дані з давача та повертає true, якщо вони прочитані коректно. Після цього можна забрати поточну температуру з getTemp (), яка вже не запитує температуру з давача, а віддає прочитаний в readTemp () результат.

Приклад 5.3

```
#include <microDS18B20.h>
#define DS_PIN 4                          // пін для давача
                                  // Унікальні адреси давачів
uint8_t s1_addr[] = {0x28, 0x52, 0x1E, 0x7, 0xD6, 0x1, 0x3C,
0x17};
uint8_t s2_addr[] = {0x28, 0xA7, 0x4E, 0x4B, 0x3, 0x0, 0x0,
0x1C};

MicroDS18B20<DS_PIN, s1_addr> sensor1;  // Створюємо давач з
   //адресацією
MicroDS18B20<DS_PIN, s2_addr> sensor2;  // Створюємо давач з
                                        //адресацією
void setup() {
  Serial.begin(9600);
}
void loop() {
  sensor1.requestTemp();       // робимо запит на перетворення
                //температури
  sensor2.requestTemp();       // робимо запит на перетворення
                               // температури

  delay(1000);                 // очікуємо результат

  Serial.print("Sensor_1 t= ");
  if (sensor1.readTemp()) Serial.println(sensor1.getTemp());
  else Serial.println("error");

  Serial.print("Sensor_2 t= ");
  if (sensor2.readTemp()) Serial.println(sensor2.getTemp());
  else Serial.println("error");
}
```

5.3 Порядок виконання роботи

1. Розглянути основні теоретичні відомості вимірювання температури за допомогою терморезистора та цифрового давача.
2. Скласти схему відповідно до схем на рисунках 5.2–5.3.

3. Реалізувати програмний код вимірювання температури за допомогою терморіезистора з урахуванням прикладу реалізації (див. приклад 5.1).

4. Здійснити виведення значення температури на LED-індикатор TM1637, при цьому на два останніх молодші розряди здійснити виведення символів «°», «C» hex, значення яких 0×63 та 0×39 відповідно.

5. Скласти схему відповідно до схеми на рисунку 5.8.

6. Завантажити програмний код (див. приклад 5.2) для зчитування адреси давачів та отримати їх значення (результати заносимо у звіт лабораторної роботи).

7. Реалізувати програмний код вимірювання температури за допомогою цифрових давачів DS18B20 згідно з Вашим варіантом (табл. 5.2) та з урахуванням прикладу реалізації (див. приклад 5.3).

8. Вивести значення температури на LED-індикатор TM1637.

Таблиця 5.2 – Варіанти завдань

Варіант	Порт	Кількість давачів	Варіант	Порт	Кількість давачів
1.	P2	1	16.	P6	4
2.	P3	2	17.	P7	3
3.	P4	3	18.	P8	2
4.	P5	4	19.	P8	1
6.	P7	3	21.	P6	2
7.	P8	2	22.	P7	3
8.	P8	1	23.	P8	4
9.	P10	2	24.	P8	4
10.	P13	4	25.	P10	2
11.	P12	1	26.	P13	3
12.	P11	3	27.	P10	1
13.	P10	1	28.	P13	2
14.	P9	2	29.	P3	3
15.	P8	3	30.	P4	1

5.4 Зміст звіту

1. Звіт з лабораторної роботи має бути виконаний на аркушах формату А4.

2. Звіт має містити: назву лабораторної роботи, її мету і короткі теоретичні відомості.

3. В розділі «Результати виконання лабораторної роботи» студент має навести необхідні розрахунки, побудувати відповідні графіки, схему електричну принципову, схему електричну комутаційну, алгоритм реалізації, а також навести код програмної реалізації.

4. Написати висновки до даної лабораторної роботи.

5. Оформити звіт та подати його викладачу на захист.

Контрольні запитання

1. Що таке терморезистор?
2. Які види терморезисторів Ви знаєте?
3. Наведіть схему під’єднання терморезистора до аналогового порту мікроконтролера.
4. Наведіть модифіковане рівняння (*B*-рівняння) Стейнхарта-Харта.
5. Як розуміти поняття: номінальний опір термістора?
6. Що таке *B*-коефіцієнт, яке він може приймати значення?
7. Яка, в теорії, можлива кількість давачів DS18B20, яку можна під’єднати до однієї лінії даних?
8. Що таке «паразитне живлення» давача DS18B20?
9. Наведіть схему під’єднання давача DS18B20 до дротової лінії?
10. Наведіть схему під’єднання давача DS18B20 до дротової лінії.
11. Поясніть різницю між вимірюванням температури давачем DS18B20 з адресацією та без адресації.

