МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

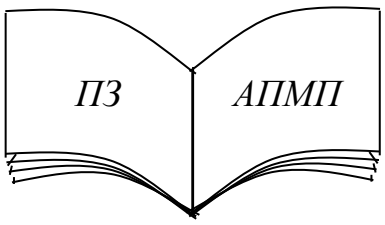
МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних робіт з навчальної дисципліни

«Архітектура та програмування мікропроцесорів»

для студентів денної та заочної форми навчання

за спеціальністю «Комп’ютерна інженерія»



Харків 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних робіт з навчальної дисципліни
«Архітектура та програмування мікропроцесорів»

для студентів денної та заочної форми навчання
за спеціальністю «Комп’ютерна інженерія»

Затверджено
редакційно-видавничою
радою НТУ «ХПІ»,
протокол № 1 від 15.02.2024

Харків
НТУ «ХПІ»
2024

Методичні вказівки до виконання практичних робіт з навчальної дисципліни «Архітектура та програмування мікропроцесорів» для студентів денної та заочної форми навчання за спеціальністю «Комп’ютерна інженерія» / А. О. Подорожняк, Г. В. Гейко, С. Г. Межерицький, Н. Ю. Любченко. – Харків : НТУ «ХПІ», 2024. – 94 с.

Укладачі: А. О. Подорожняк, Г. В. Гейко, С. Г. Межерицький, Н. Ю. Любченко

Рецензент: проф. В. В. Усик

Кафедра комп’ютерної інженерії та програмування

ВСТУП

Методичні вказівки містять методику виконання практичних занять, метою яких є отримання студентами спеціальних навичок з базових основ архітектури, програмування та режимів функціонування мікропроцесорів і мікропроцесорних засобів та ознайомлення з контролером прямого доступу до пам’яті, контролером переривання, системним таймером та годинником реального часу; вивчення основних режимів роботи мікропроцесора, включаючи різноманітні варіанти вирішення практичних задач з використанням мікропроцесора та багатозадачним режимом.

При виконанні практичних робіт слід керуватися наступними положеннями:

1) практичні заняття проводяться фронтально у всій групі, об’єм завдань визначає викладач;

2) до кожної практичної роботи необхідна самостійна підготовка, що включає вивчення теоретичного матеріалу та виконання необхідних проєктних робіт, теоретичний аналіз розроблених схем, побудова часових діаграм;

3) в ході виконання практичної роботи студенти повинні зібрати у програмі моделювання і на універсальній монтажній платі схему та виконати її дослідження;

4) по кожній практичній роботі складається звіт;

5) всі розроблені схеми та результати досліджень необхідно представити викладачу для демонстрації працездатності схеми або пристрою.

При здачі звіту про виконання практичної роботи, студенти повинні бути готові відповісти на контрольні питання, які знаходяться в методичних вказівках, і на додаткові питання за матеріалом, що вивчається.

Практична робота 1

ДОСЛІДЖЕННЯ ОРГАНІЗАЦІЇ ПРОЦЕСУ БЛИМАННЯ СВІТЛОДІОДІВ НА МІКРОПРОЦЕСОРІ ATMEGA328

Мета роботи: Дослідження організації процесу блимання світлодіодів на мікропроцесорі ATMega328 на платформі Arduino. Одержання практичних навичок видачі заданого сигналу на виводи портів мікропроцесора ATMega328 на платі Arduino UNO R3.

Теми для попереднього проороблення:

– теоретичні відомості про мікропроцесор ATMega328;

– теоретичні відомості про платформу Arduino;

– теоретичні відомості про середовище Arduino IDE;

– теоретичні відомості про проєктування мікропроцесорних пристроїв у середовищі PROTEUS VSM.

1. Ознайомлення із основними принципами підключення світлодіодів до мікропроцесорів та кольоровим маркуванням резисторів

Світлодіод, або світловипромінюючий діод (LED – light emitting diode) – напівпровідниковий прилад, що випромінює некогерентне світло при пропусканні через нього електричного струму. Робота заснована на фізичному явищі виникнення світлового випромінювання при проходженні електричного струму через *p-n* перехід. Колір світіння (довжина хвилі максимуму спектра випромінювання) визначається типом використовуваних напівпровідникових матеріалів, що утворюють *p-n* перехід (рис. 1.1).

Рисунок 1.1 – Світловипромінюючі діоди

Переваги світлодіодів:

1) світлодіоди не мають ніяких скляних колб і ниток розжарювання, що забезпечує високу механічну міцність і надійність (ударна і вібраційна стійкість);

2) відсутність розігріву і високої напруги гарантує високий рівень електро- та пожежобезпеки;

3) безінерційність робить світлодіоди незамінними, коли потрібна висока швидкодія;

4) мініатюрність;

5) довгий термін служби;

6) високий коефіцієнт корисної дії;

7) відносно низькі напруги живлення та споживані струми, низьке енергоспоживання;

8) велика кількість різних кольорів світіння та спрямованість випромінювання;

9) регульована інтенсивність.

У світлодіодів є декілька основних параметрів:

– тип корпусу;

– типовий (робочий) струм та типова (робоча) напруга;

– колір світіння (довжина хвилі, нм);

– кут розсіювання.

Загалом, під типом корпусу розуміють діаметр та колір колби (лінзи). Світлодіод необхідно живити струмом, який називається типовим. При цьому на світлодіоді падає певна напруга. Колір випромінювання визначається як використовуваними напівпровідниковими матеріалами, так і легуючими домішками.

Найважливішими елементами, використовуваними в світлодіодах, є алюміній (Al), галій (Ga), індій (In), фосфор (P), що викликають світіння в діапазоні від червоного до жовтого кольору. Індій (In), галій (Ga), азот (N) використовують для набуття блакитного і зеленого світіння. Крім того, якщо до кристала, що викликає блакитне (синє) світіння, додати люмінофор, то отримаємо білий колір. Кут випромінювання також визначається виробничими характеристиками матеріалів, а також колбою (лінзою) світлодіода.

Схема підключення та розрахунок необхідних параметрів

Оскільки світлодіод є напівпровідниковим приладом, то при включенні в електричне коло необхідно дотримувати полярність. Світлодіод має два виводи, один з яких катод (“мінус”), а інший – анод (“плюс”). Щоб правильно підключити світлодіод у самому простішому випадку, необхідно підключити його через струмообмежувальний резистор (рис. 1.2).

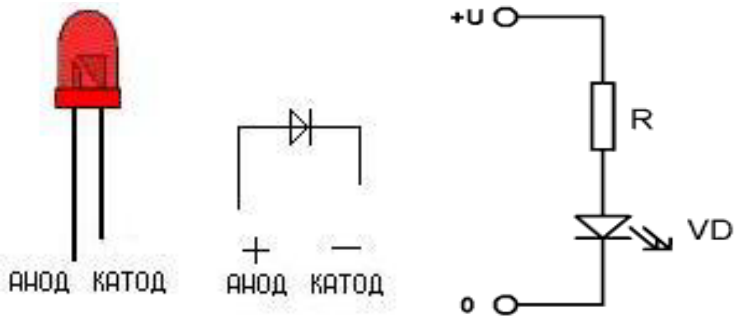


Рисунок 1.2 – Схема підключення світлодіода

У нашому випадку розрахунок опору струмообмежувального резистора виконується за формулами:

$$R = U_{\text{гасяча}} / I_{\text{світлодіода}}$$

$$U_{\text{гасяча}} = U_{\text{живлення}} - U_{\text{світлодіода}}$$

У випадку з використанням мікропроцесора ATMega328 з напругою живлення 5 В використовуємо R = 200 Ом.

Кольорове маркування резисторів

На резисторі є три смуги зліва та одна трохи зміщена праворуч (рис. 1.3). Кожна кольорова смужка відповідає певній цифрі (табл. 1.1).

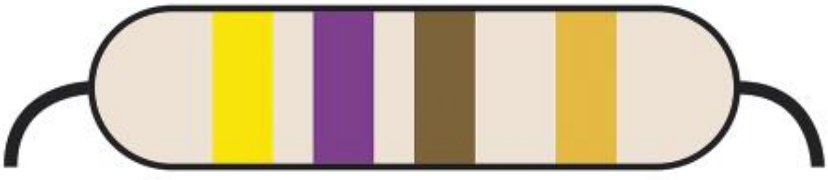


Рисунок 1.3 – Зображення резистора 470 Ом з кольоровим маркуванням

Таблиця 1.1 – Кольорове цифрове кодування

Колір	Цифра	Колір	Цифра
Чорний	0	Зелений	5
Коричневий	1	Синій	6
Червоний	2	Фіолетовий	7
Помаранчевий	3	Сірий	8
Жовтий	4	Білий	9

Якщо подивитись на перші дві смуги (рис. 1.3) – це базове значення, отже, отримуємо число “47”. Третя смуга – це множник, її чисельне значення є числом нулів, які треба додати до числа 47. Оскільки на рис. 1.3 третя смуга коричнева, треба додати один “0”, тобто значення резистора буде 470 Ом.

Четверта смуга на рис. 1.3 позначає точність опору резистора, так званий допуск, який вказує невідповідність номіналу у відсотках. Це важливо при складанні схем, в яких опір має бути відкалібрований. Точність номіналу резистора позначається у відсотках і для четвертої смуги існують спеціальні колірні коди (табл. 1.2).

Оскільки на рис. 1.3 четверта смуга золотого кольору, то точність резистора буде ± 5 %, тобто істинне значення опору резистора буде від 447 Ом до 494 Ом.

Таблиця 1.2 – Кольорове кодування точності номіналу резисторів

Колір	Точність
Срібний	± 10 %
Золотий	± 5 %
Червоний	± 2 %
Коричневий	± 1 %

2. Ознайомлення із процесом блимання вбудованим світлодіодом на платі Arduino UNO R3

Arduino – це електронний конструктор і зручна платформа швидкої розробки електронних пристроїв для новачків і професіоналів. Переваги платформи: зручна

у використанні, простота мови програмування, відкрита архітектура і програмний код. Пристрій програмується через USB без використання програматорів.

Пристрої на базі Arduino можуть отримувати інформацію про навколишнє середовище за допомогою різних датчиків, а також можуть управляти різними виконавчими пристроями.

Існує декілька версій платформ Arduino. Ми будемо розглядати платформу Arduino UNO R3 (рис. 1.4), яка базується на мікроконтролері Atmel ATmega328. Мікроконтролер на платі програмується за допомогою мови Arduino C++ у середовищі розробки Arduino (заснована на середовищі Processing). Проєкти пристроїв, засновані на Arduino, можуть працювати самостійно, або взаємодіяти з програмним забезпеченням на комп'ютері (наприклад, з Flash, Processing, MaxMSP).

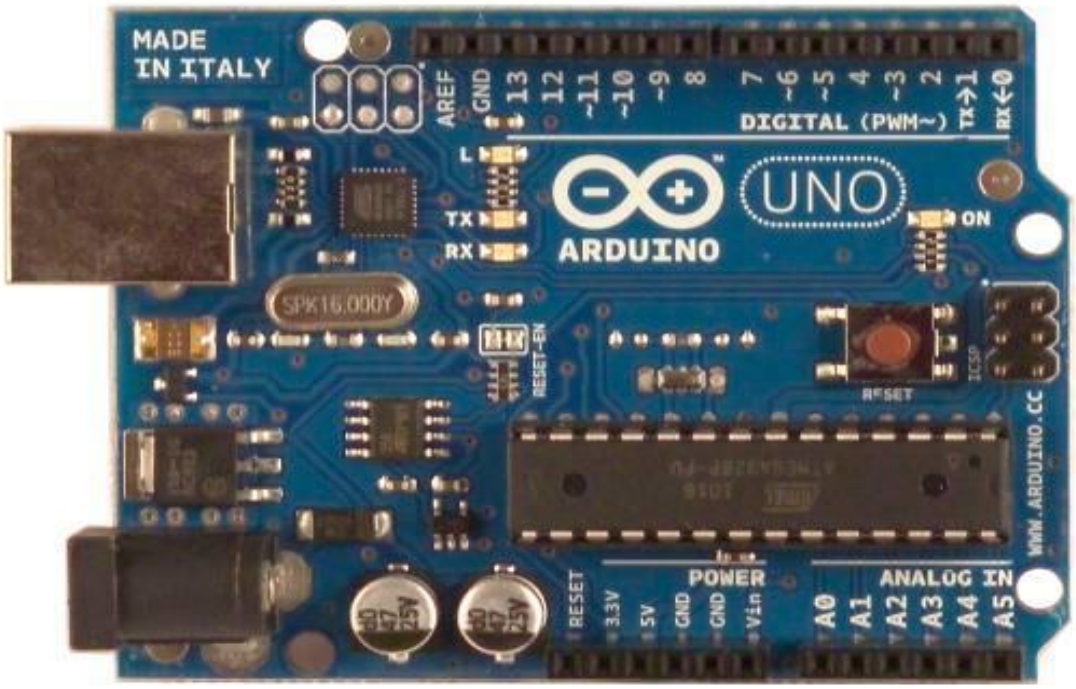


Рисунок 1.4 – Зовнішній вигляд Arduino UNO R3

Плати можуть бути зібрані користувачем самостійно. Програмне забезпечення доступне для безкоштовного завантаження. Вихідні креслення схем (файли CAD) є загальнодоступними.

Платформа має 14 цифрових входів/виходів (6 з яких можуть використовуватися як виходи широтно-імпульсної модуляції), 6 аналогових входів, кварцовий

генератор 16 МГц, роз'єм USB, силовий роз'єм, роз'єм ICSP та кнопку перезавантаження. Кожен з 14 цифрових виводів UNO може бути налаштований як вхід або вихід, використовуючи функції pinMode (), digitalWrite (), digitalRead (). Кожен вивід має навантажувальний резистор (за замовчуванням відключений) 20-50 кОм та може пропускати струм до 40 мА. Для роботи необхідно підключити платформу до комп'ютера за допомогою кабелю USB, або подати живлення за допомогою адаптера AC/DC чи батареї.

Платформа може працювати при зовнішньому живленні від 6 В до 20 В. При напрузі живлення нижче 7 В, вивід 5V може видавати менше 5 В, при цьому платформа може працювати нестабільно. При використанні напруги вище 12 В регулятор напруги може перегрітися та пошкодити плату. Рекомендований діапазон від 7 В до 12 В.

Деякі виводи мають особливі функції:

– VIN. Вхід використовується для подачі живлення від зовнішнього джерела (при відсутності 5 В від роз'єму USB або іншого регульованого джерела живлення). Подача напруги живлення відбувається через даний вивід;

– 5V. Регульоване джерело напруги, що використовується для живлення мікроконтролера і компонентів на платі. Живлення може подаватися від виводу VIN через регулятор напруги, або від роз'єму USB, або іншого регульованого джерела напруги 5 В;

– 3V3. Напруга на виводі 3.3 В генерується вбудованим регулятором на платі. Максимальне споживання струму 50 мА;

– GND (загальний вивід "земля");

– послідовна шина: 0 (RX) та 1 (TX). Виводи використовуються для отримання (RX) і передачі (TX) даних TTL. Дані виводи підключені до відповідних роз'ємів мікросхеми послідовної шини ATmega8U2 USB-to-TTL.

– зовнішнє переривання: 2 та 3. Дані виводи можуть бути налаштовані на виклик переривання або на меншому значенні, або по передньому чи задньому фронті, або при зміні значення. Детальна інформація знаходиться в описі функції attachInterrupt ();

– ШІМ: 3, 5, 6, 9, 10, і 11. Будь-який з виводів забезпечує ШІМ з роздільною здатністю 8 біт за допомогою функції analogWrite ();

– SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). За допомогою даних виводів здійснюється зв'язок SPI, для чого використовується бібліотека SPI;

– LED: 13. Вбудований світлодіод, підключений до цифрового виводу 13. Якщо значення на виводі має високий потенціал, то світлодіод горить.

На платформі UNO встановлені 6 аналогових входів (позначених як A0…A5), кожен з роздільною здатністю 10 біт (тобто може приймати 1024 різних значення). Стандартно виводи мають діапазон вимірювання до 5 В відносно землі, проте є можливість змінити верхню межу за допомогою виводу AREF і функції analogReference ().

Деякі виводи мають додаткові функції:

– I2C: 4 (SDA) і 5 (SCL). За допомогою виводів здійснюється зв'язок I2C (TWI), для створення якої використовується бібліотека Wire.

Додаткова пара виводів платформи:

– AREF. Опорна напруга для аналогових входів. Використовується з функцією analogReference ();

– Reset. Низький рівень сигналу на виводі перезавантажує мікроконтролер. Звичайно застосовується для підключення кнопки перезавантаження на платі розширення, що закриває доступ до кнопки на самій платі Arduino.

Основні функції для роботи зі світлодіодами

Для роботи зі світлодіодом необхідно знати і вміти користуватися такими функціями і константами:

– оператор setup();

– оператор loop();

– функція pinMode();

– функція digitalWrite();

– функція delay();

– константи OUTPUT, HIGH, LOW.

Далі наведено код програми найпростішого прикладу блимання вбудованим у плату Arduino світлодіодом, який підключено до 13 виводу:

```
void setup()
{
pinMode(13, OUTPUT);
}
```

Приведена функція виконується на початку роботи програми (після запуску мікроконтролеру). Тобто послідовно виконується кожна команда, яка знаходиться між фігурними скобками цієї функції. Наприкінці кожної строки необхідно поставити символ закінчення команди “;”. Функція setup містить команду pinMode(13, OUTPUT) . Ця команда налаштовує 13 порт Arduino, як вивід. Порт 13 знаходиться на верхній колодці портів Arduino. Після функції setup виконується функція loop.

```
void loop()
{
digitalWrite(13, HIGH); // вмикаємо світлодіод
delay(1000); // чекаємо секунду
digitalWrite(13, LOW); // вимикаємо світлодіод
delay(1000); // чекаємо секунду
}
```

На відміну від setup, функція loop постійно повторюється – як тільки послідовно виконані всі команди в дужках, функція запускається знову. Функція loop для цього прикладу складається з чотирьох команд:

1) на порт 13 подається напруга (5 В) – світлодіод вмикається;

2) затримка до виконання наступної команди 1000 мс (1 сек.);

3) порт 13 з’єднується із “землею” – світлодіод вимикається;

4) ще одна затримка на 1 сек.

Після виконання усіх чотирьох команд, знову виконується перша команда (включення світлодіоду) і таким чином блимання світлодіоду (на рис. 1.5 обведений червоною лінією) продовжується до тих пір, поки Arduino включена, або поки не буде натиснута кнопка RESET.

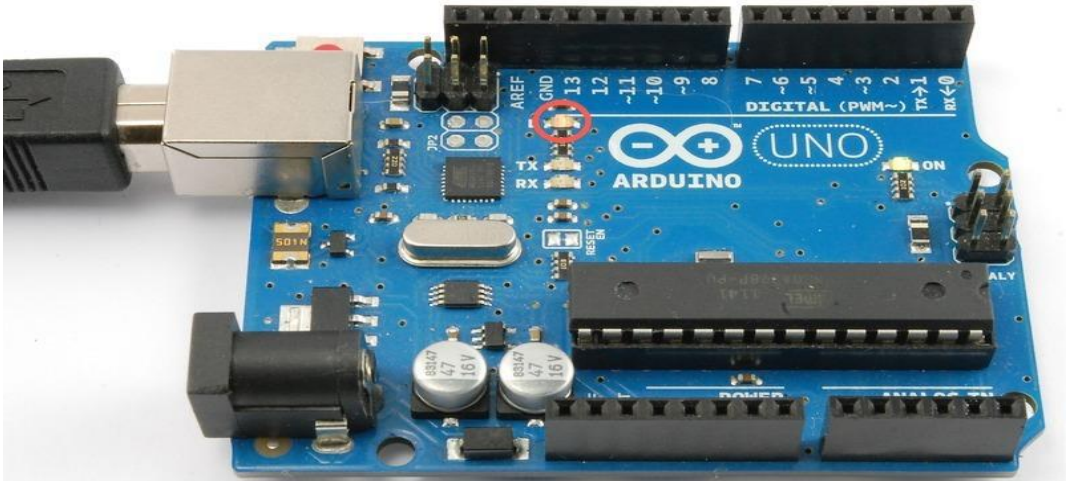


Рисунок 1.5 – Блимання світлодіоду на Arduino UNO R3

3. Ознайомлення із процесом блимання світлодіодом, розташованим на макетній платі за допомогою мікропроцесора ATMega328

Для забезпечення блимання світлодіода на Arduino і управління ним знадобиться:

– плата Arduino;

– монтажна плата (breadboard);

– два дроти «тато-тато»;

– світлодіод;

– резистор 200 Ом.

Монтажна плата (breadboard) являє собою сітку з гнізд, які зазвичай з’єднуються так, як показано на рис. 1.6.

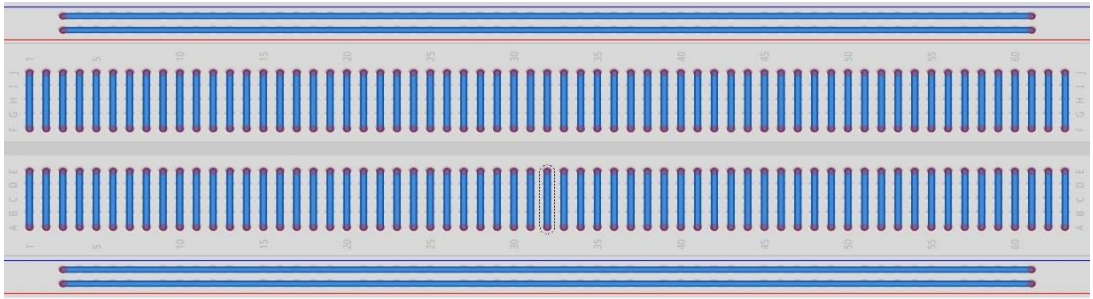


Рисунок 1.6 – З’єднання на монтажній платі (breadboard)

Схема підключення світлодіода на окремій монтажній платі для забезпечення блимання світлодіода на Arduino наведена на рис. 1.7.

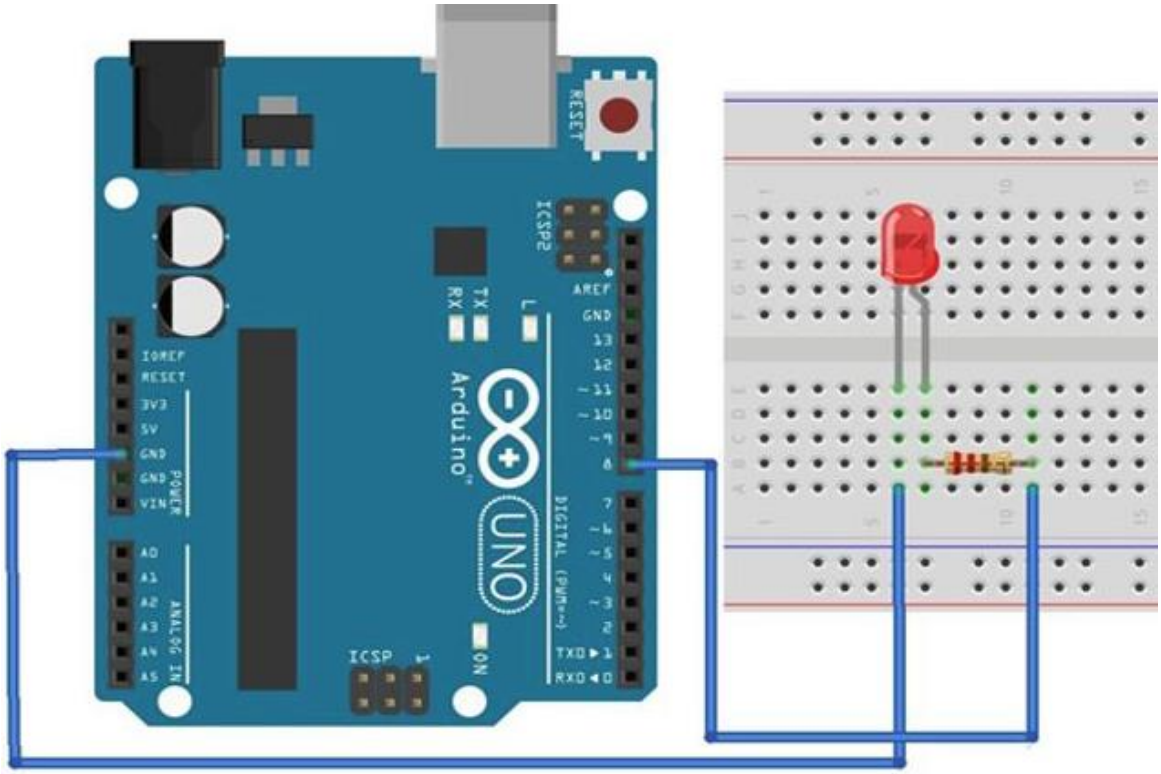


Рисунок 1.7 – Схема підключення світлодіода

Для роботи цієї моделі може бути використана наступна програма:

```
int led = 8;
void setup()
{
pinMode(led, OUTPUT);
}
void loop()
{
digitalWrite(led, HIGH);
delay(1000);
digitalWrite(led, LOW);
delay(1000);
}
```

Далі те ж саме з коментарями:

```
int led = 8; // оголошення змінної цілого типу, що містить номер порту, до якого підключено другий дріт
void setup () // обов’язкова процедура setup, яка запускається на початку програми; оголошення процедур починається словом void
{
pinMode (led, OUTPUT); // оголошення використовуваного порту, led – номер
```

порту, другий аргумент – тип використання порту: на вхід (INPUT) або на вихід (OUTPUT)

}

void loop () // обов’язкова процедура loop, що запускається циклічно після процедури setup

{

digitalWrite (led, HIGH); // ця команда використовується для включення або виключення напруги на цифровому порту; led – номер порту, другий аргумент – включення (HIGH) або вимикання (LOW)

delay (1000); // ця команда використовується для очікування між діями, аргумент – час очікування в мілісекундах

digitalWrite (led, LOW);

delay (1000);

Схема в зборі показана на рис. 1.8.

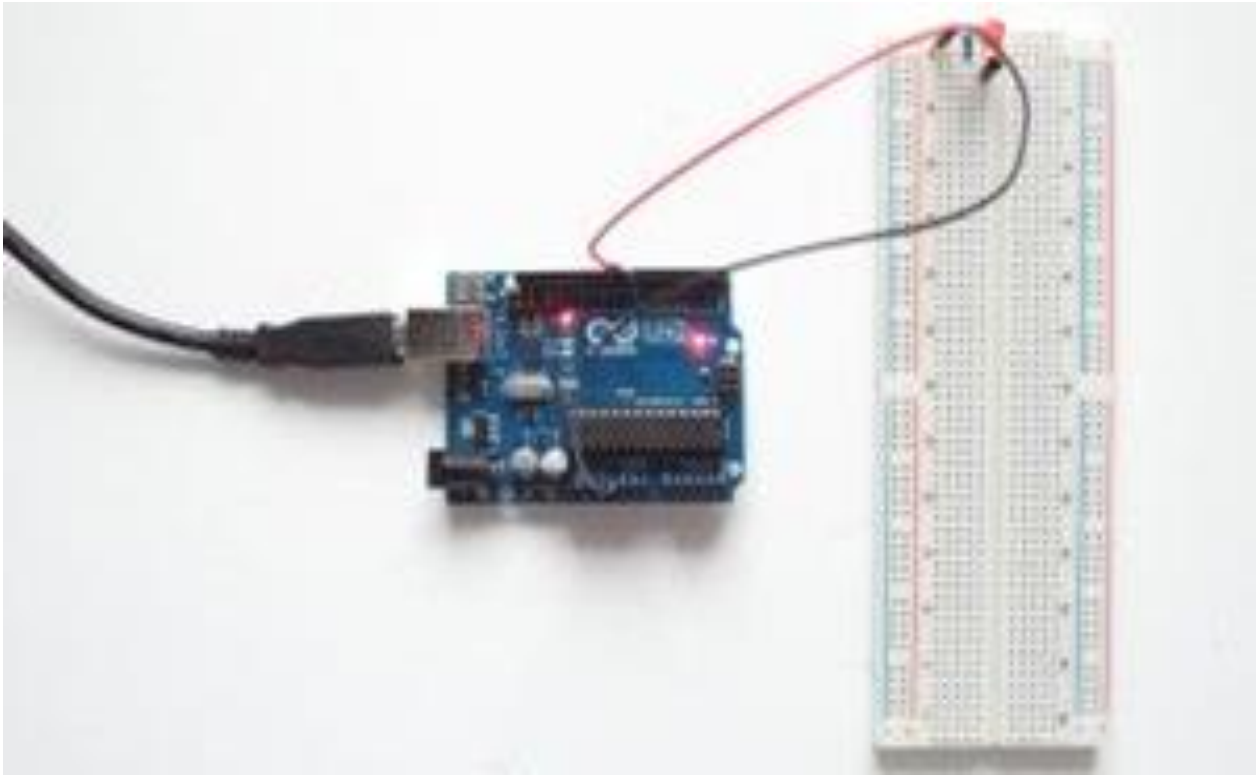


Рисунок 1.8 – Блимання світлодіода, розташованого на макетній платі

4. Індивідуальне завдання

Забезпечити блимання двох світлодіодів зазначеного кольору на визначених портах із заданим часом горіння та не горіння кожного світлодіода. Варіанти завдань наведені у табл. 1.3.

Таблиця 1.3 – Варіанти завдань

№ вар.	Світлодіод 1				Світлодіод 2			
	N порту	Колір	Час горіння, с	Час не горіння, с	N порту	Колір	Час горіння, с	Час не горіння, с
1	2	червоний	0,5	1	4	синій	1	0,8
2	3	жовтий	0,2	0,2	5	червоний	0,8	1,6
3	4	синій	0,7	1	6	жовтий	1,5	0,7
4	5	червоний	2	1	7	синій	0,5	0,5
5	6	жовтий	1	0,5	8	червоний	0,3	0,3
6	7	синій	0,4	0,4	9	жовтий	1	2
7	8	червоний	1	0,1	7	синій	2	0,4
8	9	жовтий	1,5	0,5	6	червоний	0,5	0,8
9	2	синій	0,6	1,2	5	жовтий	0,8	1,6
10	3	червоний	1,4	0,7	4	синій	0,5	1,8

Порядок виконання роботи

1. Ознайомитися з основними принципами підключення світлодіодів до мікропроцесора та кольоровим маркуванням резисторів.

2. Ознайомитися із процесом блимання вбудованим світлодіодом на платі Arduino UNO R3.

3. Ознайомитися із процесом блимання світлодіода, розташованим на макетній платі за допомогою мікропроцесора ATMega328.

4. Розробити програму блимання двома світлодіодами згідно індивідуального завдання.

5. Розробити модель схеми згідно індивідуального завдання у середовищі Tinkercad та перевірити її роботу.

6. Розробити модель схеми згідно індивідуального завдання у середовищі PROTEUS та перевірити її роботу.

7. Зібрати схему на монтажній платі згідно індивідуального завдання, ввести програму в Arduino UNO R3, запустити та перевірити її роботу.

8. Перевірити правильність функціонування програми у середовищах моделювання та на реальній схемі.

9. Оформити звіт про роботу.

Зміст звіту

1. Тема лабораторної роботи.

2. Мета роботи.

3. Індивідуальне завдання.

4. Програма та спрощена блок-схема алгоритму організації процесу блимання вбудованим світлодіодом на платі Arduino UNO R3.

5. Програма та спрощена блок-схема алгоритму організації процесу блимання світлодіодом, розташованим на макетній платі за допомогою мікропроцесора ATMega328.

6. Програма та спрощена блок-схема алгоритму організації процесу блимання двома світлодіодами згідно індивідуального завдання.

7. Працююча модель схеми для блимання двома світлодіодами згідно індивідуального завдання у середовищі Tinkercad.

8. Працююча модель схеми для блимання двома світлодіодами згідно індивідуального завдання у середовищі PROTEUS.

9. Фото працюючої зібраної схеми на монтажній платі для блимання двома світлодіодами згідно індивідуального завдання.

10. Висновки.

Практична робота 2

ДОСЛІДЖЕННЯ ОРГАНІЗАЦІЇ ВИВОДУ ІНФОРМАЦІЇ НА СЕМИСЕГМЕНТНІ ІНДИКАТОРИ НА МІКРОПРОЦЕСОРІ ATMEGA328

Мета роботи: Дослідження організації процесу виводу інформації на семисегментні індикатори на мікропроцесорі ATMega328 на платформі Arduino. Одержання практичних навичок видачі інформації на семисегментні індикатори з використанням мікропроцесора ATMega328 на платі Arduino UNO R3.

Теми для попереднього проороблення:

– теоретичні відомості про семисегментні світлодіодні індикатори;

– теоретичні відомості про динамічну індикацію.

1. Ознайомлення із основними принципами підключення світлодіодів до мікропроцесора

Семисегментний світлодіодний індикатор відображає символ за допомогою семи світлодіодів – сегментів цифри. Восьмий світлодіод засвічує децимальну точку (рис. 2.1). Сегменти позначаються латинськими літерами від «A» до «H».

Аноди або катоди кожного світлодіода об’єднуються в індикаторі та утворюють загальний провід. Тому існують індикатори із загальним анодом (ЗА) (рис. 2.3, *а*) та загальним катодом (ЗК) (рис. 2.3, *б*).

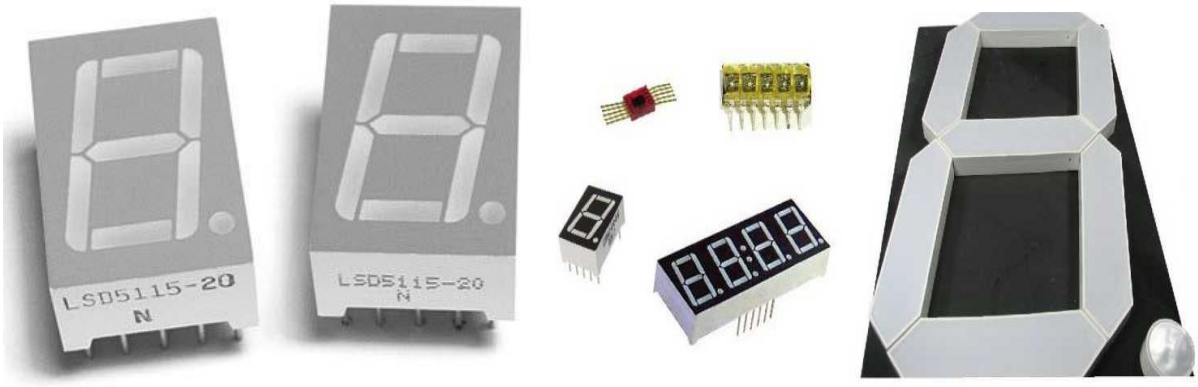


Рисунок 2.1 – Семисегментні світлодіодні індикатори різних типів

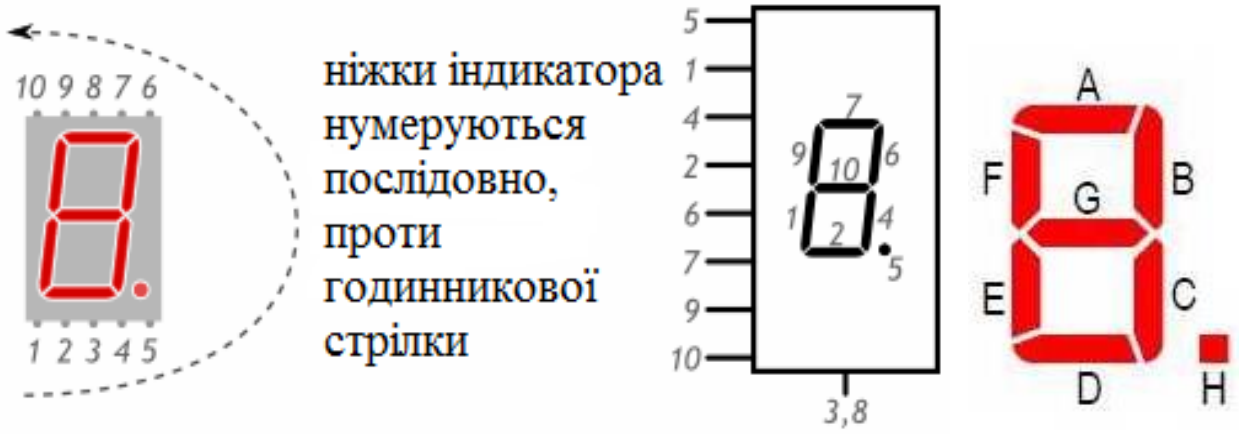
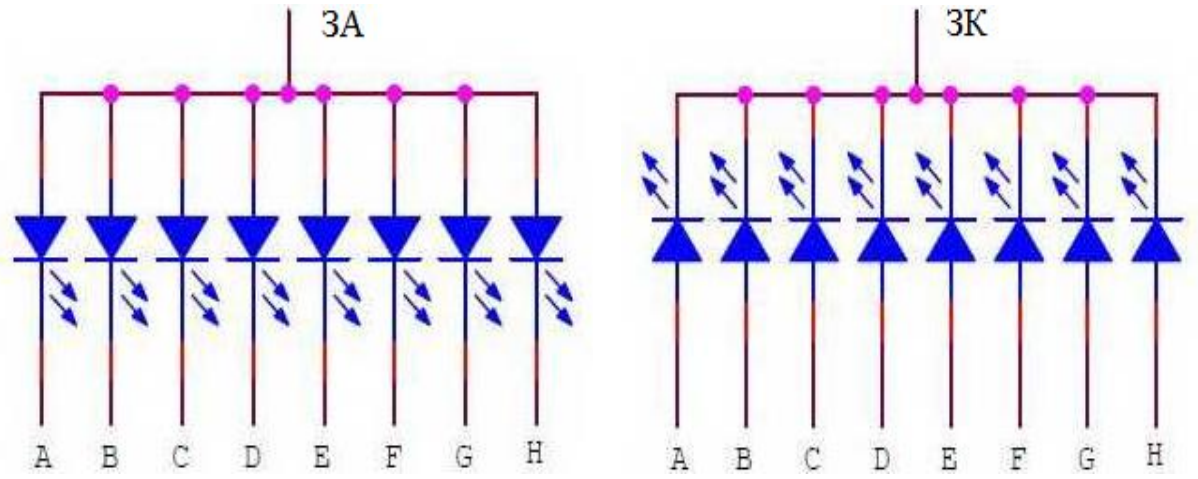


Рисунок 2.2 – Нумерація ніжок та позначення сегментів
у семисегментному світлодіодному індикаторі

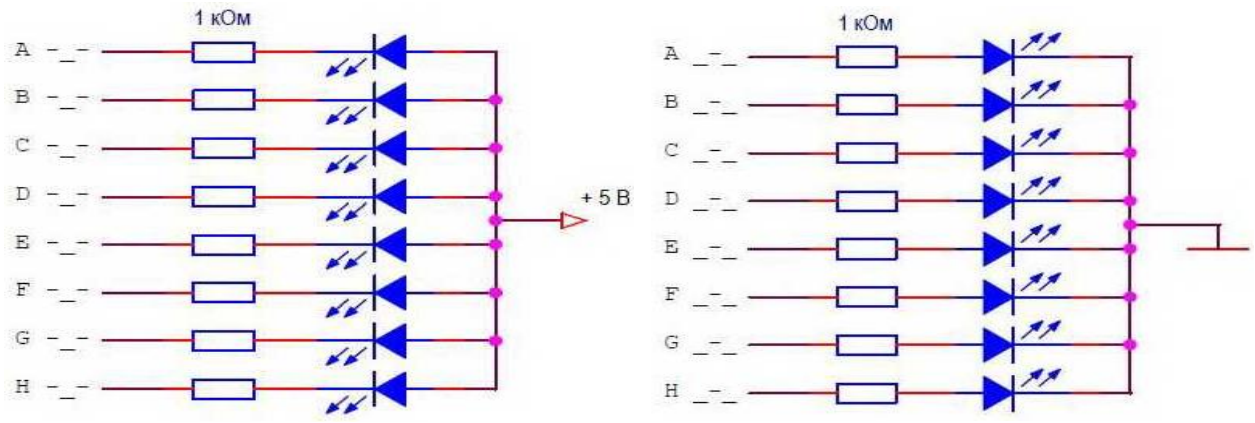


а) *б)*

Рисунок 2.3 – Схема семисегментного світлодіодного індикатора
з загальним анодом (*а*) та загальним катодом (*б*)

При статичному підключенні світлодіодні індикатори необхідно під’єднувати до мікроконтролеру через резистори (від 200 Ом до 1,5 кОм), які обмежують струм. В залежності від виду індикатора (із загальним катодом чи загальним анодом) змінюється полярність живлення та сигналів управління (рис. 2.4).

Для підключення кожного семисегментного індикатора до мікроконтролера потрібно вісім виводів. Якщо індикаторів (розрядів) 3-4, то завдання стає практично не здійсненним, тому що не вистачить виводів мікроконтролера.



а) *б)*

Рисунок 2.4 – Схема підключення семисегментного світлодіодного індикатора з загальним анодом (*а*) та загальним катодом (*б*)

При підключення багаторозрядного семисегментного світлодіодного індикатора до мікроконтролера використовується мультиплексований режим, тобто режим динамічної індикації.

Виводи однойменних сегментів кожного індикатора об’єднуються. Виходить матриця світлодіодів, підключених між виводами сегментів і загальними виводами індикаторів. Схема мультиплексованого (динамічного) управління двохрозрядним індикатором із загальним анодом наведена на рис. 2.5.

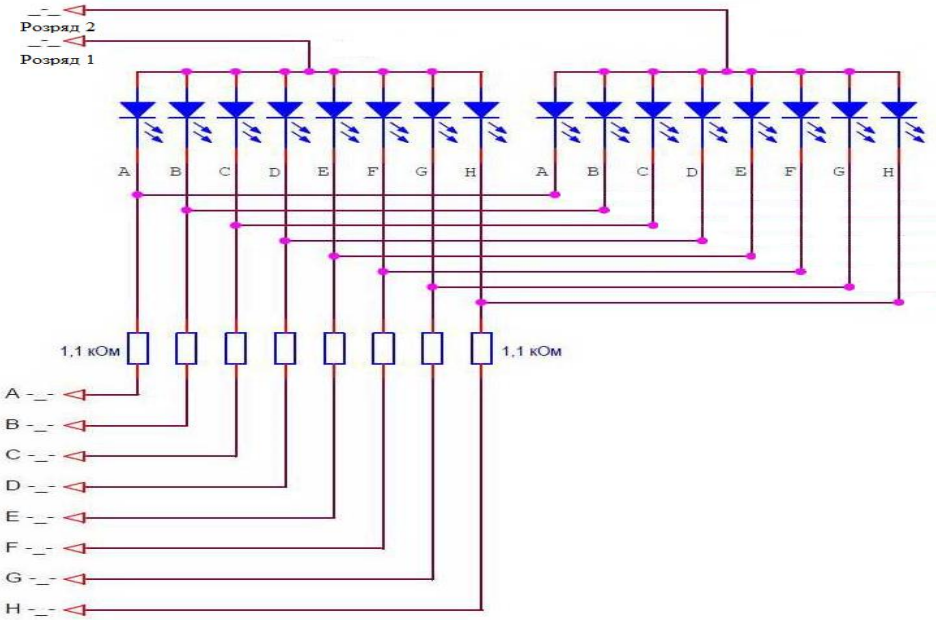


Рисунок 2.5 – Схема підключення двохрозрядного індикатора з загальним анодом

Схема мультиплексованого (динамічного) управління двохрозрядним індикатором із загальним катодом наведена на рис. 2.6.

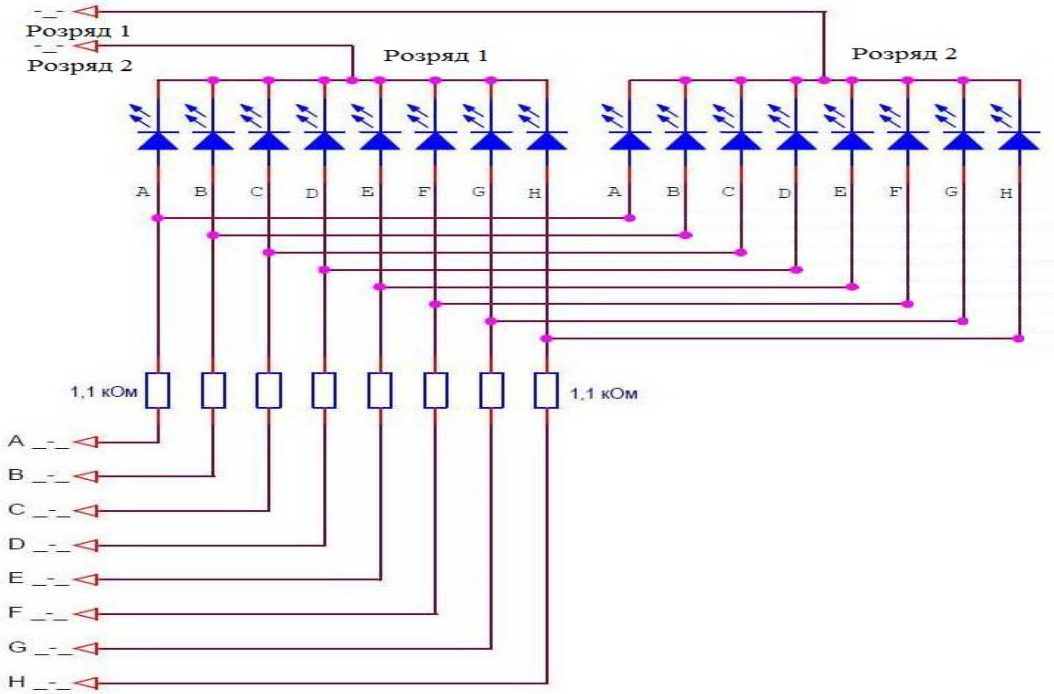


Рисунок 2.6 – Схема підключення двохрозрядного індикатора із загальним катодом

Відносно схеми із загальним анодом змінюється полярність всіх сигналів. Тепер на загальний провід активного розряду подається низький рівень, а на сегменти, які повинні світитися – високий рівень. На підключення двохрозрядного семисегментного індикатора вистачило 10 виводів замість 18, а у випадку чотирьохрозрядного індикатора буде 12 виводів замість 36.

2. Ознайомлення із процесом виводу цифр та символів на семисегментний індикатор за допомогою мікропроцесора ATMega328

Для підключення однорозрядного світлодіодного індикатора до Arduino будуть задіяні 7 цифрових виводів (кожен контакт A-G індикатора підключається до виводів Arduino через обмежувальний резистор). У наведеному випадку використовується семисегментний індикатор із загальним катодом (ЗК), загальний провід приєднуємо до землі. Схема розміщення виводів семисегментного

індикатора 5161AS приведена на рис. 2.7. На рис. 2.8 показана схема підключення однорозрядного семисегментного індикатора із ЗК до плати Arduino.