ГЛОСАРІЙ

Ардуіно – апаратна обчислювальна платформа для аматорського конструювання, основними компонентами якої є плата мікроконтролера з елементами введення/виведення та середовище розробки Processing/Wiring мовою програмування, що є спрощеною підмножиною C/C++.

Бітові оператори – оператори, які виконують побітові операції над операндами на двійковому рівні, зокрема & (AND), | (OR), ^ (XOR), << (зсув вліво), >> (зсув вправо) та ~ (побітове NOT).

Виконуваний файл – файл, що містить інструкції машинного коду, згенеровані компілятором і компонувальником, які можуть виконуватися безпосередньо процесором комп'ютера.

Давач – пристрій, який виявляє і реагує на фізичні подразники або зміни в навколишньому середовищі, забезпечуючи зворотний зв'язок з системою управління роботом.

Директива препроцесора – інструкції препроцесору C, які змінюють вихідний код перед компіляцією, такі як #define, #include та #ifdef.

Драйвер – комп'ютерна програма, за допомогою якої операційна система отримує доступ до певного приладу чи частини апаратного забезпечення.

Змінна – іменована комірка в пам'яті, яка використовується для зберігання значень даних у програмі C. Змінні мають бути оголошені з вказанням типу даних.

Інтегроване середовище розробки (IDE) – комплексне програмне рішення для розробки програмного забезпечення. Зазвичай складається з редактора початкового коду, інструментів для автоматизації складання та налагодження програм.

Інтерфейс апаратний – сукупність алгоритмів обміну і технічних засобів, що забезпечують обмін інформацією між пристроями. Апаратний інтерфейс є системою шин, роз'ємів, узгоджувальних пристроїв, алгоритмів і протоколів, що забезпечують зв'язок всіх частин мікропроцесорної системи між собою. Від характеристик інтерфейсу залежить швидкодія і надійність системи. У розгорнутих мікропроцесорних системах для розвантаження процесора апаратний інтерфейс забезпечується контролерами.

Інтерфейс користувача – Різновид інтерфейсу, в якому одна сторона – це людина (користувач), інша – машина / пристрій. Це також сукупність засобів і методів, за допомогою яких користувач взаємодіє з різними, найчастіше складними, машинами, пристроями, апаратурою і програмним забезпеченням. Часто термін застосовується щодо комп'ютерних програм, однак під ним може матися на увазі набір засобів, методів і правил взаємодії будь-якої системи,

керованої людиною. Інтерфейс вважається двонаправленим (інтерактивним), якщо пристрій, отримавши команди від користувача і виконавши їх, видає інформацію користувачу наявними у нього засобами – візуальними, звуковими та ін. Інтерфейс – це сукупність елементів, які, зокрема, також можуть складатися з елементів.

Інтерфейс програмний – набір готових класів, процедур, функцій, структур і констант, що надаються інформаційною системою (бібліотекою, сервісом) для використання у зовнішніх програмних продуктах. Використовується програмістами для написання власних програмних засобів.

Компілятор – програма, яка перекладає вихідний код, написаний мовою С, у машинний код або проміжний код, який може бути виконаний комп'ютером.

Масив – набір елементів одного типу даних, що зберігаються в суміжних комірках пам'яті, доступ до яких здійснюється за допомогою індексу.

Мікроконтролер (або однокристальний мікрокомп'ютер) – виконаний у вигляді мікросхеми спеціалізований комп'ютер, що містить мікропроцесор, пам'ять (оперативну та постійну) для збереження виконуваного коду програм і даних, порти введення / виведення і блоки зі спеціальними функціями.

Мова програмування С – процедурна мова програмування загального призначення, розроблена Деннісом Рітчі на початку 1970-х років у Bell Labs. С широко використовується для розробки системного та прикладного програмного забезпечення.