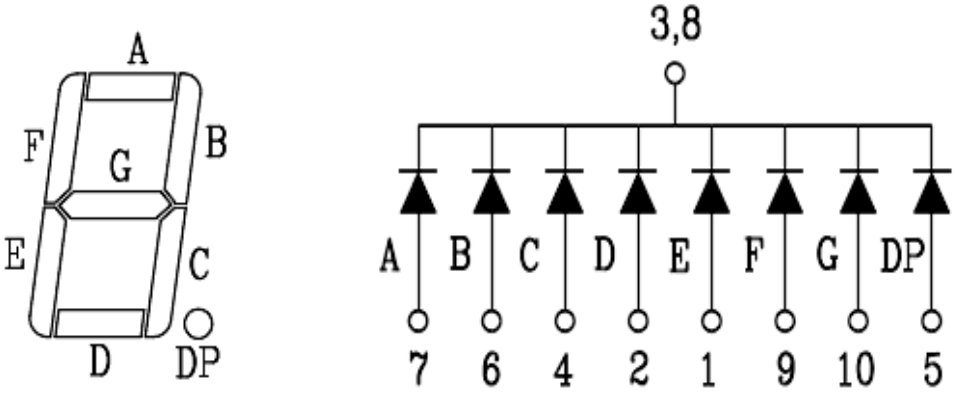


Рисунок 2.7 – Схема виводів семисегментного індикатора 5161AS

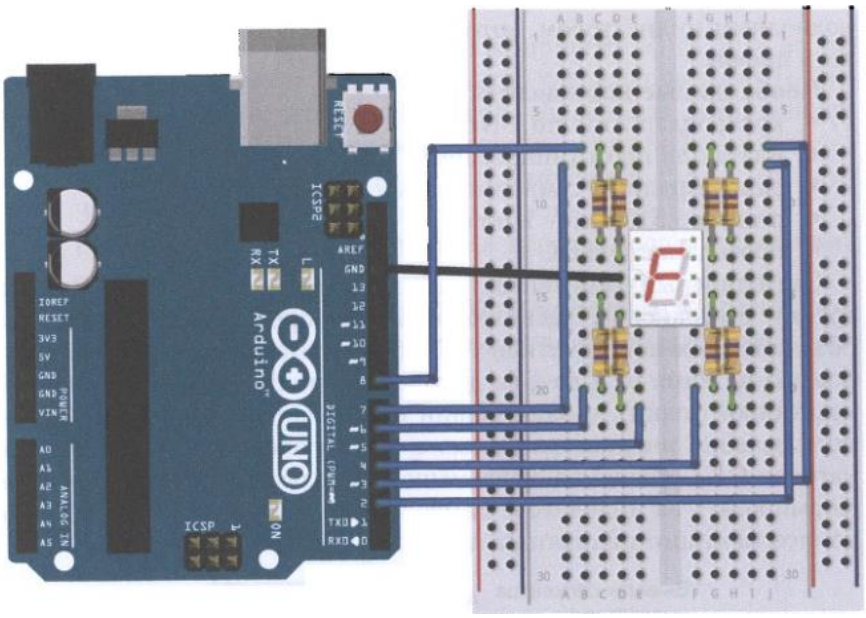


Рисунок 2.8 – Схема підключення індикатора до плати Arduino

На семисегментний індикатор в циклі будемо виводити цифри від 0 до 9 з паузою в 1 сек. Сформуємо масив значень для цифр 0…9, де старший розряд байта відповідає мітці сегмента A індикатора, а молодший – сегменту G:

byte numbers[10] = { B11111100, B01100000, B11011010, B11110010, B01100110, B10110110, B10111110, B11100000, B11111110, B11110110}; // змінна для зберігання значення поточної цифри

Відповідність відображуваного знаку даним порту наведена у табл. 2.1.

Таблиця 2.1 – Відповідність відображуваного знаку даним відповідного порта

Знак	Загальний анод		Загальний катод	
	Двійкова система	Десяткова система	Двійкова система	Десяткова система
«0»	00000011	3	11111100	252
«1»	10011111	159	01100000	96
«2»	00100101	37	11011010	218
«3»	00001101	13	11110010	242
«4»	10011001	153	01100110	102
«5»	01001001	73	10110110	182
«6»	01000001	65	10111110	190
«7»	00011111	31	11100000	224
«8»	00000001	1	11111110	254
«9»	00001001	9	11110110	246

Для перетворення цифри в дані для виведення значення на виводи Arduino будемо використовувати бітові операції мови:

bitRead (x, n); // отримання значення n розряду байта x

Нижче наведено програму, яка виводить на індикатор цифри від 0 до 9:

```
// список виводів для підключення до розрядів семисегментного індикатора
int pins[7]={2,3,4,5,6,7,8};
// значення для виводу цифр 0-9
byte numbers[10] = {B11111100, B01100000, B11011010, B11110010, B01100110,
B10110110, B10111110, B11100000, B11111110, B11100110};
// змінна для зберігання значення поточної цифри
int number=0;
void setup()
  {
  // сконфігурувати контакти як виходи
  for (int i=0;i<7;i++)
    pinMode(pins[i],OUTPUT);
  }
void loop()
  {
  showNumber(number);
  delay(1000);
  number=(number+1)%10;
  }
// функція виводу цифри на семисегментний індикатор
```

```
void showNumber(int num)
  {
  for (int i=0;i<7;i++)
    {
    if(bitRead(numbers[num],7-i)–HIGH) // включити сегмент
      digitalWrite(pins[i],HIGH);
    else // виключити сегмент
      digitalWrite(pins[i],LOW);
    }
  }
```

Порядок підключення:

1) підключаємо семисегментний індикатор до плати Arduino згідно рис. 2.8;

2) завантажуємо в Arduino наведену вище програму;

3) спостерігаємо за виведенням цифр на екран семисегментного індикатора.

3. Ознайомлення із процесом динамічної індикації на матрицю семисегментних індикаторів з використанням мікропроцесора ATMega328 на платі Arduino UNO R3

Розглянемо процес динамічної індикації на чотирьохрозрядній матриці семисегментних індикаторів із загальним анодом 3641BS. Зовнішній вигляд індикатора наведено на рис. 2.9, схема наведена на рис. 2.10, розміщення виводів – на рис. 2.11.

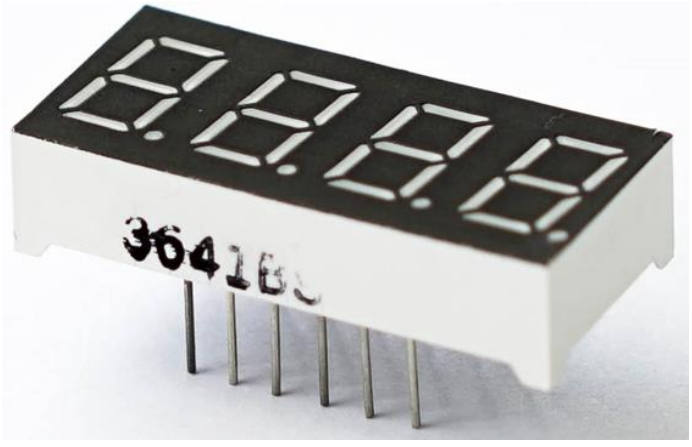


Рисунок 2.9 – Зовнішній вигляд чотирьохрозрядної матриці семисегментних індикаторів

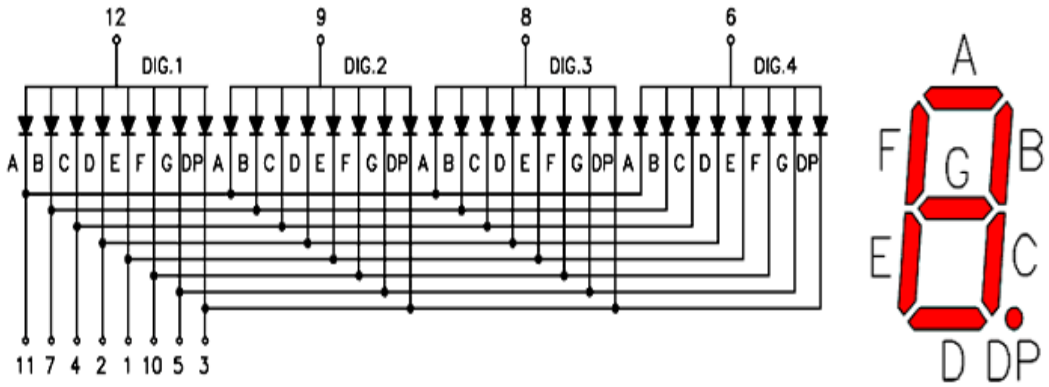
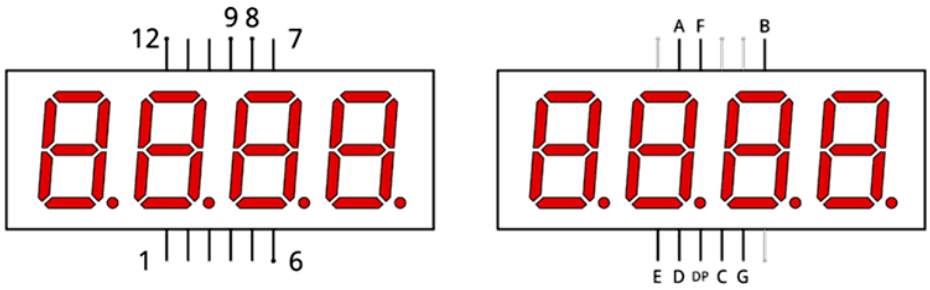


Рисунок 2.10 – Схема чотирьохрозрядної матриці семисегментних індикаторів



а) *б)*

Рисунок 2.11 – Розміщення виводів чотирьохрозрядної матриці семисегментних індикаторів

На рис. 2.11 (*а*) – 12, 9, 8, 6 – це виводи-аноди для кожного розряду, на рис. 2.11 (*б*) – відповідність виводів індикатора його сегментам.

За аналогією з семисегментним індикатором для відображення потрібної інформації необхідно по черзі подавати напругу на аноди, а катодами формувати потрібну інформацію – і так для кожного розряду по черзі. Робити це потрібно так швидко, щоб людині здавалося, що одночасно безперервно горять всі чотири розряди. Збираємо схему, як показано на рис. 2.12.

Перевіряємо працездатність схеми:

```
1   int anodPins[] = {A1, A2, A3, A4};
2   int segmentsPins[] = {5, 6, 7, 8, 9, 10, 11, 12};
3
4   void setup() {
5     for (int i = 0; i < 4; i++) {
6       pinMode(anodPins[i], OUTPUT);
7     }
```

```
8     for (int i = 0; i < 8; i++) {
9       pinMode(segmentsPins[i], OUTPUT);
10    }
11  }
12
13  int seg[] = {0, 1, 1, 0, 1, 1, 1, 0}; // літера H
14
15  void loop() {
16     // відображаємо літеру H на всіх розрядах
17    for (int i = 0; i < 4; i++) {
18    for (int k = 0; k < 8; k++) {
19       digitalWrite(segmentsPins[k], ((seg[k] == 1) ? LOW : HIGH));
20     }
21     digitalWrite(anodPins[i], HIGH); // подали напругу на анод – всі
22     // індикатори горять
23     delay(1); // пауза
24     digitalWrite(anodPins[i], LOW); // зняли напругу з аноду, щоб
25     // переключення сегментів не викликало мерехтіння
26    }
27  }
```

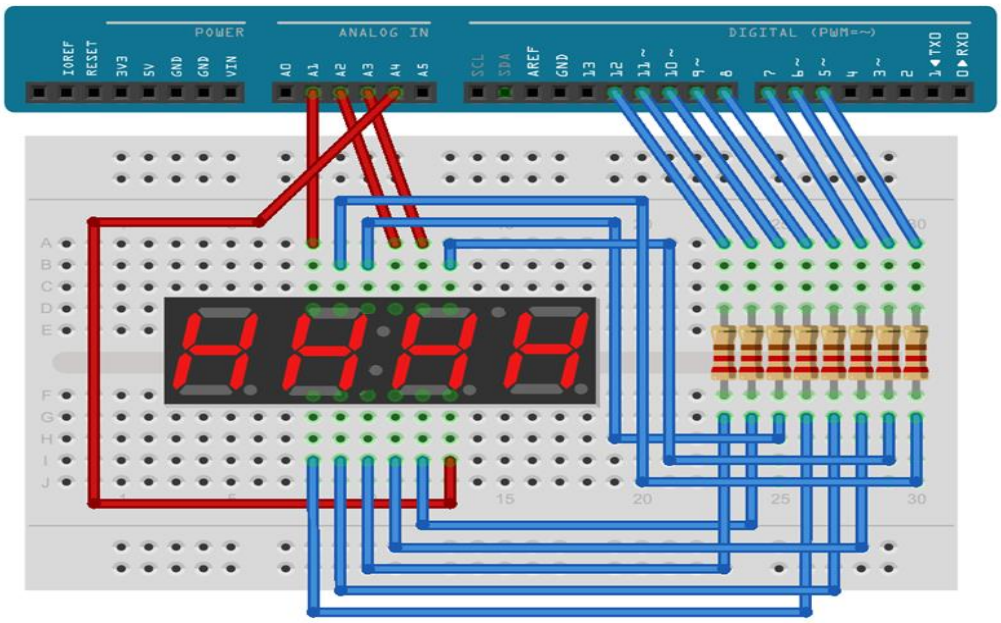


Рисунок 2.12 – Схема підключення чотирьохрозрядної матриці семисегментних індикаторів до Arduino UNO

Модифікуємо програмний код для відображення чисел:

```
1   int anodPins[] = {A1, A2, A3, A4}; // задаємо піни для кожного розряду
2   int segmentsPins[] = {5, 6, 7, 8, 9, 10, 11, 12}; // задаємо піни для кожного
```

```
3    //сегмента (із 7 + 1 (крапка))
4   void setup() {
5     // всі виходи програмуємо як OUTPUT
6     for (int i = 0; i < 4; i++) {
7       pinMode(anodPins[i], OUTPUT);
8     }
9     for (int i = 0; i < 8; i++) {
10      pinMode(segmentsPins[i], OUTPUT);
11    }
12  }
13  //{A, B, C, D, E, F, G, DP} – розпіновка сегментів
14  int seg[10][8] = {
15    {1, 1, 1, 1, 1, 1, 0, 0}, //Цифра 0
16    {0, 1, 1, 0, 0, 0, 0, 0}, //Цифра 1
17    {1, 1, 0, 1, 1, 0, 1, 0}, //Цифра 2
18    {1, 1, 1, 1, 0, 0, 1, 0}, //Цифра 3
19    {0, 1, 1, 0, 0, 1, 1, 0}, //Цифра 4
20    {1, 0, 1, 1, 0, 1, 1, 0}, //Цифра 5
21    {1, 0, 1, 1, 1, 1, 1, 0}, //Цифра 6
22    {1, 1, 1, 0, 0, 0, 0, 0}, //Цифра 7
23    {1, 1, 1, 1, 1, 1, 1, 0}, //Цифра 8
24    {1, 1, 1, 1, 0, 1, 1, 0}  //Цифра 9
25  };
26  int t = 0;
27  int digid = 0;
28  void loop() {
29    t += 1;
30    if (t > 9999) t = 0;
31    if ((t % 1000) == 0) {
32      digid = t / 1000; // щосекунди  відображаємо цифри підряд
33    }
34    for (int i = 0; i < 4; i++) { // кожний розряд по черзі
35    for (int k = 0; k < 8; k++) { // кожний сегмент по черзі
36        digitalWrite(segmentsPins[k], ((seg[digid][k] == 1) ? LOW : HIGH));
37      }
38      digitalWrite(anodPins[i], HIGH);
39      delay(1);
40      digitalWrite(anodPins[i], LOW);
41    }
42  }
```

Схема підключення чотирьох однорозрядних семисегментних індикаторів і результат роботи програми при виведенні цифри 2 показано на рис. 2.13.

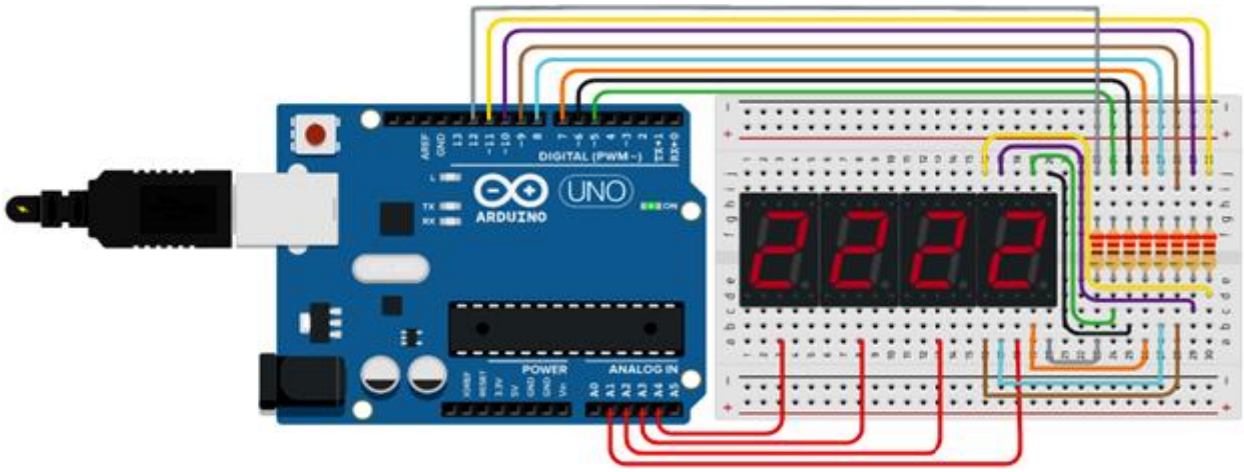


Рисунок 2.13 – Схема підключення чотирьох семисегментних індикаторів до Arduino UNO

Якщо потрібно виводити різні символи на різні розряди індикатора, модифікуємо програму з використанням макровизначення sizeofArray(a), що виконує операцію sizeof array / sizeof array[0], яка визначає кількість елементів у масиві array (sizeof(array) – повертає кількість байтів, що займає масив; оскільки кожен елемент може займати більше 1 байта простору, то потрібно розділити результат на розмір одного елемента (sizeof (array [0])).

Програмний код для відображення різних символів на різних розрядах індикатора:

```
#define sizeofArray(a) (sizeof a / sizeof a[0])

const int anodPins[] = {A1, A2, A3, A4}; // задаємо піни для кожного розряду
const int segmentPins[] = {5, 6, 7, 8, 9, 10, 11, 12}; // задаємо піни для кожного сегменту
// 1234. – інформація, що висвічується на чотирьохрозрядному індикаторі
const int symbols[sizeofArray(anodPins)][sizeofArray(segmentPins)] =
{//a, b, c, d, e, f, g, dp – сегменти індикатора
  {1, 0, 0, 1, 1, 0, 0, 0}, // 1 розряд: 4.
  {0, 0, 0, 0, 1, 1, 0, 1}, // 2 розряд: 3
  {0, 0, 1, 0, 0, 1, 0, 1}, // 3 розряд: 2
  {1, 0, 0, 1, 1, 1, 1, 1}  // 4 розряд: 1
};
void setup()
{
 // визначаємо піни для кожного розряду як вихідні
  for (int i = 0; i < sizeofArray(anodPins); i++)
    pinMode(anodPins[i], OUTPUT);
```

```
17  // визначаємо піни для кожного сегменту як вихідні
18   for (int i = 0; i < sizeofArray(segmentPins); i++)
19     pinMode(segmentPins[i], OUTPUT);
20  }
21  void setDisplay(const int symbol[])
22   // встановлення значень на пінах для кожного сегменту згідно визначеного рядка масиву symbol
23  {
24   for (int i = 0; i < sizeofArray(segmentPins); i++)
25     digitalWrite(segmentPins[i], symbol[i]);
26  }
27  void loop()
28  {
29   for (int i = 0; i < sizeofArray(anodPins); i++)
30   {
31     // встановлення значень на пінах для кожного сегменту згідно чергового рядка масиву symbol
32     setDisplay(symbols[i]);
33     // включення кожного розряду по черзі
34     digitalWrite(anodPins[i], HIGH);
35     // визначення часу включення кожного розряду по черзі
36     delay(5);
37     // виключення кожного розряду по черзі
38     digitalWrite(anodPins[i], LOW);
39   }
40  }
```

Схема і результат роботи програми показано на рис.2.14.

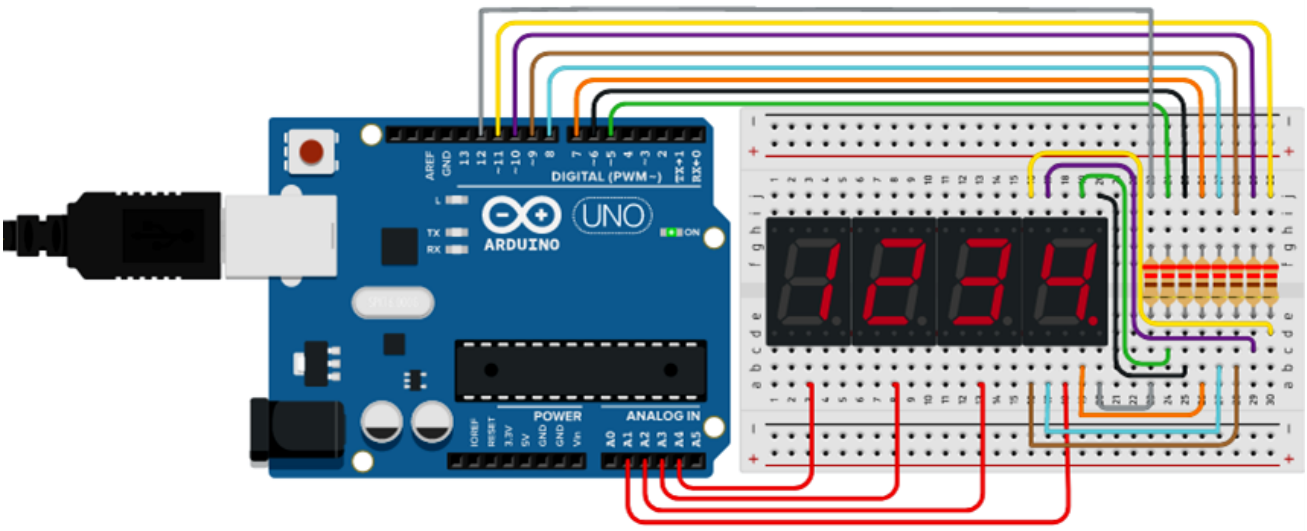


Рисунок 2.14 – Схема підключення чотирьох семисегментних індикаторів до Arduino UNO і результат роботи програми

4. Індивідуальне завдання

4.1. Забезпечити почергове виведення заданих символів на семисегментний світлодіодний індикатор. Варіанти завдань наведені у табл. 2.2.

4.2. Забезпечити виведення заданих символів на чотирьохрозрядний семисегментний світлодіодний індикатор. Варіанти завдань наведені у табл. 2.2.

Таблиця 2.2 – Варіанти завдань

№ варіанта	Інформація до п. 4.1	Інформація до п. 4.2
1	0124Г0	1, 5, 2, F
2	1Г1904	2, 4, C, .F
3	РН1234	3, 6, F, A
4	0ГРС01	3, 4, 5, 6
5	НПГО14	B, C, D, 3
6	ГПН813	D, F, F, 1
7	ОНП145	C, F, 8, 9
8	ГРУ128	A, B, F, 6
9	ПОГ061	9, 7, B, A

Порядок виконання роботи

1. Ознайомитися з процесом почергового виведення заданих символів на семисегментний світлодіодний індикатор за допомогою мікропроцесора ATMega328.

2. Ознайомитися з процесом виведення заданих символів на чотирьохрозрядний семисегментний світлодіодний індикатор за допомогою мікропроцесора ATMega328.

3. Розробити програми почергового виведення заданих символів на семисегментний світлодіодний індикатор та на чотирьохрозрядний індикатор згідно індивідуального завдання.

4. Розробити моделі схем згідно індивідуального завдання у середовищі Tinkercad та перевірити її роботу.

5. Розробити моделі схем згідно індивідуального завдання у середовищі PROTEUS та перевірити її роботу.

6. Зібрати схеми згідно індивідуального завдання, ввести програми в Arduino

UNO R3, запустити та перевірити їх роботу.

7. Перевірити правильність функціонування програм у середовищах моделювання та на реальній схемі.

8. Оформити звіт про роботу.

Зміст звіту

1. Тема лабораторної роботи.

2. Мета роботи.

3. Індивідуальне завдання.

4. Програма та спрощена блок-схема алгоритму організації процесу почергового виведення заданих символів на семисегментний світлодіодний індикатор за допомогою мікропроцесора ATMega328 згідно індивідуального завдання.

5. Програма і спрощена блок-схема алгоритму організації процесу виведення заданих символів на чотирьохрозрядний семисегментний світлодіодний індикатор за допомогою мікропроцесора ATMega328 згідно індивідуального завдання.

6. Працюючі моделі схем згідно індивідуального завдання у середовищі Tinkercad.

7. Працюючі моделі схем згідно індивідуального завдання у середовищі PROTEUS.

8. Фото працюючих зібраних на макетній платі схем згідно індивідуального завдання.

9. Висновки.

Практична робота 3

ДОСЛІДЖЕННЯ ОРГАНІЗАЦІЇ ВИВОДУ ІНФОРМАЦІЇ НА LCD ІНДИКАТОРИ НА МІКРОПРОЦЕСОРІ ATMEGA328

Мета роботи: Дослідження організації процесу виводу інформації на LCD індикатори на мікропроцесорі ATMega328 на платформі Arduino. Одержання практичних навичок видачі інформації на LCD індикатори з використанням мікропроцесора ATMega328 на платі Arduino UNO R3.

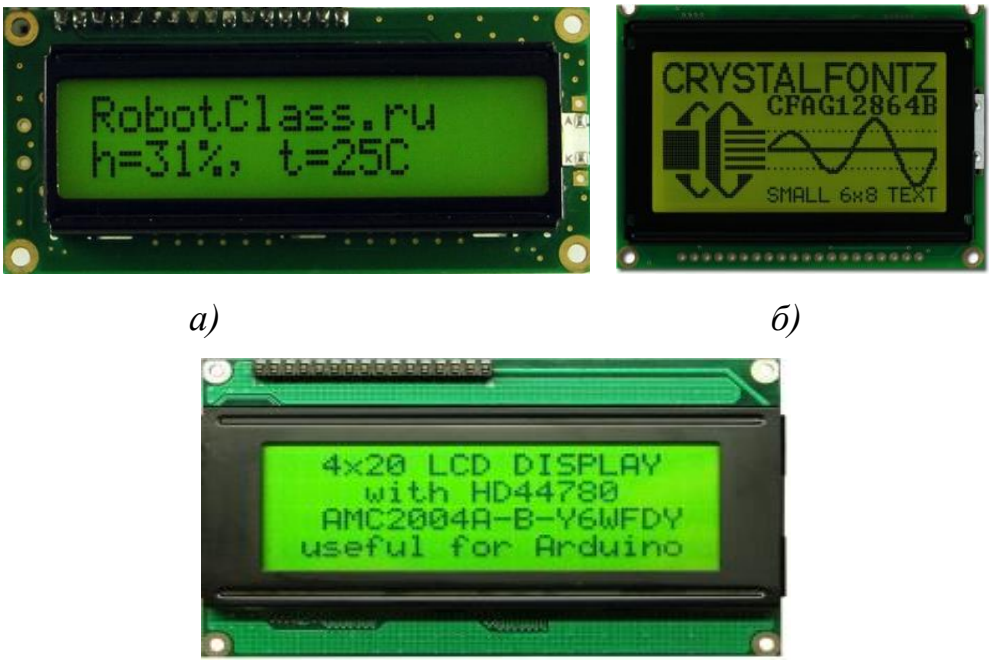
Теми для попереднього проороблення:

– теоретичні відомості про LCD індикатори;

– теоретичні відомості про динамічну індикацію.

1. Ознайомлення із основними принципами підключення LCD індикаторів до мікропроцесора

Електронні пристрої, призначені для візуального відображення цифрової, цифро-літерної або графічної інформації, називаються дисплеями. Розглянемо роботу з рідкокристалічними (LCD) дисплеями (рис. 3.1).



а) *б)*

в)

Рисунок 3.1 – Рідкокристалічні дисплеї різних типів

Рідкокристалічні індикатори, що відображають символьну інформацію, наприклад текст і числа, є найбільш недорогими і простими у використанні серед усіх LCD та називаються індикаторами. Вони мають різні розміри, вимірювані числом і довжиною рядків, що відображаються. Деякі мають підсвічування і дозволяють вибирати колір символів і фону. Будь-які рідкокристалічні індикатори з HD44780 від Hitachi, KS0066 від Samsung або сумісних з ними та напругою живлення підсвічування 5 В працюють із платою Arduino. Такі індикатори мають декілька модифікацій, які в тому числі можуть бути з різним підсвічуванням (блакитним, зеленим).

Текстові (рядкові) дисплеї відрізняються від графічних тільки логічним управлінням матрицею точок. У них точки заздалегідь згруповані в певні місця для виведення знаків, між якими залишені проміжки. Такі знакомісця утворюють один, два, або чотири рядки по 8, 12, 16, 20 і більше символів у кожному. Кожне знакомісце містить окремо керовану матрицю (наприклад, 5x8 точок), яка призначена для виведення одного символу. Конфігурація такого дисплея зашифрована в його назві, де цифри 1202, 1602, 1204 або схожі вказують на кількість рядків (в прикладах 2 або 4) і знаків в кожному рядку (12 і 16). Є також й однорядкові дисплеї такого типу, але досліджувати будемо більш вживані дворядкові, а саме – LCD 1602 з підсвічуванням, здатний відображати 2 рядки по 16 символів у кожному, зображений на рис. 3.2.

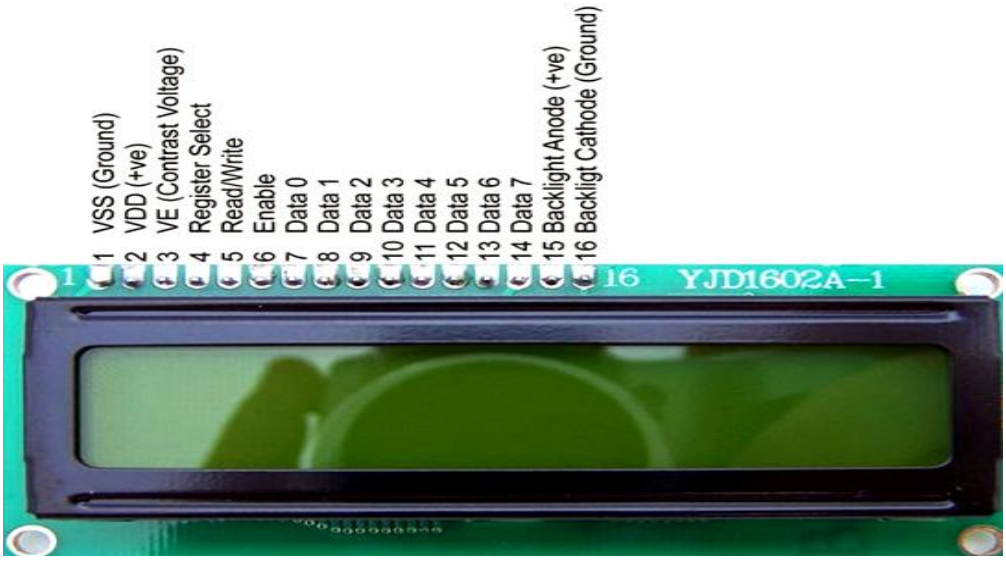


Рисунок 3.2 – Дворядковий рідкокристалічний індикатор LCD 1602

Кожен з виводів індикатора має своє призначення:

1 – VSS (GND), “земля” (“мінус живлення”);

2 – VDD (Vcc), живлення +5 В;

3 – VE (V_0), установка контрастності монітора (індикатора);

4 – RS, вибір регістра (команди, дані);

5 – R/W, вибір напрямку передачі даних (запис або читання даних);

6 – Enable (EN, E), синхронізація;

7-14 – DB0–DB7, шина даних;

15 – LED+, плюс підсвічування (+5 В);

16 – LED–, мінус підсвічування (“земля”, “мінус живлення”).

Вибраний індикатор має тип відображення з можливістю завантаження символів, світлодіодну підсвітку та управляється контролером HD44780. Схема підключення LCD 1602 до Arduino UNO наведена на рис. 3.3.

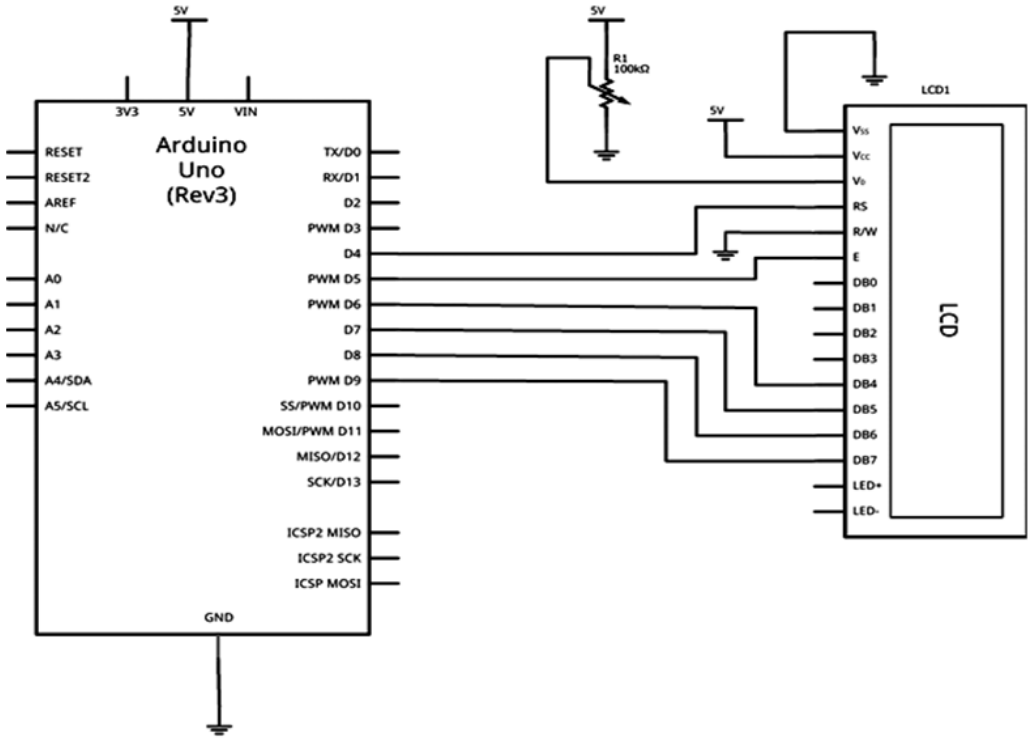


Рисунок 3.3 – Схема підключення LCD 1602 до Arduino UNO

Лінії VE (E), RS під’єднуються до цифрових виводів 5, 4 Arduino UNO, а чотири лінії даних DB4, DB5, DB6, DB7 підключаються до цифрових виводів 6, 7, 8, 9

контролера. Лінія “R / W” підключається до “землі” контролера (тому що нам потрібна тільки функція запису в пам’ять дисплея).

Розпіновка з’єднань між LCD 1602 та Arduino UNO показана на рис. 3.4, схема під’єднання – на рис. 3.5.

LCD 1602	1	2	4	6	11	12	13	14	15	16
Arduino UNO	GND	+5V	4	5	6	7	8	9	+5V	GND

Рисунок 3.4 – Розпіновка з’єднань між LCD 1602 та Arduino UNO

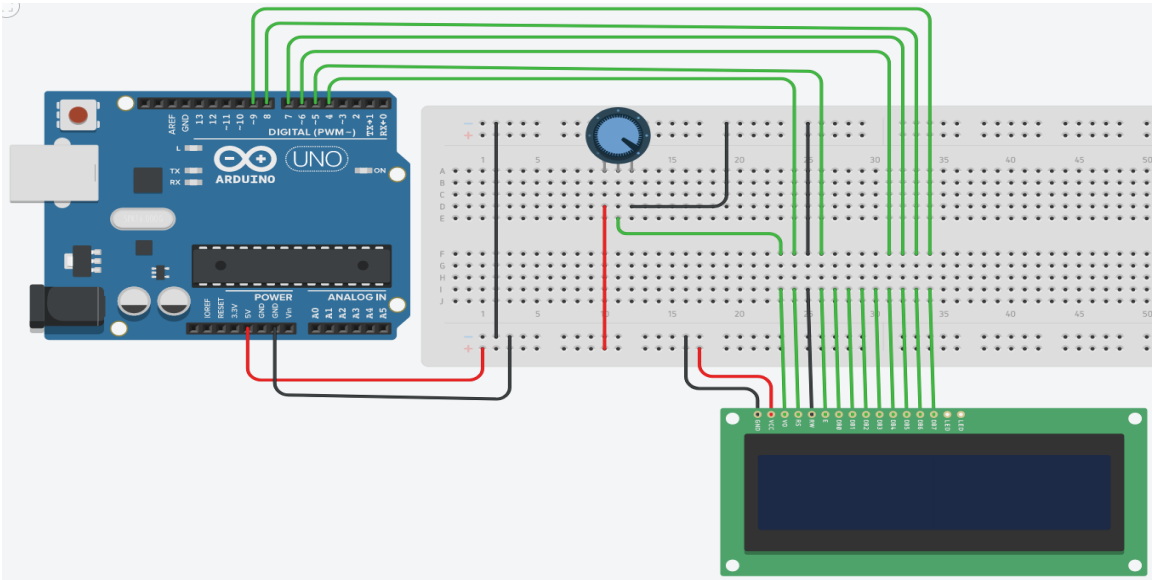


Рисунок 3.5 – Схема під’єднання LCD 1602 до Arduino UNO

Для виводу символів на екран використовується бібліотека LiquidCrystal.h. Після підключення зібраної схеми до персонального комп’ютера необхідно відрегулювати контрастність дисплея. Для цього треба виконати наступні дії:

– встановити потенціометр на макетної платі і підключити три його піна (рис. 3.6);

– підключити перший пін на потенціометрі до “землі” (“загальний вивід”) на макетній платі;

– підключити середній пін потенціометра до 3 піну на дисплеї (він позначений як V0);

– підключити третій пін на потенціометрі до “плюса” (+5 В) на макетній платі.

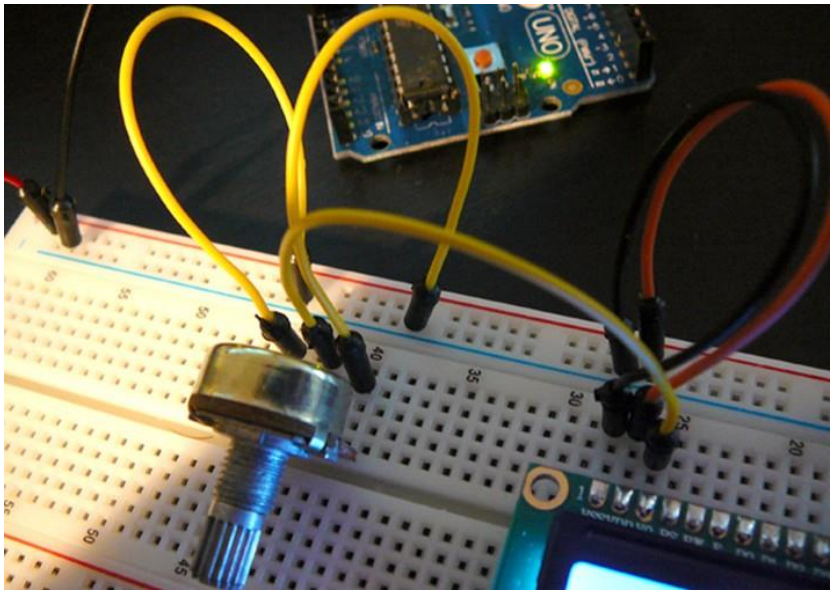


Рисунок 3.6 – Підключення потенціометра R1 для відрегулювання контрастності

Після подачі живлення на плату через USB-кабель на дисплеї перший рядок повинен заповнитися прямокутниками. Якщо їх не видно, треба трохи повернути ручку потенціометра зліва направо, щоб відрегулювати контраст. Правильно відрегульований дисплей виглядає приблизно так, як показано на рис. 3.7.

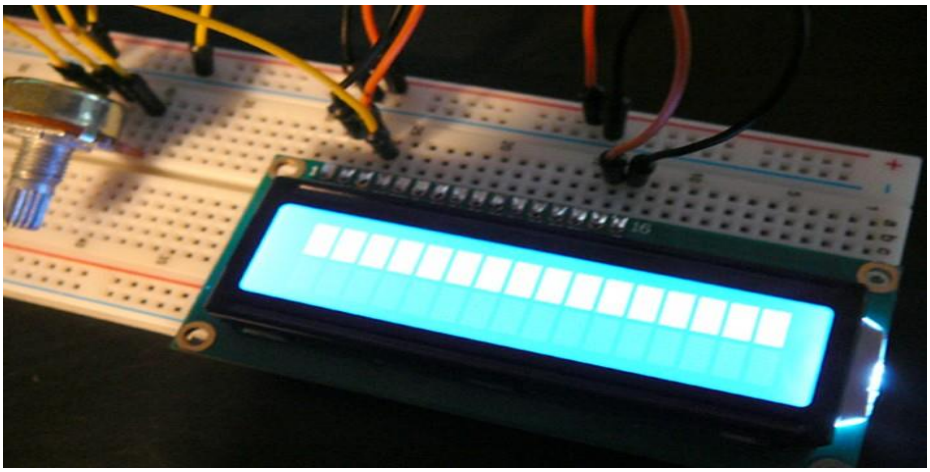


Рисунок 3.7 – Зовнішній вигляд LCD 1602 з відрегульованою контрастністю

2. Ознайомлення із основними принципами підключення LCD індикаторів до мікропроцесорів по інтерфейсу I2C

Недоліком наведеної вище схеми підключення рідкокристалічного індикатора до мікроконтролера (рис. 3.5) є необхідність задіяти мінімум 6 виводів контролера.

При необхідності відслідковувати дані з декількох датчиків і передавати керуючі сигналі на декілька виконуючих пристроїв, така схема накладає серйозні обмеження на використання плат Arduino UNO або NANO у реальних проєктах.

Але рідкокристалічний монітор із підтримкою інтерфейсу I²C (IIC, I2C, I²C) підключається до плати Arduino за допомогою лише чотирьох проводів – два для даних, два – для живлення.

I²C (IIC, англ. Inter-Integrated Circuit) – послідовна асиметрична шина для зв'язку між інтегральними схемами всередині електронних приладів. Використовує дві двонаправлені лінії зв'язку (SDA і SCL), застосовується для з'єднання низькошвидкісних периферійних компонентів з процесорами і мікроконтролерами (наприклад, на материнських платах, у вбудованих системах, в мобільних телефонах). Розроблена компанією Philips на початку 1980-х як проста шина внутрішнього зв'язку для створення керуючої електроніки.

I²C використовує дві двонаправлені лінії, підтягнуті до напруги живлення та керовані через відкритий колектор або відкритий стік – послідовна лінія даних (SDA, англ. Serial Data) і послідовна лінія тактування (SCL, англ. Serial Clock). Стандартні напруги +5 В або +3,3 В, проте допускаються й інші.

У зв'язку із ліцензійними причинами даний інтерфейс також може називатися двопровідним інтерфейсом Two Wire Interface (TWI), або Two Wire Serial Interface (TWSI).

Класична адресація включає 7-бітовий адресний простір з 16 зарезервованими адресами. Це означає, що є до 112 вільних адрес для підключення периферії на одну шину. Основний режим роботи – 100 кбіт/с; 10 кбіт/с в режимі роботи із зниженою швидкістю. Зауважимо, що стандарт допускає припинення тактування для роботи з повільними пристроями.

Після перегляду стандарту 1992 року стає можливим підключення ще більшої кількості пристроїв на одну шину (за рахунок можливості 10-бітної адресації), а також швидкість до 400 кбіт/с. Відповідно, доступна кількість вільних вузлів зросла до 1008. Максимальна допустима кількість мікросхем, приєднаних до однієї шині, обмежується максимальною ємністю шини в 400 пФ.

Найпростіша схема I²C може містити один провідний пристрій (найчастіше це мікроконтролер Arduino (ATMega 326) і кілька ведених (наприклад, дисплей LCD, різноманітні датчики та виконуючі пристрої). Кожен пристрій має адресу в діапазоні від 7 до 127. Двох пристроїв з однаковою адресою в одній схемі бути не повинно. Приклади мікропроцесорних систем з'єднаних за допомогою інтерфейсу I2C наведені на рис. 3.8.

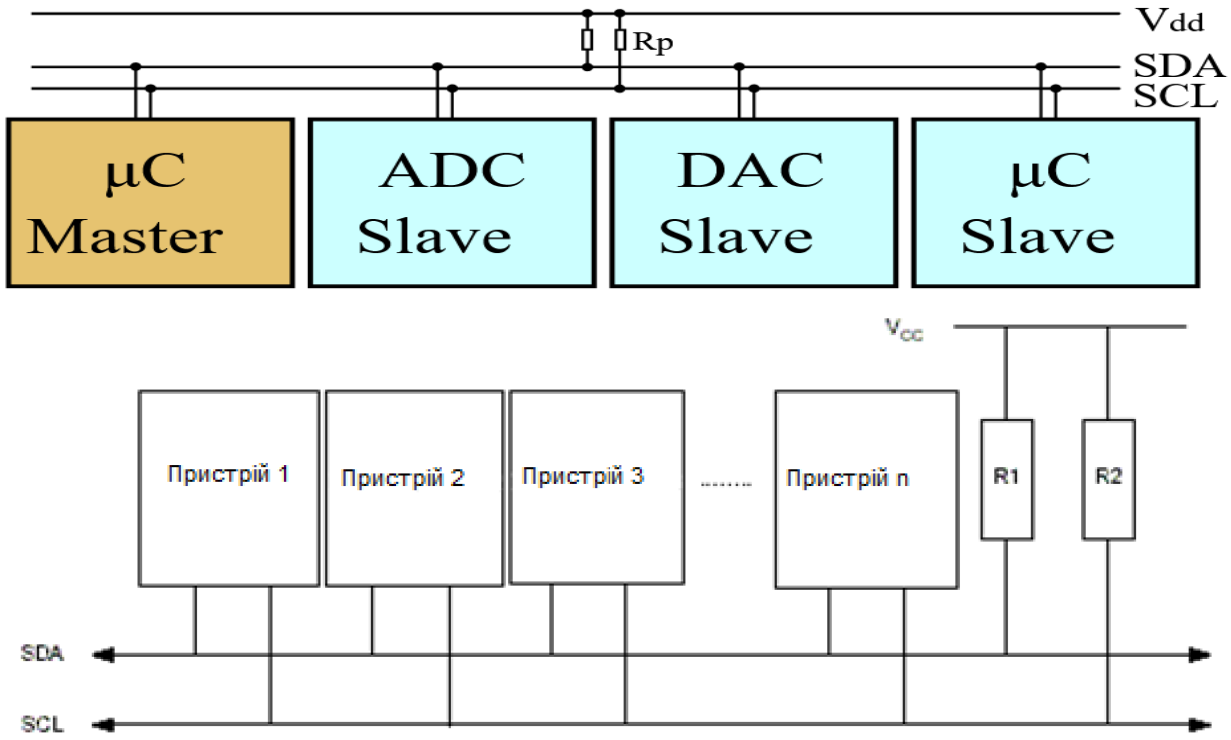
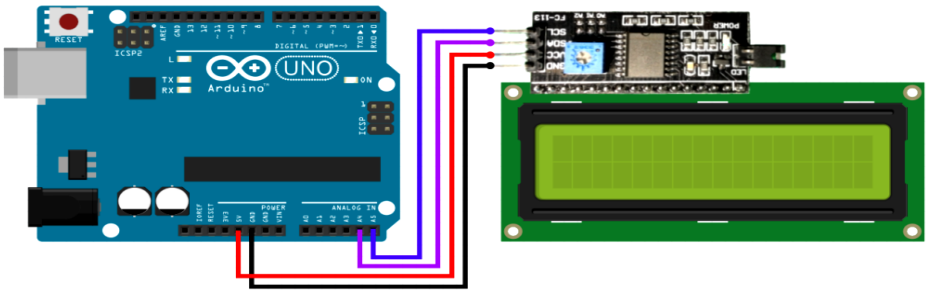
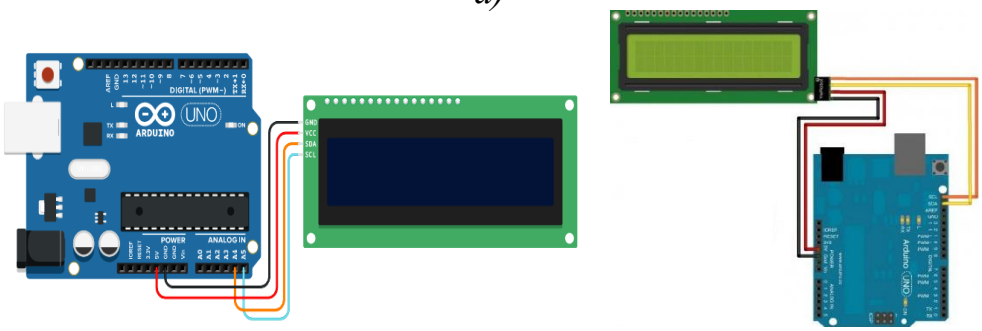


Рисунок 3.8 – Приклад мікропроцесорної системи з одним мікроконтролером (µC Master) і трьома підлеглими (slave) пристроями: пристрій 1 – ADC (англ. Analog-to-Digital Converter – аналого-цифровий перетворювач); пристрій 2 – DAC (англ. Digital-to-Analog Converter – цифро-аналоговий перетворювач); пристрій 3 – другий мікроконтролер µC Slave), та навантаженими (pull-up) резисторами Rp

Плата Arduino підтримує I²C на апаратному рівні. Для роботи потрібно всього дві лінії – SDA (лінія даних) та SCL (лінія синхронізації). Схеми підключення LCD 1602 до плати Arduino UNO наведені на рис. 3.9. Для виводу символів на екран використовуються бібліотеки Wire.h (є у стандартній Arduino IDE) та LiquidCrystal_I2C.h.