Оператори – символи, які є представниками операцій, що виконуються над операндами. Прикладами у С є арифметичні оператори (+, -, *, /), реляційні оператори (==, !=, <, >, <=, >=) та логічні оператори (&&, ||, !).

Порт мікроконтролера – Це пристрої введення / виведення, що дозволяють мікроконтролеру передавати або приймати дані. Стандартний порт мікроконтролера AVR має вісім розрядів даних, які можуть передаватися або прийматися паралельно. Кожному розряду (або біту) відповідає вивід (ніжка) мікроконтролера.

Послідовний порт (COM порт) – двонаправлений послідовний інтерфейс, призначений для обміну байтовою інформацією.

Програма – послідовність інструкцій, призначених для виконання пристроєм управління обчислювальної машини. Програма – один з компонентів програмного забезпечення. Залежно від контексту цей термін може стосуватися також і вихідних текстів програми.

Програматор – апаратно-програмний пристрій, призначений для записування / зчитування інформації в постійний запам'ятовувальний пристрій (одноразово записується, флеш-пам'ять, внутрішню пам'ять мікроконтролерів).

Синтаксис мови С – набір правил, що визначає комбінації символів, які вважаються правильно структурованими програмами або виразами у С.

Скетч – це програма, написана для платформи Arduino і має певну структуру. Скетч обов'язково містить 2 функції: функцію Setup і функцію Loop.

Типи даних – категорії значень, які визначають можливі операції над цими значеннями та спосіб їх зберігання в пам'яті. Прикладами у С є int, float, double та char.

Функція – самодостатній блок коду, який виконує певну задачу. Функції забезпечують модульність та багаторазове використання у програмах С.

Цикл – різновид керівної конструкції у високорівневих мовах програмування, призначений для організації багаторазового виконання набору інструкцій. Також циклом може називатися будь-яка багатократно виконувана послідовність команд, організована будь-яким чином.

ШІМ (PWM) – Широтно-імпульсна модуляція. Сигнал змінної шпаруватості, але постійної частоти.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Wiring [Електронний ресурс]. Режим доступу до ресурсу: http://wiring.org.co/.
2. Arduino IDE 1.8.13 [Електронний ресурс]. Режим доступу до ресурсу: https://www.arduino.cc/en/software.
3. Матеріали по програмуванню Arduino [Електронний ресурс]. Режим доступу до ресурсу: https://www.arduino.cc/.
4. Белзецький. Р. С., Яровий А. А., Іванчук Я. В. Технології захисту інформації : лабораторний практикум [Електронний ресурс]. Вінниця : ВНТУ, 2022. 118 с.
5. Arduino Uno [Електронний ресурс]. Режим доступу до ресурсу: https://arduino.ua/prod32-arduino-uno-rev3-a000066
6. Що таке тактова кнопка і як вона працює? Все, що потрібно знати! [Електронний ресурс]. https://la-crevet.com.ua/shho-take-taktova-knopka-i-yak-vona-pracyuie-vse-shho-potribno-znati/
7. Підключення тактових кнопок до Arduino [Електронний ресурс]. Режим доступу до ресурсу: https://qazf.com.ua/buttons-arduino/
8. Цирульник С. М., Лисенко Г. Л. Проєктування мікропроцесорних систем : навчальний посібник. Вінниця : ВНТУ, 2010. 201 с.
9. How to run code only one time when conditions for code to run are constanly valid. [Електронний ресурс]. Режим доступу до ресурсу: https://forum.arduino.cc/t/how-to-run-code-only-one-time-when-conditions-for-code-to-run-are-constanly-valid/1079421
10. Як позбутися від брязкоту контактів при підключенні до кнопки Arduino. [Електронний ресурс]. Режим доступу до ресурсу: https://poradumo.com.ua/183474-iak-pozbytisia-vid-briazkoty-kontaktiv-pri-pidkluchenni-do-knopki-arduino/
11. З чого складається світлодіод? [Електронний ресурс]. Режим доступу до ресурсу: https://bitkit.com.ua/svetodiod
12. Робота з цифровими сигналами. [Електронний ресурс]. Режим доступу до ресурсу: https://gb.mistoboyarka.gov.ua/files/project/1632/documents/-15120646387077_1512063065297762.pdf
13. Широтно-імпульсна модуляція. [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A8%D0%B8%D1%80%D0%BE-%D1%82%D0%BD%D0%BE%D1%96%D0%BC%D0%BF%D1%83%D0%BB%D1%8C%D1%81%D0%BD%D0%B0_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F

14. Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Основи робототехніки» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійною програмою «Інтернет речей» спеціальності 121 «Інженерія програмного забезпечення» денної форми навчання [Електронне видання] / Реут Д. Т. Рівне : НУВГП, 2022. 50 с.