а)



б)

Рисунок 3.9 – Схема підключення до Arduino UNO індикатора LCD 1602 з окремим узгоджуючим пристроєм I^2C (*а*) та вбудованим інтерфейсом I^2C (*б*)

3. Ознайомлення з основними операторами бібліотеки LiquidCrystal

Бібліотека LiquidCrystal містить багато прикладів, що дозволяють розібратися в її роботі. Коротко ознайомимося з функціями цієї бібліотеки.

Функція begin()

Синтаксис функції:

lcd.begin(cols, rows)

Функція визначає розмірність індикатора (кількість символів в ширину і висоту). Параметри функції:

lcd – змінна типу LiquidCrystal;

cols – кількість символів в рядку;

rows – кількість рядків.

Функція clear()

Синтаксис функції:

lcd.clear()

Функція очищає екран рідкокристалічного індикатора і має в своєму розпорядженні курсор у верхньому лівому кутку.

Параметр:

lcd – змінна типу LiquidCrystal.

Функція home()

Синтаксис функції:

lcd.home()

Функція має курсор у верхньому лівому кутку рідкокристалічного індикатора. Використовується для визначення початкового положення при виведенні послідовного тексту на екран індикатора. Щоб ще й очистити екран індикатора, використовують замість цієї функції функцію clear().

Параметр:

lcd – змінна типу LiquidCrystal.

Функція setCursor()

Синтаксис функції:

lcd.setCursor(col, row)

Функція позиціонує курсор рідкокристалічного індикатора, тобто встановлює положення, в якому на його екран буде виведено текст.

Параметри:

lcd – змінна типу LiquidCrystal;

col – номер знакомісця у рядку (починаючи з 0 для першого знакомісця);

row – номер рядка (починаючи з 0 для першого рядка).

Функція write()

Синтаксис функції:

lcd.write(data)

Функція записує символ в рідкокристалічний індикатор.

Параметри:

lcd – змінна типу LiquidCrystal;

data – символ, що записується в індикатор.

Функція print()

Синтаксис функції print ():

led.print(data)

led.print(data, BASE)

Функція друкує текст на рідкокристалічному індикаторі. Параметри:

lcd – змінна типу LiquidCrystal;

data – дані для друку (тип char, byte, int, long або string);

BASE (опціонально) – система числення, в якій друкуються числа: BIN для двійкових (система числення 2), DEC для десяткових (система числення 10), OCT для вісімкових (система числення 8), HEX для шістнадцятькових (система числення 16).

Функція cursor()

Синтаксис функції:

lcd.cursor()

Функція виводить на екран рідкокристалічного індикатора курсор – підкреслення знакомісця в позиції, в яку буде записано символ.

Параметр:

lcd – змінна типу LiquidCrystal.

Функція noCursor()

Синтаксис функції:

lcd.noCursor()

Функція приховує курсор з екрану рідкокристалічного індикатора.

Параметр:

lcd – змінна типу LiquidCrystal.

Функція blink()

Синтаксис функції:

lcd.blink()

Функція виводить на екран рідкокристалічного індикатора миготливий курсор.

Якщо вона використовується в комбінації з функцією cursor(), результат буде залежати від конкретного індикатора.

Параметр:

lcd – змінна типу LiquidCrystal.

Функція noBlink()

Синтаксис функції:

lcd.noBlink ()

Функція вимикає миготливий курсор на екрані рідкокристалічного індикатора. Параметр:

lcd – змінна типу LiquidCrystal.

Функція display()

Синтаксис функції:

lcd.display()

Функція включає рідкокристалічний індикатор після того, як він був вимкнений функцією noDispiay(). Ця функція відновлює текст (і курсор), який був на дисплеї.

Параметр:

lcd – змінна типу LiquidCrystal.

Функція noDisplay()

Синтаксис функції:

lcd.noDisplay()

Функція вимикає рідкокристалічний індикатор без втрати інформації, яка відображається на ньому.

Параметр:

lcd – змінна типу LiquidCrystal.

Функція scrollDisplayLeft()

Синтаксис функції:

lcd.scrollDisplayLeft()

Функція прокручує інформацію на екрані індикатора (текст і курсор) на одне знакомісце ліворуч.

Параметр:

lcd – змінна типу LiquidCrystal.

Функція scrollDisplayRight()

Синтаксис функції:

led.scrollDisplayRight()

Функція прокручує інформацію на екрані індикатора (текст і курсор) на одне знакомісце праворуч.

Параметр:

led – змінна типу LiquidCrystal.

Функція autoscroll()

Синтаксис функції:

lcd.autoscroll()

Функція включає автоматичну прокрутку екрану рідкокристалічного індикатора і змушує при кожному виводі символу на екран індикатора переміщати попередні символи на одне знакомісце.

Якщо поточний напрямок виведення символів з лівого краю індикатора до правого краю (значення за замовчуванням) – екран індикатора прокручується вліво, якщо поточний напрямок виведення символів з правого краю індикатора до лівого – екран індикатора прокручується вправо. Це справляє на екрані рідкокристалічного індикатора ефект виведення кожного нового символу в одне й теж саме знакомісце.

Параметр:

led – змінна типу LiquidCrystal.

Функція noAutoscroll()

Синтаксис функції:

lcd.noAutoscroll()

Функція вимикає автоматичну прокрутку екрану рідкокристалічного індикатора.

Параметр:

led – змінна типу LiquidCrystal.

Функція leftToRight()

Синтаксис функції leftToRight():

lcd.leftToRight()

Функція встановлює напрямок виведення символів на екран рідкокристалічного індикатора зліва направо (значення за замовчуванням). Це означає, що наступні символи, що виводяться на екран індикатора, підуть зліва направо, але не вплинуть на виведений раніше текст.

Параметр:

led – змінна типу LiquidCrystal.

Функція rightToLeft()

Синтаксис функції:

lcd.rightToLeft()

Функція встановлює напрямок виведення символів на екран рідкокристалічного індикатора справа наліво (значення за замовчуванням – зліва направо). Це означає, що наступні символи, що виводяться на екран індикатора, підуть справа наліво, але не вплинуть на виведений раніше текст.

Параметр:

led – змінна типу LiquidCrystal.

Функція createChar()

Синтаксис функції:

led.createChar(num, data)

Функція створює користувальницький символ (гліф) для використання на рідкокристалічному дисплеї. Підтримуються до восьми символів 5x8 пікселів (нумерація з 0 до 7).

Створення кожного символу користувачем визначається масивом з восьми байтів – один байт для кожного рядка. П’ять молодших значущих бітів кожного байта визначають пікселі в цьому рядку. Щоб вивести призначений для користувача символ на екран, використовують функцію write() з номером символу в якості параметра.

Параметри:

lcd – змінна типу LiquidCrystal;

num – номер створюваного символу (від 0 до 7);

data – дані символьних пікселів.

4. Ознайомлення із процесом виводу цифр та символів на LCD індикатор за допомогою мікропроцесора ATMega328

Приведемо програму, що виводить стандартне привітання на LCD 1602 за допомогою мікропроцесора ATMega328, під'єднаних за схемою наведеною на рис. 3.5.

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 5, 6, 7, 8, 9); // до виводів RS, E, DB4, DB5, DB6, DB7
void setup()
{
 lcd.begin(16, 2);
 lcd.clear();
}
void loop()
{
 lcd.setCursor(5, 0);
 lcd.print("Hello");
 lcd.setCursor(6, 1);
 lcd.print("world!");
 delay(10000);
}
```

Результат роботи програми наведений на рис. 3.10.

Рисунок 3.10 – Результат роботи програми, що виводить стандартне привітання

Проведемо детальний розгляд роботи наведеної програми. Перш за все,

програма включає два початкових рядка. Їх мета – підключити бібліотеку для роботи з рідкокристалічними індикаторами (вона автоматично встановлюється в складі Arduino IDE) і повідомити їй, до яких контактів на платі Arduino підключений LCD. Цій меті служать наступні два рядки перед функцією void setup():

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(4, 5, 6, 7, 8, 9); // до виводів RS, E, DB4, DB5, DB6, DB7
```

Якщо потрібно використовувати інші контакти на платі Arduino, змінимо номери контактів у другому рядку.

Далі, в функції void setup(), бібліотеці повідомляється розмір екрану LCD в стовпчиках і рядках. Наприклад, в наведеній програмі ми повідомляємо, що екран LCD має два рядки по 16 символів в кожному:

```
lcd.begin(16, 2);
```

Після налаштування LCD в наступному рядку проводиться очищення екрану:

```
lcd.clear();
```

Потім курсор встановлюється в позицію початку виведення тексту викликом функції:

```
lcd.setCursor(x, y);
```

Тут *x* – це номер стовпця в рядку (від 0 до 15), а *y* – номер рядка (0 або 1). Після цього проводиться вивід тексту викликом функції lcd.print(), наприклад, щоб вивести слово “text”, треба виконати команду:

```
lcd.print ("text")
```

Відображення змінних і чисел на індикаторі LCD

Щоб вивести вміст змінної на індикаторі LCD, використовується виклик

функції:

lcd.print(variable);

При виведенні змінної типу float треба вказати кількість десяткових знаків для виведення. Наприклад, lcd.print (pi, 3) відобразить значення pi з трьома десятковими знаками, як показано на рис. 3.11:

```
float pi = 3.141592654;
lcd.print("pi: ");
lcd.print(pi, 3);
```

Рисунок 3.11 – Результат роботи програми, що виводить значення pi з трьома десятковими знаками

Вміст цілочисленої змінної можливо відобразити на екрані LCD не тільки в десятковій, але й у двійковій та шістнадцятковій системах, як показано на рис. 3.12 та у програмі:

```
int kk = 170;
lcd.setCursor(0, 0);
lcd.print("Binary:");
lcd.print(kk, BIN); // вивести число 170 у двійковій системі
lcd.setCursor(0, 1);
lcd.print("Hexadecimal:");
lcd.print(kk, HEX); // вивести число 170 у шістнадцятковій системі
```

Рисунок 3.12 – Результат роботи програми

5. Ознайомлення із процесом визначення та виводу власних символів на LCD індикатор за допомогою мікропроцесора ATMega328

Окрім стандартних літер, цифр та інших символів зі стандартного набору програмно можна визначати і власні символи. На екрані LCD 1602 кожен символ відображається в матриці, що містить вісім рядків по п’ять точок (пікселів). На рис. 3.13 ці матриці показані крупним планом.

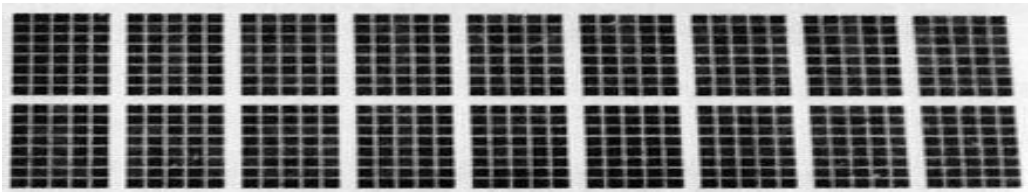


Рисунок 3.13 – Кожен символ LCD 1602 складається
з восьми рядків по п’ять пікселів

Щоб відобразити власний символ, його спочатку потрібно визначити у вигляді масиву. Наприклад, символ-смайлик визначається наступним чином:

```
byte a[8] =  {B00000,
              B01010,
              B01010,
              B00000,
              B00100,
              B10001,
              B01110,
              B00000};
```

Кожна цифра в цьому масиві відповідає пікселю: 0 – вимкненому, 1 – увімкненому. Елементи масиву представляють рядки пікселів на екрані; перший елемент відповідає верхньому рядку, наступний елемент – другому рядку і т. д.

Плануючи використання власних символів, спочатку треба намалювати їх на розлініяному папері (кожен зафарбований квадрат буде відповідати “1” в масиві, а кожен порожній квадрат – “0”).

В наведеному прикладі перший елемент масиву має значення B00000, тому всі пікселі в верхньому рядку будуть вимкнені. У другому рядку (йому відповідає елемент B01010) буде включений кожен другий піксель.

Далі зберігаємо масив, що визначає новий символ, в перший з восьми слотів, призначених для нестандартних символів, в функції void setup():

lcd.createChar(0, a); // зберегти масив a[8] в слот 0

Нарешті, щоб відобразити символ, додаємо наступний виклик в void loop():

lcd.write(byte (0));

Використовуємо наступний код для відображення нестандартних символів:

```
LiquidCrystal lcd (4, 5, 6, 7, 8, 9); // до виводів RS, E, DB4, DB5, DB6, DB7
byte a[8] = {B00000,
             B01010,
             B01010,
             B00000,
             B00100,
             B10001,
             B01110,
             B00000};
void setup ()
{
 lcd.begin(16, 2);
 lcd.createChar(0, a);
}
void loop()
{
 lcd.write(byte(0)); // вивести нестандартний символ із слота 0
// в наступну позицію курсора
}
```

На рис. 3.14 показані два рядки смайликів на екрані LCD.

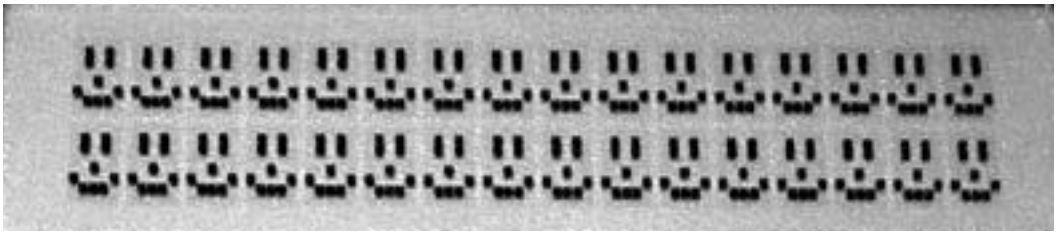


Рисунок 3.14 – Результат роботи програми, що виводить нестандартний символ

6. Індивідуальні завдання

1. Вивести найменування власної навчальної групи на перший рядок LCD 1602. Вивести власне прізвище та ініціали на другий рядок LCD 1602.

2. Вивести на екран індикатора значення e (математична константа, що є основою натуральних логарифмів, $e \approx 2{,}7182818284$) з кількістю десяткових знаків, що співпадає з останньою цифрою вашого варіанта.

3. Вивести на екран індикатора значення вашого варіанта по журналу у десятковій, двійковій та шістнадцятковій системі.

4. Вивести на екран індикатора ваше ім’я у рукописному вигляді (з використанням не більш ніж восьми нестандартних символів).

Порядок виконання роботи

1. Ознайомитися з основними принципами підключення LCD індикаторів до мікропроцесорів.

2. Ознайомитися з основними принципами підключення LCD індикаторів до мікропроцесора по інтерфейсу I^2C.

3. Ознайомитися з основними операторами бібліотеки LiquidCrystal.

4. Ознайомитися із процесом виводу цифр та символів на LCD індикатор за допомогою мікропроцесора ATMega328.

5. Ознайомитися із процесом визначення та виводу власних символів на LCD індикатор за допомогою мікропроцесора ATMega328.

6. Розробити моделі схем із підключенням LCD індикаторів до мікропроцесора по паралельному інтерфейсу та по інтерфейсу I^2C згідно індивідуального завдання у середовищі Tinkercad та перевірити їх роботу.

7. Розробити моделі схем із підключенням LCD індикаторів до мікропроцесора по паралельному інтерфейсу та по інтерфейсу I^2C згідно індивідуального завдання у середовищі PROTEUS та перевірити її роботу.

8. Зібрати схеми згідно індивідуального завдання, ввести програми в Arduino UNO R3, запустити та перевірити їх роботу.

9. Перевірити правильність функціонування програм у середовищах

моделювання та на реальній схемі.

10. Оформити звіт про роботу.

Зміст звіту

1. Тема лабораторної роботи.

2. Мета роботи.

3. Індивідуальне завдання.

4. Програма і спрощена блок-схема алгоритму організації процесу виведення найменування власної навчальної групи і власного прізвища та ініціалів, значення змінної із заданою кількістю десяткових знаків після коми, значення цілого числа (власного номера варіанта) у десятковій, двійковій та шістнадцятковій системі, власного імені з використанням розроблених нестандартних символів на LCD 1602 за допомогою мікропроцесора ATMega328 під'єднаного за паралельним інтерфейсом згідно індивідуального завдання.

5. Програма і спрощена блок-схема алгоритму організації процесу виведення найменування власної навчальної групи і власного прізвища та ініціалів, значення змінної із заданою кількістю десяткових знаків після коми, значення цілого числа (власного номера варіанта) у десятковій, двійковій та шістнадцятковій системі, власного імені з використанням розроблених нестандартних символів на LCD 1602 за допомогою мікропроцесора ATMega328 під'єднаного за послідовним інтерфейсом I2C згідно індивідуального завдання.

6. Працюючі моделі схем із підключенням LCD індикатора до мікропроцесора по паралельному інтерфейсу та по інтерфейсу I^2C згідно індивідуального завдання у середовищі Tinkercad.

7. Працюючі моделі схем із підключенням LCD індикаторів до мікропроцесора по паралельному інтерфейсу та по інтерфейсу I^2C згідно індивідуального завдання у середовищі PROTEUS.

8. Фото працюючих зібраних схем на монтажній платі згідно індивідуального завдання.

9. Висновки.

Практична робота 4

ДОСЛІДЖЕННЯ ОРГАНІЗАЦІЇ ЧАСОВИХ ФУНКЦІЙ КЕРУВАННЯ НА МІКРОПРОЦЕСОРІ ATMEGA328

Мета роботи: Одержання практичних навичок зчитування з кнопок керуючих сигналів, усунення брязкоту контактів та організації декількох одночасних часових функцій керування з використанням мікропроцесора ATMega328 на платі Arduino UNO R3.

Теми для попереднього проороблення:

– теоретичні відомості про функції часу в Arduino;

– теоретичні відомості про програмування в Arduino.

1. Ознайомлення із підключенням кнопки до мікропроцесора ATMega328

Кнопка (кнопковий перемикач) – найпростіший і найдоступніший з усіх видів датчиків (рис. 4.1). Натиснувши на неї, на контролер подається сигнал, який призводить до таких дій: включаються світлодіоди, відтворюються звуки, запускаються мотори.

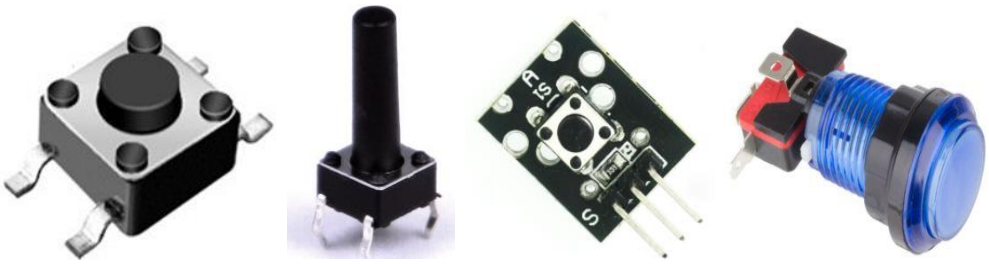


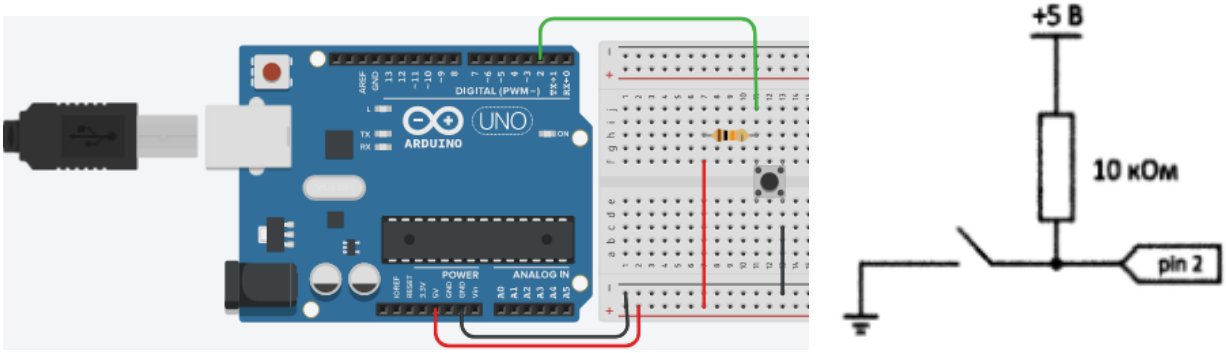
Рисунок 4.1 – Кнопки (кнопкові перемикачі) різних типів

Кнопка – це перемикач з двома парами контактів. Контакти в одній парі з’єднані між собою, тому більше одного вимикача в схемі реалізувати не вдасться, але можливо одночасно керувати двома паралельними сегментами.

Щоб підключити до Arduino нормально розімкнуту кнопку, можливо піти найпростішим шляхом: один вільний провідник кнопки з’єднати з живленням або “землею”, а інший – з цифровим виводом (піном) Arduino. Але це неправильно,

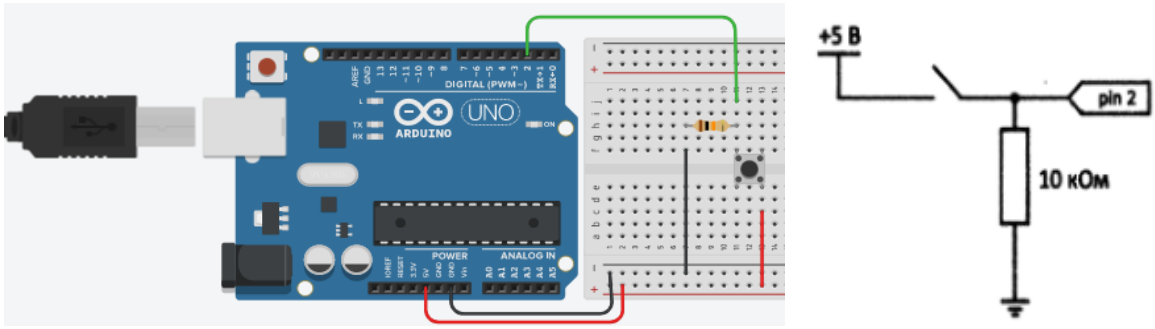
адже коли кнопка не замкнута, на цифровому виводі Arduino можуть з'являтися електромагнітні наведення і через це можливі помилкові випадкові спрацьовування.

Щоб уникнути наведень, цифровий вивід зазвичай підключають через досить великий резистор (5-10 кОм) або до живлення чи до "землі". У першому випадку це називається схемою з підтягуючим (pull-up) резистором (рис. 4.2), у другому – схемою зі стягуючим (pull-down) резистором (рис. 4.3). Резистори в обох схемах використовуються для установки "значення за замовчуванням" на вхідному контакті: в схемі з підтягуючим резистором це "1" (HIGH), в схемі зі стягуючим резистором – "0" (LOW).



а) *б)*

Рисунок 4.2 – Монтажна і принципова схеми підключення кнопки до Arduino (схема з підтягуючим резистором)



а) *б)*

Рисунок 4.3 – Монтажна і принципова схеми підключення кнопки до Arduino (схема зі стягуючим резистором)

Всі виводи всередині плати Arduino приєднані до шини живлення 5 В через резистори опором порядку 20-50 кОм. Ці резистори можна програмно підключати

до виводів або відключати від них. Програмне включення резисторів здійснюється так:

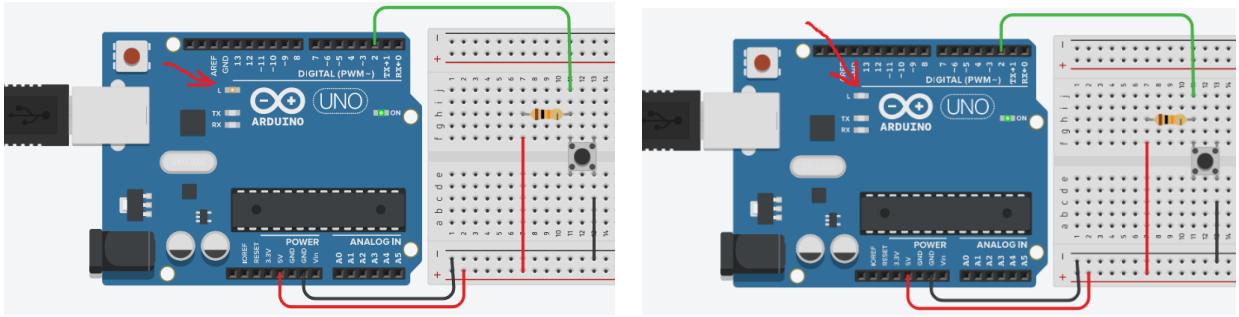
pinMode (2, INPUT_PULLUP); // внутрішній підтягуючий резистор 20 кОм підключений

Напишемо програму, яка встановлює стан світлодіода, підключеного до виводу 13 (знаходиться на платі Arduino), в залежності від стану кнопки (кнопка натиснута – світлодіод “горить”, кнопка не натиснута – світлодіод “не горить”).

```
const int LED = 13; // контакт 13 для підключення світлодіода
const int BUTTON = 2; // контакт 2 для підключення кнопки
boolean buttonState; // змінна статусу кнопки buttonState
void setup () {
// визначаємо вивід LED (світлодіод) як вихід
pinMode (LED, OUTPUT);
// визначаємо вивід BUTTON (кнопка) як вхід
pinMode (BUTTON, INPUT_PULLUP);
}
void loop () {
// зчитуємо стан BUTTON входу (кнопки) і записуємо в buttonState
buttonState = digitalRead (BUTTON);
// інверсія змінної buttonState для схеми з підтягуючим резистором
buttonState = ! buttonState;
// записуємо стан з buttonState на вихід LED (світлодіод)
digitalWrite (LED, buttonState);
}
```

Для схеми з підтягуючим резистором стан змінної buttonState інвертуємо, тому що у цьому випадку при натиснутій кнопці стан сигналу низький, а світлодіод світиться при високому. Для схем зі стягуючим резистором стан змінної buttonState інвертувати не треба.

Результатом виконання наведеної програми є світіння вбудованого в Arduino UNO світлодіоду (під’єднаний до піна 13) під час натиснення на кнопку (контакти замкнені) з підтягуючим резистором (підключений до +5 В), яку під’єднано до піна 2 (рис. 4.4, *а*). Коли на кнопку не натискають (контакти розімкнені) – вбудований світлодіод не світиться (рис. 4.4, *б*). В обох випадках вбудований світлодіод позначений червоною стрілкою.



а) *б)*

Рисунок 4.4 – Результат роботи вищенаведеної програми під час натискання кнопки (а) та коли на кнопку не натискають (б)

2. Ознайомлення з способами усунення брязкоту контактів кнопки

Розглянемо наступне завдання: будемо перемикати стан світлодіода (включений / виключений) при кожному натисканні кнопки.

Для цього завантажимо в Arduino наступну програму:

```
const int LED = 13; // контакт 13 для підключення світлодіода
const int BUTTON = 2; // контакт 2 для підключення кнопки
boolean buttonState; // змінна статусу кнопки buttonState
boolean buttonStatePrev = LOW; // змінна статусу кнопки попередня
boolean ledState = LOW; // змінна статусу світлодіода
void setup () {
// запуск послідовного порта
Serial.begin (9600);
// визначаємо вивід LED (світлодіод) як вихід
pinMode (LED, OUTPUT);
// визначаємо вивід BUTTON (кнопка) як вхід через підтягуючий резистор
pinMode (BUTTON, INPUT_PULLUP);
// початковий стан світлодіода
digitalWrite (LED, ledState);
}
void loop () {
// зчитуємо стан BUTTON входу (кнопки)
buttonState = digitalRead (BUTTON);
// якщо кнопка натиснута (зміна стану з LOW на HIGH)
if (buttonState == HIGH && buttonStatePrev = LOW) {
ledState = ! ledState;
// записуємо стан з ledState на вихід LED (світлодіод)
digitalWrite (LED, ledState);
// вивід значення стану ledState у послідовний порт для відслідковування
// його значення на комп’ютері за допомогою монітору послідовного порта
```

```
Serial.println (ledState);
}
buttonStatePrev = buttonState;
}
```

Натискаємо на кнопку і бачимо, що у послідовному порту при одноразовому натисканні кнопки відбувається декілька змін її стану (рис. 4.5), і відповідно, декілька перемикань світлодіода.

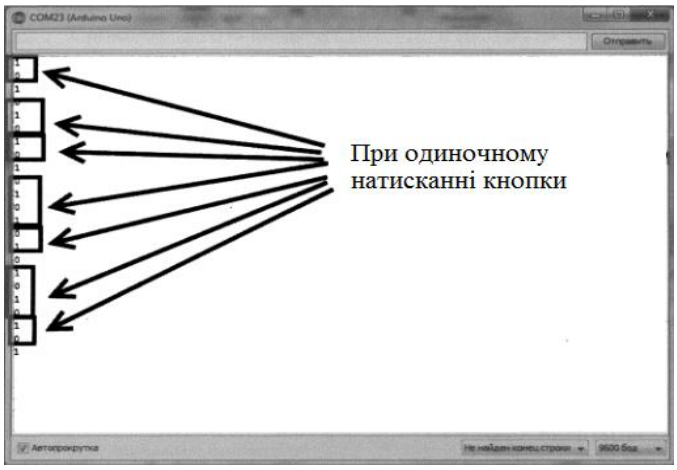
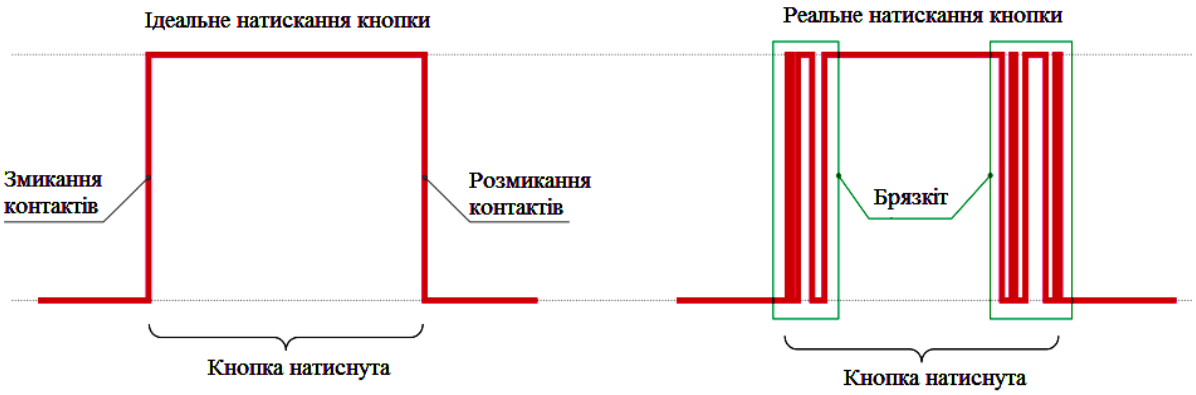


Рисунок 4.5 – Вивід даних відладки у монітор послідовного порта

Це обумовлено тим, що кнопки являють собою механічні пристрої з системою пружинного контакту, схильною до явища, званого брязкотом – в процесі натискання кнопки контакт, особливо на початку натискання, кілька разів замикається і розмикається. Тобто, коли ви натискаєте на кнопку, сигнал не просто змінюється від низького до високого – він протягом декількох мілісекунд декілька разів змінює значення, перш ніж установиться значення LOW. Графіки, наведені на рис. 4.6, ілюструють відмінність очікуваного явища від реального.

Натиснення кнопки відбувається приблизно десятки – сотні мілісекунд. Бажано відразу дізнатися про стан кнопки, прочитавши значення з входу контакту, як показано на рис. 4.6 (а). Однак, кнопка фактично рухається вгору-вниз, поки значення встановиться, як показано на рис. 4.6 (б). Перехідні процеси протікають дуже швидко, але, обробляючи сигнал від кнопки на Arduino, ми можемо зіткнутися з перехідними ефектами і повинні їх враховувати.



а) *б)*

Рисунок 4.6 – Ілюстрація явища брязкоту контактів кнопки

Алгоритм роботи програми, що усуває негативні наслідки явища брязкоту контактів, може бути наступним:

1) зберігаємо попередній стан кнопки і поточний стан кнопки (при ініціалізації – LOW);

2) зчитуємо поточний стан кнопки;

3) якщо поточний стан кнопки відрізняється від її попереднього стану, чекаємо визначений проміжок часу (наприклад, 5 мс), тому що кнопка, можливо, змінила стан;

4) після закінчення визначеного проміжку часу (наприклад, 5 мс) зчитуємо стан кнопки і використовуємо його у якості поточного;

5) якщо попередній стан кнопки був LOW, а поточний стан – HIGH, перемикаємо стан світлодіода;

6) встановлюємо попередній стан кнопки для її поточного стану;

7) повернення до п. 2 приведеного алгоритму (при необхідності знову зчитати поточний стан контактів кнопки).

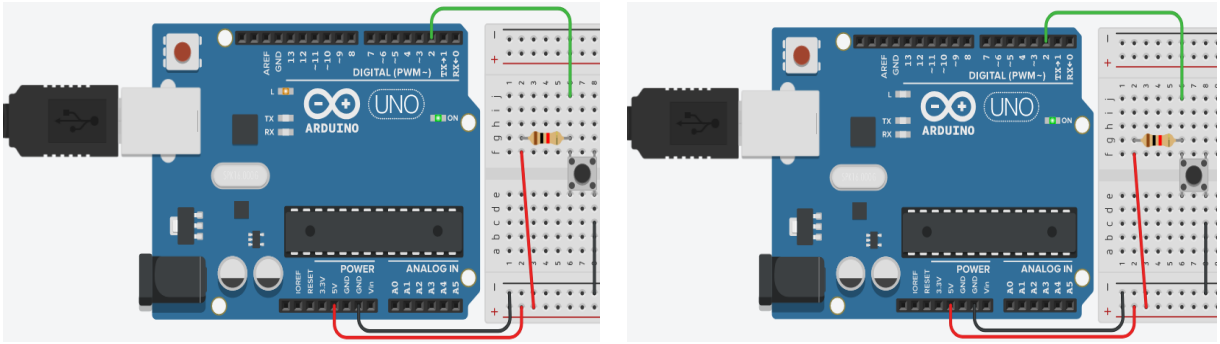
Програмна реалізація алгоритму:

```
const int LED = 13; // контакт 13 для підключення світлодіода
const int BUTTON = 2; // контакт 2 для підключення кнопки
boolean lastButton = LOW; // змінна для збереження попереднього стану кнопки
boolean currentButton = LOW; // змінна для збереження поточного стану кнопки
boolean ledOn = false; // поточний стан світлодіода (включений / виключений)
```

```
void setup () {
Serial.begin (9600); // запуск послідовного порту
pinMode (LED, OUTPUT); // конфігурувати контакт світлодіода як вихід
pinMode (BUTTON, INPUT); // конфігурувати контакт кнопки як вхід
}
void loop () {
currentButton = debounce (lastButton);
if (lastButton == LOW && currentButton == HIGH)
// якщо кнопку натиснули
{
ledOn = !ledOn; // інвертувати значення стану світлодіода та вивід значення
// стану ledOn у послідовний порт для відслідковування його значення
// на комп'ютері за допомогою монітору послідовного порта
Serial.println (ledOn);
}
lastButton = currentButton;
digitalWrite (LED, ledOn); // змінити статус стану світлодіода
}
// функція згладжування ефекту брязкоту контактів приймає в якості
// аргумента попередній стан кнопки, та видає її фактичний стан
boolean debounce (boolean last) {
boolean current = digitalRead (BUTTON); // зчитати стан кнопки
if (last != current) // якщо змінився
{
delay (5);                          // чекаємо 5 мс
current = digitalRead (BUTTON); // зчитуємо стан кнопки
return current;                     // повертаємо стан кнопки
}
}
```

При застосуванні наведеної програми одноразове натискання кнопки призводить до одноразової зміни стану світлодіода, ілюстрація роботи програми наведена на рис. 4.7. При застосуванні наведеної програми одноразове натискання кнопки призводить до одноразової зміни стану світлодіода. Також для боротьби з ефектом брязкоту контактів кнопки можливе застосування бібліотеки Bounce.

Усунення брязкоту контактів за допомогою затримок в програмі – спосіб дуже поширений і не вимагає зміни самої схеми. Але далеко не завжди його можна використовувати – програмний спосіб неможливо застосовувати при використанні у програмі переривань, тому що брязкіт призведе до багаторазового виклику функцій і вплинути на цей процес в програмі ми не зможемо.



а) *б)*

Рисунок 4.7 – Результат роботи вищенаведеної програми після непарного натискання кнопки (а) та після парного натискання кнопки (б)

Також використовують апаратні рішення усунення ефекту брязкання контактів, що згладжують імпульси, які надходять з кнопки.

Апаратний спосіб усунення брязкоту базується на використанні згладжуючих фільтрів. Згладжучий фільтр займається згладжуванням сплесків сигналів за рахунок додавання в схему елементів, які мають своєрідну “інерцію” по відношенню до таких електричних параметрів, як струм або напруга. Найпоширенішим прикладом таких “інерційних” електронних компонентів є конденсатор. Схема підключення фільтра з використанням конденсатору для усунення брязкоту наведена на рис. 4.8.

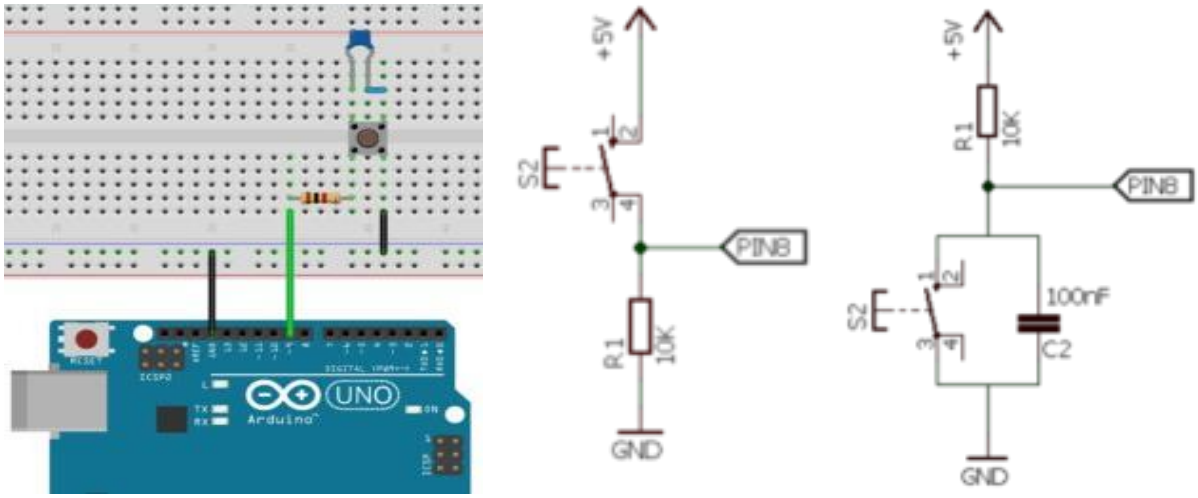


Рисунок 4.8 – Схема підключення фільтра з використанням конденсатора для усунення брязкоту

Для отримання прямокутної форми сигналу після згладжувального ланцюжка використовують тригер Шмітта.

3. Ознайомлення з основними функціями часу в Arduino

В Arduino існує кілька різних команд, які відповідають за роботу з часом та створення пауз (вони відрізняються за точністю і мають свої особливості, які варто враховувати при написанні коду):

– delay ();

– delayMicroseconds ();

– millis ();

– micros ().

Функцію delay () по суті вона є затримкою, яка призупиняє роботу програми, на вказане в дужках число мілісекунд. Максимальне значення може бути 4294967295 мс, що приблизно дорівнює 50 діб. Недоліком застосування цієї функції є неможливість її застосування при необхідності одночасної обробки сигналів від декількох датчиків та синхронної видачі управляючих сигналів на декілька виконуючих пристроїв, тому що на час її застосування призупиняється виконання всіх цих процесів. Під час виконання delay () не можливо опитувати входи, обробляти дані і змінювати стан виходів. Ця функція завантажує процесор на всі 100 %. Обійти це обмеження можна за допомогою переривань.

Функція delayMicroseconds () є повним аналогом delay () за винятком того, що одиниці вимірювання у неї не мілісекунди, а мікросекунди. Максимальне значення функції є 16383, що дорівнює 16 мс.

Функція micros () є повним аналогом millis () за винятком того, що одиниці вимірювання у неї не мілісекунди, а мікросекунди.

Функція millis() повертає кількість мілісекунд, що пройшли з моменту старту програми Arduino. Це число має формат unsigned long, який використовується для зберігання позитивних цілих чисел в діапазоні від 0 до 4294967295 ($2^{32} – 1$) та займає 32 біта (4 байта) в пам’яті.

При спробі виконання математичних операцій між значенням формату unsigned long та значеннями іншого типу (наприклад, int) буде згенерована помилка. Саме за допомогою цієї функції ми будемо організовувати опитування датчиків (кнопок) та видачу керуючих сигналів на управляючі пристрої (світлодіоди).

4. Ознайомлення з організацією часових функцій керування на мікропроцесорі ATMega328

Наведемо приклад заміни оператора delay () на millis (). Наприклад, delay (1000) що забезпечує затримку на 1 сек. (1000 мс), може бути замінений на такий фрагмент коду:

```
unsigned long timing; // змінна для зберігання точки відліку
void setup () {
  Serial.begin (9600);
}
void loop () {
// у цьому місці починається виконання аналога delay ()
// обчислюємо різницю між поточним моментом і раніше збереженої точкою
// відліку, якщо різниця більше потрібного значення, то виконуємо код,
// якщо немає – нічого не робимо
  if (millis () – timing> 1000) {
// замість 1000 пишемо потрібне значення паузи
   timing = millis ();
   Serial.println ("1 second");
  }
}
```

Приклад організації блимання світлодіодом за допомогою функції millis():

```
const int ledPin = 13; // номер піна з підключеним світлодіодом
int ledState = LOW; // стан світлодіода: вмикання/вимикання
long previousMillis = 0; // час, коли стан світлодіода оновлювався
long interval = 1000; // половина періоду миготіння (в мс)
void setup () {
  // встановлюємо цифровий пін із світлодіодом в режим виведення
  pinMode (ledPin, OUTPUT);
}
void loop () {
  // з’ясовуємо, чи не настав момент змінити стан світлодіода
```

```
  unsigned long currentMillis = millis (); // поточний час в мілісекундах
  if (currentMillis – previousMillis> interval) {
    // зберігаємо останній момент, коли змінився стан світлодіода
    previousMillis = currentMillis;
    // змінюємо стан світлодіода на протилежний
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;
    // встановлюємо високий або низький рівень напруги на виході
    // грунтуючись на значенні змінної ledState
    digitalWrite (ledPin, ledState);
  }
}
```

Приклад організації блимання світлодіодом з різним часом ввімкнення та вимкнення за допомогою функції millis():

```
// ці змінні зберігають часовий шаблон для інтервалів миготіння
// та поточний стан світлодіода
int ledPin = 13; // номер піна з світлодіодом
int ledState = LOW; // стан світлодіода
// останній момент часу, коли стан світлодіода змінювався
unsigned long previousMillis = 0;
long OnTime = 250; // тривалість світіння світлодіода (в мілісекундах)
long OffTime = 750; // світлодіод не горить (в мілісекундах)
void setup () {
  // встановлюємо цифровий пін з світлодіодом як “вихід”
  pinMode (ledPin, OUTPUT);
}
void loop () {
  // з’ясовуємо, чи не настав момент змінити стан світлодіода
 unsigned long currentMillis = millis (); // поточний час в мілісекундах
  // якщо світлодіод включений і світиться більше ніж треба
  if ((ledState == HIGH) && (currentMillis – previousMillis> = OnTime))
  {
    ledState = LOW; // вимикаємо
    previousMillis = currentMillis; // запам’ятовуємо момент часу
    digitalWrite (ledPin, ledState); // реалізуємо новий стан
  }
  else if ((ledState == LOW) && (currentMillis – previousMillis> = OffTime))
  {
    ledState = HIGH; // вимикаємо
    previousMillis = currentMillis; // запам’ятовуємо момент часу
```

```
    digitalWrite (ledPin, ledState); // реалізуємо новий стан
  }
}
```

Приклад організації блимання двома світлодіодами з різним часом ввімкнення та вимкнення за допомогою функції millis():

```
// ці змінні зберігають часовий шаблон для інтервалів миготіння
// та поточний стан світлодіодів
int ledPin1 = 12; // номер піна з світлодіодом
int ledState1 = LOW; // стан світлодіода
// останній момент часу, коли стан світлодіода змінювався
unsigned long previousMillis1 = 0;
long OnTime1 = 250; // тривалість світіння світлодіода (в мілісекундах)
long OffTime1 = 750; // світлодіод не горить (в мілісекундах)
int ledPin2 = 13; // номер піна з світлодіодом
int ledState2 = LOW; // стан світлодіода
// останній момент часу, коли стан світлодіода змінювався
unsigned long previousMillis2 = 0;
long OnTime2 = 330; // тривалість світіння світлодіода (в мілісекундах)
long OffTime2 = 400; // світлодіод не горить (в мілісекундах)
void setup () {
// встановлюємо цифрові піни із світлодіодами як “вихід”
pinMode (ledPin1, OUTPUT);
pinMode (ledPin2, OUTPUT);
}
void loop () {
// з’ясовуємо, чи не настав момент змінити стан світлодіода
unsigned long currentMillis = millis (); // поточний час в мілісекундах
// програма управління для першого світлодіода
if ((ledState1 == HIGH) && (currentMillis – previousMillis1> = OnTime1))
  {
   ledState1 = LOW; // вимикаємо
   previousMillis1 = currentMillis; // запам’ятовуємо момент часу
   digitalWrite (ledPin1, ledState1); // реалізуємо новий стан
  }
else if ((ledState1 == LOW) && (currentMillis – previousMillis1> = OffTime1))
  {
   ledState1 = HIGH; // вимикаємо
   previousMillis1 = currentMillis; // запам’ятовуємо момент часу
   digitalWrite (ledPin1, ledState1); // реалізуємо новий стан
  }
 // програма управління для другого світлодіода
if ((ledState2 == HIGH) && (currentMillis – previousMillis2> = OnTime2))
```

```
  {
   ledState2 = LOW; // вимикаємо
   previousMillis2 = currentMillis; // запам’ятовуємо момент часу
   digitalWrite (ledPin2, ledState2); // реалізуємо новий стан
  }
 else if ((ledState2 == LOW) && (currentMillis – previousMillis2> = OffTime2))
  {
   ledState2 = HIGH; // вимикаємо
   previousMillis2 = currentMillis; // запам’ятовуємо момент часу
   digitalWrite (ledPin2, ledState2); // реалізуємо новий стан
  }
}
```

Мова програмування Arduino є різновидом C++, яка підтримує об’єктно-орієнтоване програмування. Використавши об’єктно-орієнтовані властивості мови, ми можемо зібрати разом всі змінні стану і функціональність для миготливого світлодіода в клас C++. Це зовсім не складно зробити. Адже весь код вже є. Нам просто потрібно перепакувати його в клас. Починаємо з оголошення класу Flasher:

```
class Flasher
{
  // змінні – члени класу ініціалізувалися під час запуску
  int ledPin; // номер піна з світлодіодом
  long OnTime; // час включення в мілісекундах
  long OffTime; // час, коли світлодіод вимкнений
   // поточний стан
  int ledState; // стан вмикання/вимикання
  unsigned long previousMillis; // останній момент зміни стану
};
```

Далі додаємо конструктор класу. Конструктор має те ж саме ім’я, що й клас та призначений для ініціалізації змінних.

```
// конструктор створює екземпляр класу <em> Flasher </ em>
// та ініціалізує змінні – члени класу і стану
public:
Flasher (int pin, long on, long off)
{
  ledPin = pin;
  pinMode (ledPin, OUTPUT);
```

```
    OnTime = on;
   OffTime = off;
   ledState = LOW;
   previousMillis = 0;
}
```

Візьмемо наш цикл і перетворимо його в функцію – член класу, яка називається Update (). Вона ідентична оригінальному void loop () – змінено тільки назву.

```
void Update ()
{
  // з'ясовуємо, чи не настав момент змінити стан світлодіода
unsigned long currentMillis = millis (); // поточний час в мілісекундах
if ((ledState == HIGH) && (currentMillis – previousMillis> = OnTime))
{
   ledState = LOW; // вимикаємо
   previousMillis = currentMillis; // запам'ятовуємо момент часу
   digitalWrite (ledPin, ledState); // реалізуємо новий стан
}
else if ((ledState == LOW) && (currentMillis – previousMillis> = OffTime))
{
   ledState = HIGH; // вимикаємо
   previousMillis = currentMillis; // запам'ятовуємо момент часу
   digitalWrite (ledPin, ledState); // реалізуємо новий стан
}
}
```

Повторно скомпонувавши існуючий код в клас Flasher, ми інкапсулювали всі змінні (стан) та отримали функціональність для миготіння світлодіодом. Тепер, для блимання кожного з світлодіодів, ми створюємо екземпляр класу Flasher викликом конструктора і на кожному кроці циклу викликаємо Update () для кожного екземпляра Flasher.