15. Evans B. Arduino programming notebook. [Електронний ресурс] Режим доступу до ресурсу: http://engineering.nyu.edu/gk12/ampscbri/pdf/ArduinoBooks/Arduino%20Programming%20Notebook.pdf.

16. Khaled Magdy. Arduino RGB LED Tutorial . [Електронний ресурс]. Режим доступу до ресурсу: https://www.circuitgeeks.com/arduino-rgb-led-tutorial/

17. 5611AS - 0.56-inch Red 1-Digit CC LED 7-Segment Display URL: http://www.xlitx.com/datasheet/5611AS.pdf

18. Alexander Fakoo. Siekoo-Alphabet. [Електронний ресурс]. Режим доступу до ресурсу: https://fakoo.de/en/siekoo/siekoo-alphabet.html

19. Подільник напруги. [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%96%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA_%D0%BD%D0%B0%D0%BF%D1%80%D1%83%D0%B3%D0%B8

20. Multiple buttons on one pin with an Arduino. [Електронний ресурс]. Режим доступу до ресурсу: https://www.g7smy.co.uk/2015/09/multiple-buttons-on-one-pin-with-an-arduino/

21. Електроніка та мікросхемо техніка : підручник / Воробйова О. М., Панфілов І. П., Савицька М. П., Флейта Ю. В. Одеса : ОНАЗ ім. О. С. Попова, 2015. 298 с.

22. Компаратор. [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BE%D1%80

23. Антонов О. С., Хіхловська І. В. Обчислювальна техніка та мікропроцесори : підр. Вид. 2-ге. Одеса : Вид. центр ОНАЗ ім. О. С. Попова, 2009. 440 с.

24. LM393 Datasheet – Low Power Dual Comparators – Motorola. [Електронний ресурс]. Режим доступу до ресурсу: https://www.datasheetcafe.com/lm393-datasheet-motorola/

25. Терморезистор. [Електронний ресурс]. Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D1%80%D0%BC%D0%BE%D1%80%D0%B5%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80

26. Термістор і Arduino. [Електронний ресурс]. Режим доступу до ресурсу: https://diy-arduino.com/moduli-na-arduino/termistor-i-arduino/

27. LED-індикатор семисегментний 4-розрядний з драйвером TM1637. [Електронний ресурс]. Режим доступу до ресурсу: https://arduino.ua/prod1629-posledovatelnii-4-razryadnii-semisegmentnii-led-indikator-s-i2c-draiverom-tm1637

28. Matus, Michael (October 2011). Temperature Measurement in Dimensional Metrology – Why the Steinhart–Hart Equation works so well (PDF). MacroScale 2011. Wabern, Switzerland.

29. DS18B20 Datasheet (PDF) – Dallas Semiconductor. [Електронний ресурс]. Режим доступу до ресурсу: https://pdf1.alldatasheet.com/datasheet-pdf/view/58557/DALLAS/DS18B20.html

30. DS18b20 addressing at runtime. [Електронний ресурс]. Режим доступу до ресурсу: https://forum.arduino.cc/t/ds18b20-addressing-at-runtime/471865

31. Денисюк В. О., Цирульник С. М. Мікропроцесорні системи управління : навч. посіб. Вінниця : ТВОРИ, 2021. 204 с.

Навчальне електронне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Руслан Станіславович Белзецький
Ярослав Володимирович Іванчук

КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ОСНОВИ РОБОТОТЕХНІКИ

Лабораторний практикум

Рукопис оформив Р. Белзецький

Редактор В. Дружиніна

Оригінал-макет виготовила Т. Старічек

Підписано до видання 03.07.2024 р.
Гарнітура Times New Roman.
Зам. № Р2024-126.
Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
press.vntu.edu.ua;
E-mail: irvc.ed.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.