Приклад програми, що реалізує незалежне миготіння трьох світлодіодів:

```
class Flasher
{
 // змінні – члени класу ініціалізуються при запуску
 int ledPin; // номер піна із світлодіодом
 long OnTime; // час включення в мілісекундах
 long OffTime; // час, коли світлодіод вимкнений
 // поточний стан
```

```
 int ledState; // стан вмикання-вимикання
 unsigned long previousMillis; // останній момент зміни стану
// конструктор створює екземпляр Flasher
// та ініціалізує змінні – члени класу і стан
 public:
 Flasher (int pin, long on, long off)
 {
  ledPin = pin;
  pinMode (ledPin, OUTPUT);
  OnTime = on;
  OffTime = off;
  ledState = LOW;
  previousMillis = 0;
 }
 void Update ()
 {
  // з'ясовуємо, чи не настав момент змінити стан світлодіода
  unsigned long currentMillis = millis (); // поточний час в мілісекундах
  if ((ledState == HIGH) && (currentMillis – previousMillis> = OnTime))
  {
    ledState = LOW; // вимикаємо
    previousMillis = currentMillis; // запам'ятовуємо момент часу
    digitalWrite (ledPin, ledState); // реалізуємо новий стан
  }
  else if ((ledState == LOW) && (currentMillis – previousMillis> = OffTime))
  {
    ledState = HIGH; // вимикаємо
    previousMillis = currentMillis; // запам'ятовуємо момент часу
    digitalWrite (ledPin, ledState); // реалізуємо новий стан
  }
 }
};
Flasher led1 (11, 100, 400);
Flasher led2 (12, 350, 350);
Flasher led3 (13, 300, 700);
void setup ()
{
}
void loop ()
{
 led1.Update ();
 led2.Update ();
 led3.Update ();
}
```

Таким чином, кожен додатковий світлодіод вимагає лише два рядки коду.

5. Індивідуальне завдання

Забезпечити почергове блимання світлодіодів, підключених до заданих роз'ємів (пінів) при кожному натисканні на управляючі кнопки, які підключені до заданих пінів. Варіанти завдань наведені у табл. 4.1.

Таблиця 4.1 – Варіанти завдань

№ вар.	Світлодіод 1			Світлодіод 2		
	N пінів управляючої кнопки	N пінів підключення світлодіоду	Час включення/ виключення світлодіоду, с	N пінів управляючої кнопки	N пінів підключення світлодіоду	Час включення/ виключення світлодіоду, с
1	2	3	0,5/1	4	5	1/2
2	3	4	0,2/1,2	5	6	1,8/2,5
3	4	5	0,7/1,5	6	7	1,5/0,8
4	5	6	2/1	7	8	1,5/2,5
5	6	7	0,7/1,8	8	9	1,3/2,1
6	7	8	0,4/0,9	9	10	1/2
7	8	9	0,8/1,5	7	6	2/1
8	9	10	0,5/1,1	6	5	1,5/2,1
9	2	11	0,6/1,3	5	4	1,8/1
10	3	12	1,4/0,8	4	2	1,5/3,2

Зміна режиму роботи світлодіодів відбувається при кожному натисканні відповідної управляючої кнопки. Забезпечити захист створеного проєкту від ефекту брязкоту контактів.

Порядок виконання роботи

1. Ознайомитися з підключенням кнопки до мікропроцесора ATMega328.

2. Ознайомитися з способами усунення брязкоту контактів.

3. Ознайомитися з організацією часових функцій керування на мікропроцесорі ATMega328.

4. Розробити програму почергового блимання світлодіодів, підключених до заданих портів при кожному натисканні на управляючої кнопки, що підключені до заданих пінів, згідно свого варіанту.

5. Розробити модель схеми згідно індивідуального завдання у середовищі Tinkercad та перевірити її роботу.

6. Розробити модель схеми згідно індивідуального завдання у середовищі PROTEUS та перевірити її роботу.

7. Зібрати схему на монтажній платі згідно індивідуального завдання, ввести програми в Arduino UNO R3, запустити та перевірити її роботу.

8. Перевірити правильність функціонування програм у середовищах моделювання та на реальній схемі.

9. Оформити звіт про роботу.

Зміст звіту

1. Тема лабораторної роботи.

2. Мета роботи.

3. Індивідуальне завдання.

4. Програма і спрощена блок-схема алгоритму організації процесу захисту від брязкання контактів.

5. Програма і спрощена блок-схема алгоритму організації процесу почергового включення/виключення світлодіодів, підключених до заданих пінів при кожному натисканні на управляючі кнопки, підключені до заданих портів.

6. Працюючі моделі схем згідно індивідуального завдання у середовищі Tinkercad.

7. Працюючі моделі схем згідно індивідуального завдання у середовищі PROTEUS.

8. Фото працюючої зібраної схеми на монтажній платі згідно індивідуального завдання.

9. Висновки.

Практична робота 5

ДОСЛІДЖЕННЯ ПРИНЦИПІВ ВИКОРИСТАННЯ ШИРОТНО-ІМПУЛЬСНИХ МОДУЛЯТОРІВ НА МІКРОПРОЦЕСОРІ ATMEGA328

Мета роботи: Ознайомитись із принципами використання широтно-імпульсних модуляторів та особливістю побудови і різновидами серводвигунів, дослідити роботу сервоприводу, що керується мікропроцесором ATMega328 на платформі Arduino. Отримати практичні навички управління сервоприводів мікропроцесором ATMega328 на платі Arduino UNO R3.

Теми для попереднього пророблення:

– теоретичні відомості про підключення кнопки до мікропроцесора ATMega328;

– теоретичні відомості про підключення потенціометра до мікропроцесора ATMega328;

– теоретичні відомості про програмування в Arduino.

1. Широтно-імпульсна модуляція в Arduino

Широтно-імпульсна модуляція (ШІМ, або PWM, pulse-width modulation) поділяється на два види: аналогова та цифрова. Кожен із видів має свої переваги і схемотехнічно може реалізовуватися різними способами.

Принцип дії аналогового ШІ-модулятора ґрунтується на порівнянні двох сигналів, частота яких відрізняється на кілька порядків. Елементом порівняння виступає операційний підсилювач (компаратор). На один з його входів подають пилкоподібну напругу високої постійної частоти, а на інший – низькочастотна модулююча напруга зі змінною амплітудою. Компаратор порівнює обидва значення і на виході формує прямокутні імпульси, тривалість яких визначається поточним значенням сигналу, що модулює.

ШІМ це ще й спосіб управління потужністю на навантаженні за допомогою зміни скважності імпульсів при постійній амплітуді та частоті імпульсів.

Можна виділити дві основні сфери застосування ШІМ:

1) у вторинних джерелах живлення, різних регуляторах потужності,

регуляторах яскравості джерел світла, швидкості обертання колекторних двигунів тощо. У цих випадках застосування ШІМ дозволяє значно збільшити коефіцієнт корисної дії системи та спростити її реалізацію;

2) для отримання аналогового сигналу за допомогою цифрового виходу мікроконтролера. Своєрідний цифро-аналоговий перетворювач (ЦАП), який простий у реалізації і вимагає мінімум зовнішніх компонентів (часто достатньо одного RC-ланцюжка).

Принцип регулювання за допомогою ШІМ – зміна ширини імпульсів за постійної амплітуди та частоти сигналу (рис. 5.1).

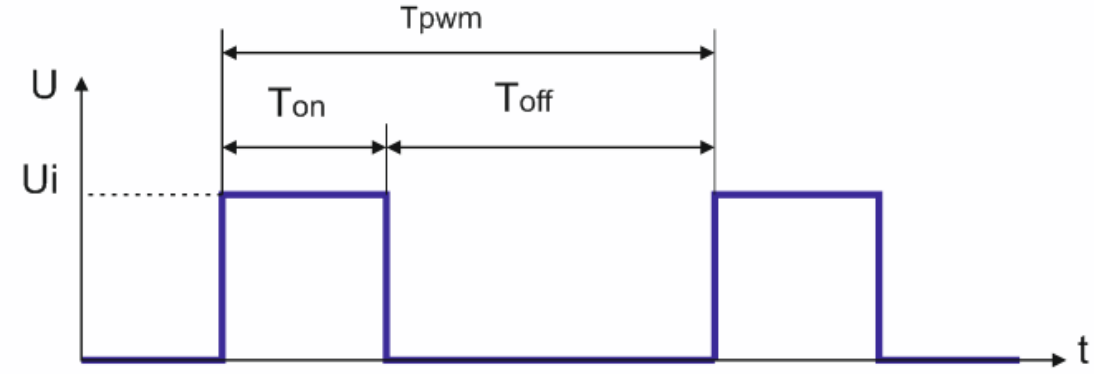


Рисунок 5.1 – Структура ШІМ сигналу

Основними параметрами ШІМ сигналу є:

U_i – амплітуда імпульсів;

T_{on} – час високого (увімкненого) стану сигналу;

T_{off} – час низького (відключеного) стану сигналу;

T_{pwm} – час періоду ШІМ.

Потужність навантаження пропорційна співвідношенню часу включеного і відключеного стану сигналу.

Це співвідношення визначає коефіцієнт заповнення ШІМ:

$$K_w = T_{on}/T_{pwm}.$$

Він показує, яку частину періоду сигнал перебуває в увімкненому стані. Може змінюватися у межах від 0 (сигнал завжди вимкнено) до 1 (сигнал постійно перебуває у включеному стані).

Найчастіше використовують відсотковий коефіцієнт заповнення, який знаходиться в межах від 0 до 100 % (рис. 5.2).

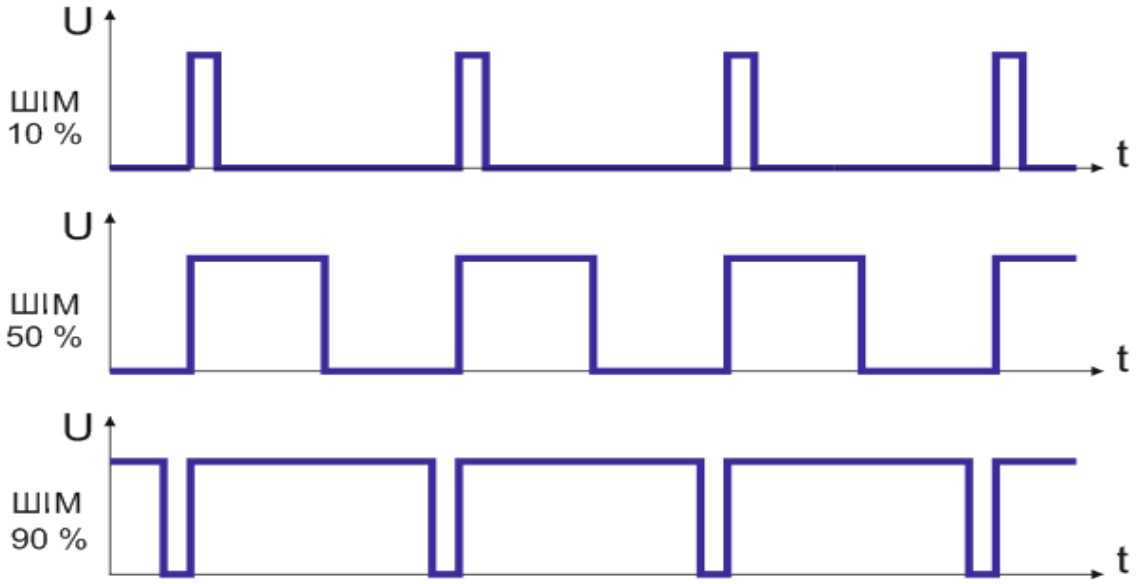


Рисунок 5.2 – Ілюстрація відсоткового коефіцієнту заповнення

Середнє значення електричної потужності на навантаженні суворо пропорційне коефіцієнту заповнення. Коли кажуть, що ШІМ дорівнює, наприклад, 10 %, мають на увазі саме коефіцієнт заповнення.

Активне використання контролерів на основі ШІМ обумовлено їх незаперечними перевагами:

– висока ефективність перетворення сигналу;

– стабільність роботи;

– економія енергії, що споживається навантаженням;

– низька вартість;

– висока надійність всього пристрою.

Плати Arduino на базі мікроконтролерів ATMega168/328 мають 6 апаратних ШІ-модуляторів. Сигнали ШІМ можуть бути згенеровані на виводах 3, 5, 6, 9, 10, 11.

Управління апаратними ШІМ здійснюється за допомогою системної функції analogWrite().

void analogWrite(pin, val)

Функція переводить вивід у режим ШІМ і задає йому коефіцієнт заповнення. Перед використанням analogWrite() функцію pinMode() для встановлення виведення в режим “вивід” викликати необов’язково.

Аргументи оператора analogWrite():

pin – номер виводу для генерації ШІМ сигналу;

val – коефіцієнт заповнення ШІМ. Без додаткових установок діапазон val від 0 до 255 відповідає коефіцієнту заповнення від 0 до 100 %.

Тобто необхідно враховувати, що розрядність системних ШІМ Arduino – 8 розрядів.

Приклад:

```
analogWrite(9, 25); // на виводі 9 ШІМ = 10 %
```

Для генерації ШІМ задіяно всі три таймери Arduino (табл. 5.1).

Таблиця 5.1 – Таймери Arduino, що використовуються для ШІМ

Таймер	Використовується для генерації ШІМ на виводах
Таймер 0	виводи 5 і 6
Таймер 1	виводи 9 і 10
Таймер 2	виводи 3 і 11

За замовчуванням система Arduino встановлює на всіх виводах ШІМ параметри:

– частота 976,56 Гц для Timer 0;

– частота 488,28 Гц для Timer 1 та Timer 2;

– роздільна здатність 8 розрядів (0…255).

Якщо таймер використовується для інших цілей, наприклад для управління перериванням, параметри ШІМ відповідних виводів можуть не відповідати наведеним вище значенням. Тому, при використанні деяких бібліотек, наприклад MsTimer2, TimerOne або подібних до них деякі виводи у якості генераторів ШІМ сигналів використовувати не можна.

2. Ознайомлення із поняттям сервопривод

Структура серводвигуна є досить складною (рис. 5.3). У сервоприводі є внутрішній механізм зворотного зв’язку, що дозволяє контролеру точно визначати позицію сервоприводу. Цей механізм використовується для забезпечення точного руху сервоприводу на задану позицію.

У верхній частині приладу розміщується шестерінчастий редуктор, який дозволяє значно збільшити крутний момент двигуна постійного струму за рахунок зниження швидкості його обертання. Нижче розташований потенціометр, завдання якого – визначати, на який кут повернути вихідний вал редуктора. Нарешті, в глибині корпусу знаходиться невелика плата управління, яка і робить серводвигун розумним. Ця плата постійно відстежує поточний стан вала і коригує його у разі, якщо вал намагається піти із заданої позиції.

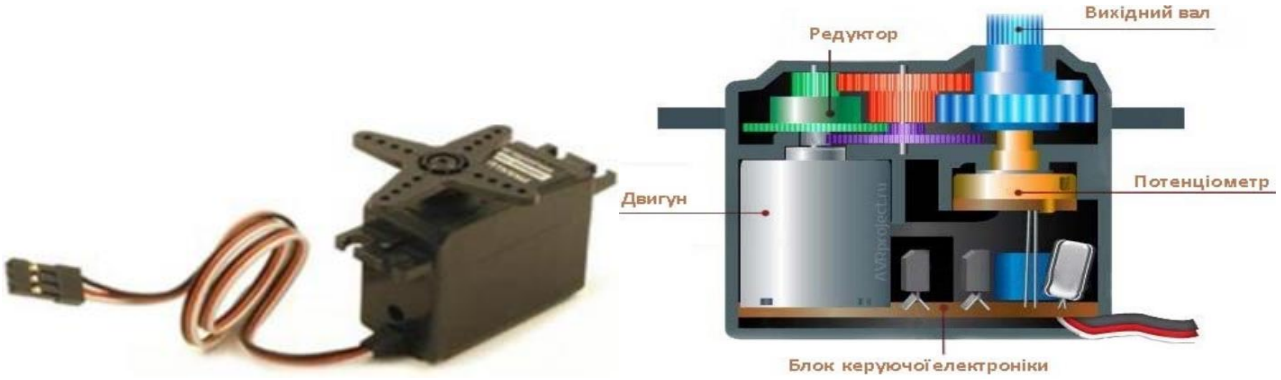


Рисунок 5.3 – Зовнішній вигляд та внутрішня структура серводвигуна

3. Створення проєкту управління сервоприводами в середовищі Proteus

Спроектуємо сервопривід у Proteus та розробимо керуючий привід постійного струму з використанням логічних елементів. Почнемо зі створення простого двигуна постійного струму, який вже є у Proteus та який легко використовувати. Для керування двигуном ми будемо застосовувати прямий метод, тобто подачу напруги на обидві його сторони. Цей тип двигуна потребує різної полярності на двох його кінцях. Якщо полярність підключення є прямою, то двигун постійного струму рухається в одному напрямку, і зміна полярності спричинить його рух у протилежному напрямку.

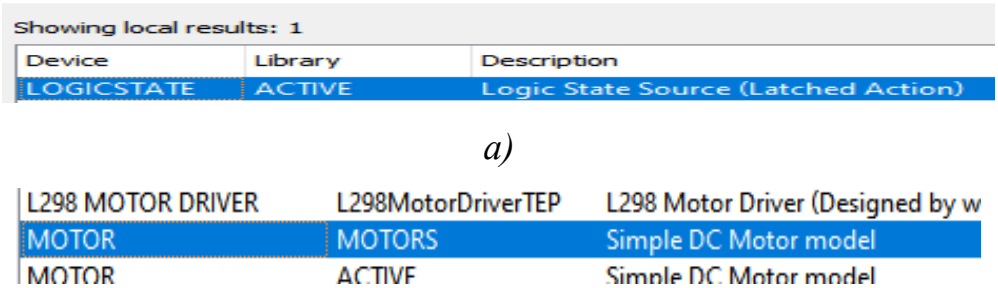
Для реалізації цього методу:

– створимо новий проєкт у середовищі Proteus;

– додаємо два елементи LOGICSTATE (рис. 5.4, а);

– додаємо серводвигун MOTOR (рис. 5.4, б);

– з’єднаємо ці елементи так, як показано на рис. 5.5.



а)

б)

Рисунок 5.4 – Додавання елементів

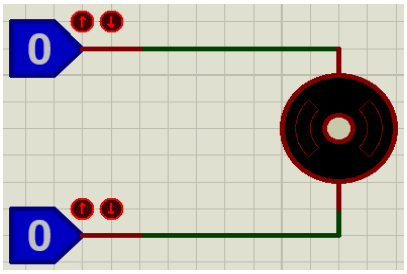


Рисунок 5.5 – З’єднання елементів

LOGICSTATE має два стани: “1” і “0”. Коли встановлено значення “0” це означає 0 В, а коли значення “1” – це 5 В. Щоб змінити стан, треба натиснути на необхідний Logic State.

Напрямок двигуна буде залежати від логіки станів Logic State. Таким чином, їх буде всього чотири стани:

1. Коли обидва стани “0”, двигун не буде рухатися і залишатися нерухомим.

2. Коли обидва стани “1”, двигун не буде рухатися і залишатися нерухомим.

3. Двигун буде рухатися за годинниковою стрілкою, коли верхній стан “1”, а нижній “0”.

4. Двигун буде рухатися проти годинникової стрілки, коли верхній стан “0”, а нижній “1”.

Керування сервоприводами за допомогою ШІМ в середовищі Proteus

ШІМ – це техніка керування аналоговими сигналами з використанням цифрових сигналів. ШІМ зазвичай використовується для керування сервоприводами, які забезпечують точне позиціонування об’єктів, наприклад, руху робота. На відміну від звичайного двигуна постійного струму, тут рівень ШІМ задає не швидкість обертання, а кут повороту.

Контролер сервоприводу зазвичай працює за принципом PWM (pulse-width modulation, ШІМ – широтна імпульсна модуляція). Це означає, що сервопривод приймає вхідний сигнал у вигляді імпульсів, і його позиція залежить від ширини цих імпульсів (відносної тривалості імпульсу за час його періоду). Для регулювання позиції сервоприводу зазвичай використовують сигнали з шириною імпульсу від 1 мс до 2 мс. Ширина імпульсу відповідає кутовій позиції сервоприводу, де 1 мс відповідає повному обертанню в одному напрямку, 1,5 мс – нейтральній позиції, а 2 мс – повному обертанню в іншому напрямку (рис. 5.6).

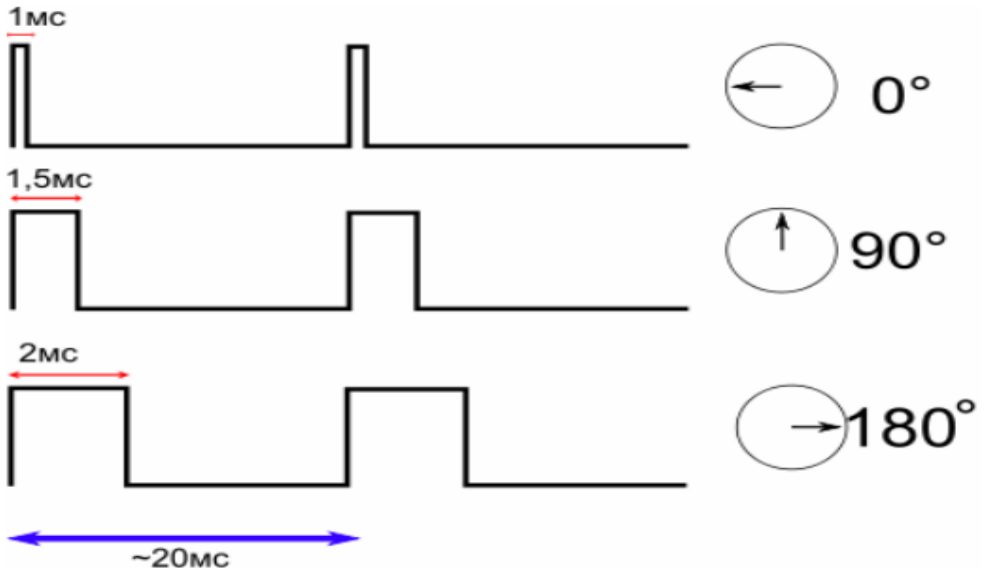


Рисунок 5.6 – Сигнали керування

Створимо новий проєкт у середовищі Proteus та додаємо три елементи MOTOR-PWMSERVO. Потім під’єднаємо до серводвигунів живлення та, натиснувши двічі на елемент POWER, задаємо йому значення +12 В. Також додаємо до зібраної схеми GROUND. З’єднаємо все, як показано на рис. 5.7.

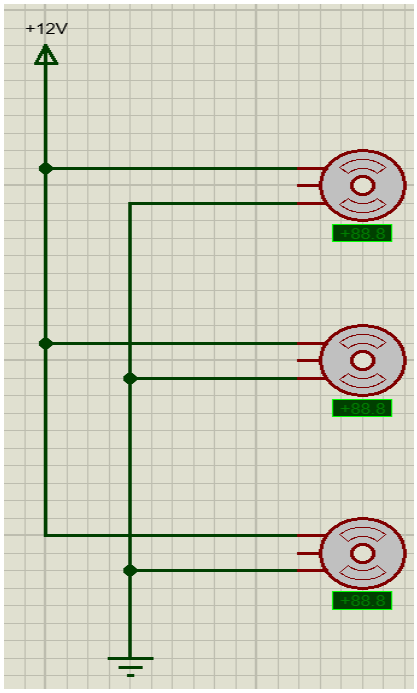


Рисунок 5.7 – З’єднання елементів MOTOR-PWMSERVO

Для створення сигналів потрібної частоти та тривалості ми можемо встановити елементи, які згенерують три різні сигнали для кожного з сервоприводів. Щоб це зробити, перейдіть до режиму генератора (Generator Mode) в лівій панелі інструментів та виберіть компонент PULSE. Потім додайте три екземпляри компонента PULSE до схеми та з’єднайте їх, як показано на рис. 5.8.

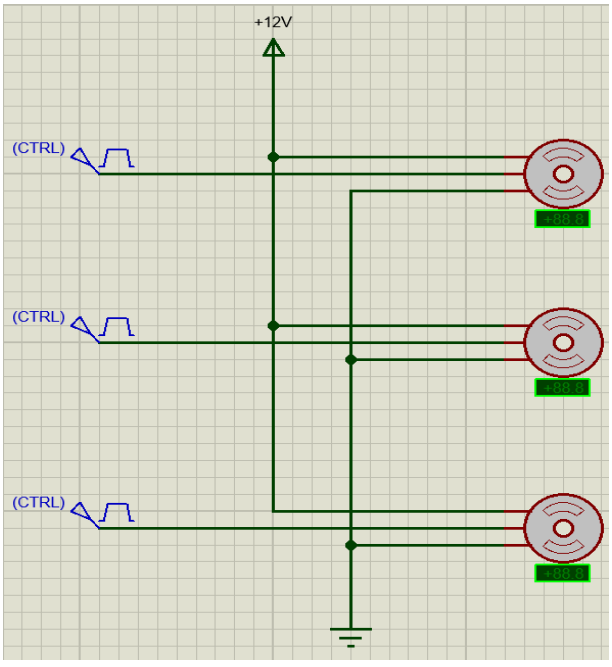


Рисунок 5.8 – Під’єднання елементів MOTOR-PWMSERVO

Згідно з рис. 5.6 налаштуємо значення ширини імпульсу трьох елементів PULSE на 1 мс, 1.5 мс та 2 мс відповідно (рис. 5.9).

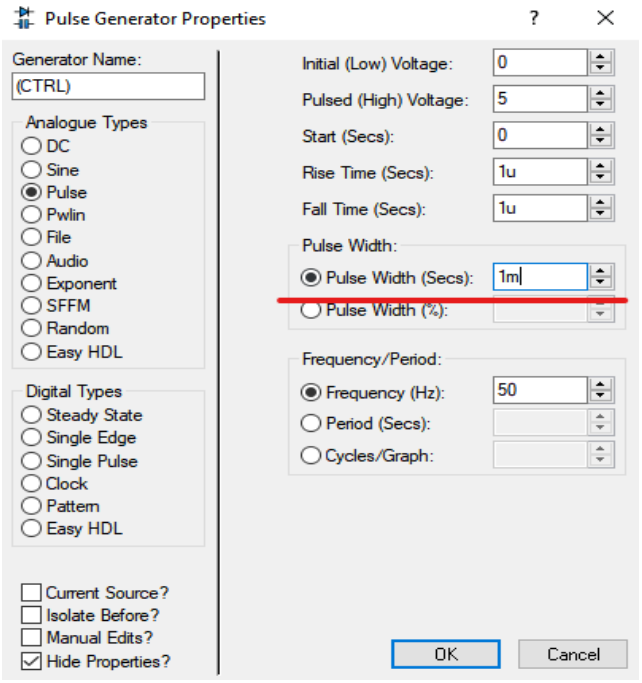


Рисунок 5.9 – Налаштування значень ширини імпульсів для елементів PULSE

Після того, як ми усе під’єднали та налаштували, можемо запускати симуляцію. Результати симуляції зображено на рис. 5.10.

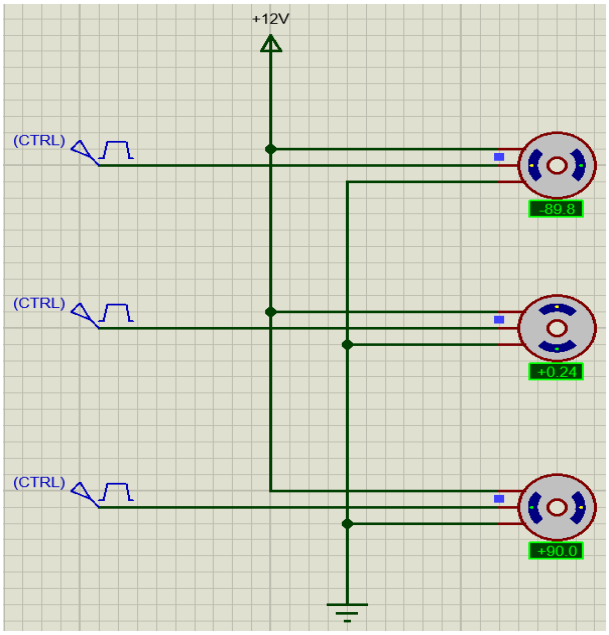


Рисунок 5.10 – Результат моделювання роботи сервомоторів в середовищі Proteus

4. Створення проєкту з сервоприводом із використанням Arduino IDE та Tinkercad

Схема підключення серводвигуна до плати Arduino зображена на рис. 5.11. Як вже було сказано, серводвигун має три приводи: живлення, сигнальний та заземлення.

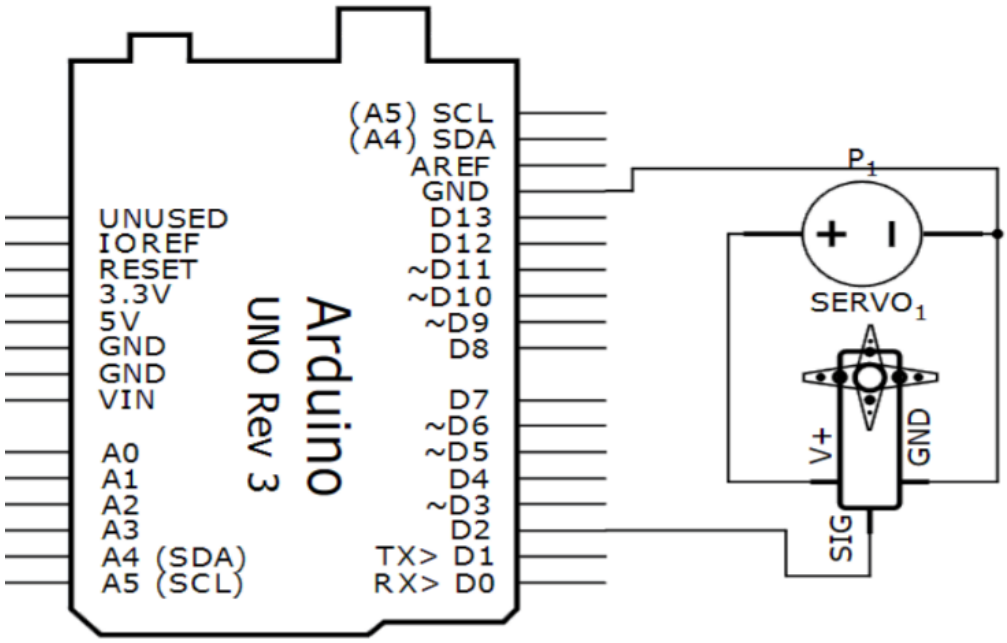


Рисунок 5.12 – Приклад підключення сервомотора

У мікроконтролері ATMega328, який використовується на платі Arduino UNO R3, для керування сервоприводом використовуються вихідні піни з підтримкою PWM, наприклад, піни 9 і 10. Для генерування PWM сигналу використовуються функції бібліотеки Servo.h, яка доступна в Arduino IDE.

Програма для керування сервоприводом може мати наступний вигляд:

```
#include <Servo.h>
Servo servo;
void setup() {
  servo.attach(9); // встановлюємо пін для керування сервоприводом
}
void loop() {
  servo.write(90); // встановлюємо кут повороту сервоприводу в 90 градусів
  delay(1000); // чекаємо 1 секунду
  servo.write(0); // встановлюємо кут повороту сервоприводу в 0 градусів
  delay(1000); // чекаємо 1 секунду
}
```

Приклад схеми для керування сервоприводом у середовищі Tinkercad може мати наступний вигляд (5.12).

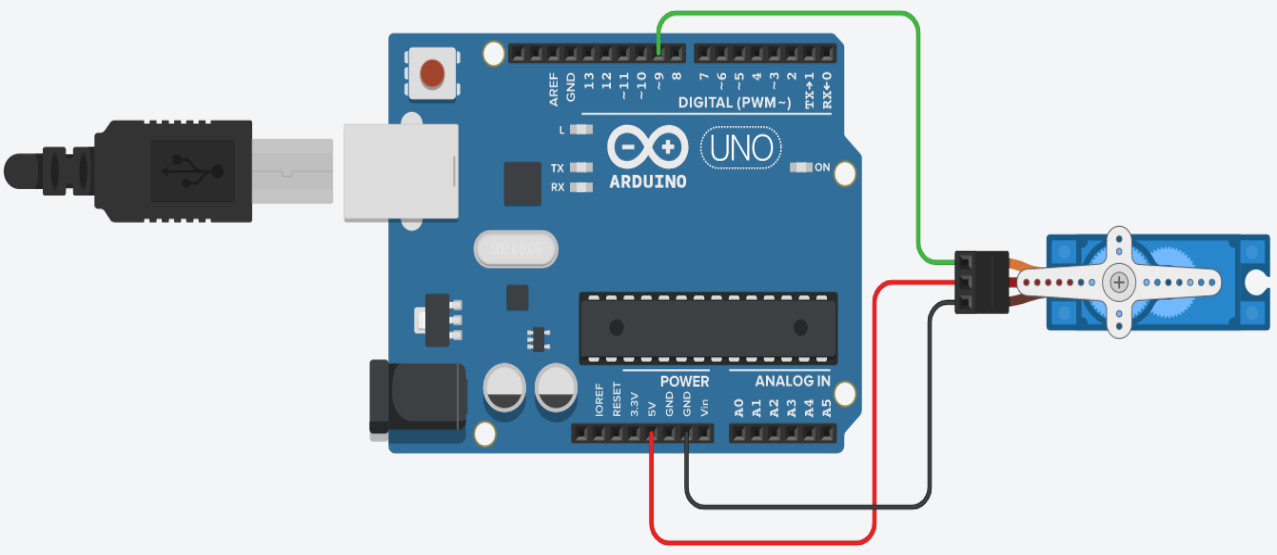


Рисунок 5.12 – Приклад програми для керування сервоприводом

4. Індивідуальне завдання

Забезпечити роботу серводвигуна відповідно завдань, наведених в табл. 5.2.

Таблиця 5.2 – Варіанти завдань

№ вар.	Завдання
1	2
1	Створити проєкт «Рух серводвигуна за допомогою потенціометра». Під'єднати серводвигун до піна **3**. При обертанні потенціометра, серводвигун повинен рухатись від 0 до 180 градусів, залежно від положення потенціометра.
2	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **11**. Кнопку під'єднати до піна **3**. При натисканні кнопки, серводвигун повинен рухатись проти годинникової стрілки. При відтисканні кнопки серводвигун повинен зупинятися.
3	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **9**. Кнопку під'єднати до піна **2**. При натисканні кнопки, серводвигун повинен рухатись за годинниковою стрілкою. При відтисканні кнопки серводвигун повинен зупинятися.
4	Створити проєкт «Рух серводвигуна за допомогою потенціометра». Під'єднати серводвигун до піна **5**. При обертанні потенціометра, серводвигун повинен рухатись від 0 до 180 градусів, залежно від положення потенціометра.

1	2
5	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **6**. Кнопку під'єднати до піна **2**. При натисканні кнопки, серводвигун повинен повернутись на **15** градусів проти годинникової стрілки, при відтисканні кнопки повернутися в початкове положення.
6	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **10**. Кнопку під'єднати до піна **3**. При натисканні кнопки, серводвигун повинен повернутись на **15** градусів за годинниковою стрілкою, при відтисканні кнопки повернутися в початкове положення.
7	Створити проєкт «Рух серводвигуна за допомогою потенціометра». Під'єднати серводвигун до піна **9**. При обертанні потенціометра, серводвигун повинен рухатись від 0 до 180 градусів, залежно від положення потенціометра.
8	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **3**. Кнопку під'єднати до піна **2**. При натисканні кнопки, серводвигун повинен повернутись на **30** градусів проти годинникової стрілки, при відтисканні кнопки повернутися в початкове положення.
9	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **6**. Кнопку під'єднати до піна **3**. При натисканні кнопки, серводвигун повинен повернутись на **30** градусів за годинниковою стрілкою, при відтисканні кнопки повернутися в початкове положення.
10	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **3**. Кнопку під'єднати до піна **2**. При натисканні кнопки, серводвигун повинен повернутись на **45** градусів проти годинникової стрілки, при відтисканні кнопки повернутися в початкове положення.
11	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **6**. Кнопку під'єднати до піна **3**. При натисканні кнопки, серводвигун повинен повернутись на **45** градусів за годинниковою стрілкою, при відтисканні кнопки повернутися в початкове положення.
12	Створити проєкт «Рух серводвигуна за допомогою потенціометра». Під'єднати серводвигун до порту **5**. При обертанні потенціометра, серводвигун повинен рухатись від 0 до 180 градусів, залежно від положення потенціометра.
13	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **6**. Кнопку під'єднати до піна **2**. При натисканні кнопки, серводвигун повинен повернутись на **75** градусів проти годинникової стрілки, при відтисканні кнопки повернутися в початкове положення.
14	Створити проєкт «Рух серводвигуна за допомогою кнопки». Під'єднати серводвигун до порту **10**. Кнопку під'єднати до піна **3**. При натисканні кнопки, серводвигун повинен повернутись на **75** градусів за годинниковою стрілкою, при відтисканні кнопки повернутися в початкове положення.

Порядок виконання роботи

1. Ознайомитися з основними принципами підключення серводвигунів до мікропроцесора ATMega328.

2. Ознайомитися із керуванням сервоприводом за допомогою ШІМ в середовищі

Proteus за допомогою мікропроцесора ATMega328.

3. Розробити програму повороту серводвигуна згідно з індивідуальним завданням.

4. Розробити модель схеми згідно з індивідуальним завданням у середовищі Tinkercad та перевірити її роботу.

5. Розробити модель схеми згідно з індивідуальним завданням у середовищі Proteus та перевірити її роботу.

6. Зібрати схему на монтажній платі згідно індивідуального завдання, ввести програму в Arduino UNO R3, запустити та перевірити їх роботу.

7. Перевірити правильність функціонування програм у середовищах моделювання та на реальній схемі.

8. Оформити звіт про роботу.

Зміст звіту

1. Тема лабораторної роботи.

2. Мета роботи.

3. Індивідуальне завдання.

4. Програма і спрощена блок-схема алгоритму організації процесу управління сервоприводом за допомогою мікропроцесора ATMega328 згідно індивідуального завдання.

5. Скріншот та посилання на працюючу модель схеми згідно індивідуального завдання у середовищі Tinkercad.

6. Працююча модель схеми згідно індивідуального завдання у середовищі Proteus.

7. Фото працюючої зібраної схеми на монтажній платі згідно індивідуального завдання.

8. Висновки.

Практична робота 6

ДОСЛІДЖЕННЯ ОРГАНІЗАЦІЇ ВВОДУ ІНФОРМАЦІЇ З ДАТЧИКІВ ТА ІНДИКАЦІЇ РЕЗУЛЬТАТІВ НА МІКРОПРОЦЕСОРІ ATMEGA328

Мета роботи: Ознайомитись з особливістю будови та різновидами датчиків, дослідити роботу елементів вводу та виводу на мікропроцесорі ATMega328 на платформі Arduino. Отримати практичні навички роботи з датчиками та елементами індикації результатів на мікропроцесорі ATMega328 на платі Arduino UNO R3.

Теми для попереднього проороблення:

– теоретичні відомості про підключення LCD дисплея до мікропроцесора ATMega328;

– теоретичні відомості про динамічну індикацію;

– теоретичні відомості про програмування в Arduino.

1. Ознайомлення із поняттям датчика

Датчики для Arduino – це електронні пристрої, які вимірюють різні параметри фізичного середовища та перетворюють їх у сигнали, які можуть бути зрозумілі мікроконтролеру Arduino. Ці датчики можуть вимірювати різні параметри, такі як температура, вологість, тиск, освітленість, рух, звук та багато інших.

Датчики можуть бути підключені до Arduino за допомогою різних інтерфейсів, таких як аналогові та цифрові порти, інтерфейси шини I2C, SPI та UART. Деякі датчики можуть також використовувати протоколи бездротового зв’язку, такі як Bluetooth та Wi-Fi.

Один з ключових аспектів датчиків для Arduino – це їх точність та надійність. Якщо датчик недостатньо точний або надійний, то це може призвести до неправильних вимірів, які можуть призвести до невірної поведінки системи, що контролюється за допомогою Arduino. Одним з найпопулярніших датчиків для Arduino є датчик температури DS18B20, який вимірює температуру з точністю до $0{,}5^0$C. Інший популярний датчик – це датчик вологості DHT11, який вимірює

вологість та температуру повітря. Датчик освітленості BH1750FVI є досить точним датчиком освітленості з інтерфейсом I2C (рис. 6.1).

а) *б)* *в)*

Рисунок 6.1 – Датчики: температури DS18B20 (а), вологості DHT11 (б), освітленості BH1750FVI (в)

Загалом, датчики для Arduino є важливою складовою систем контролю та збору даних. Вони дозволяють вимірювати різні параметри фізичного середовища та забезпечують необхідну інформацію для прийняття рішень та керування системами.

Датчик температури DS18S20 – це цифровий термометр з одним проводом, який може вимірювати температуру в діапазоні від -55°C до +125°C з точністю до ±0,5°C. Цей датчик здатний працювати з мікроконтролерами, такими як Arduino, за допомогою інтерфейсу шини 1-Wire.

DS18B20 складається з термістора, з якого зчитується температура, а також з логічного інтерфейсу, який перетворює сигнали від термістора на цифрові дані, що передаються через шину 1-Wire. Це означає, що датчик має тільки один провідник для передачі даних та живлення, що дозволяє йому бути досить простим у використанні та підключенні до мікроконтролерів. На рис. 6.2 показано, як можна підключити DS18B20 до Arduino.

DS18B20 має вбудовану пам’ять для зберігання конфігураційних даних та останнього виміру температури. Для зчитування даних з датчика необхідно відправити запит на отримання даних, після чого DS18B20 передає цифрові дані через шину 1-Wire. Також, DS18B20 може бути налаштований на роботу в режимі зворотного зв’язку, в якому він постійно надсилає дані мікроконтролеру без запиту.

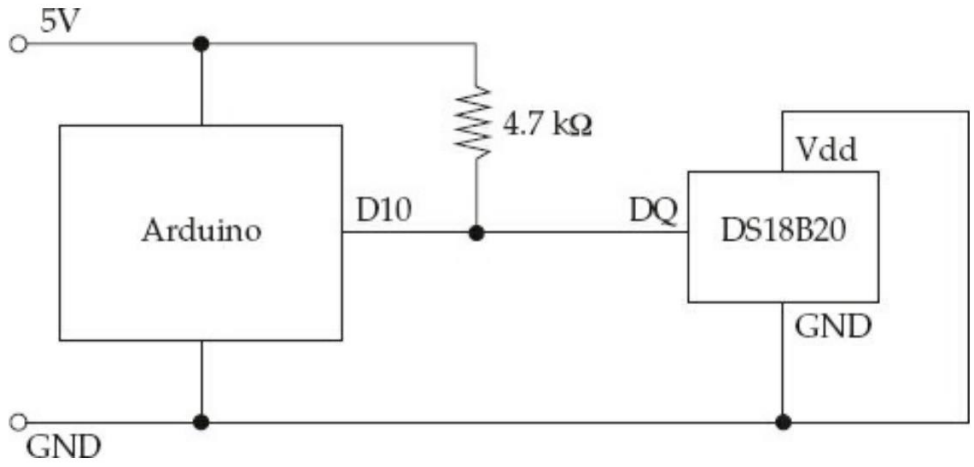


Рисунок 6.2 – Підключення 1-Wire пристрою до Arduino

2. Створення проєкту з датчиком температури TMP36 в середовищі TinkerCad і Proteus

TMP36 та DS18B20 – це два різних типи датчиків температури зі своїми особливостями. Основна відмінність між цими датчиками полягає у їхній конструкції та інтерфейсі з платою мікроконтролера. DS18B20 – це цифровий датчик температури, який працює за допомогою 1-провідного інтерфейсу, тоді як TMP36 – це аналоговий датчик температури, що вимірює напругу, яка змінюється в залежності від температури.

TMP36 – це інтегральна мікросхема, яка є датчиком температури з високою точністю, низьким споживанням енергії та простотою використання. Вона має вбудований прецизійний термостат з точністю ±1 °C, що дозволяє вимірювати температуру в діапазоні від -40 °C до +125 °C. TMP36 має три виводи: VCC, GND та VOUT. До джерела живлення підключається VCC, а GND – до землі, VOUT видає аналоговий сигнал напруги, який змінюється в залежності від температури (рис. 6.3). За допомогою цього сигналу можна виміряти температуру, підключивши VOUT до аналогового входу мікроконтролера або АЦП. TMP36 має низький коефіцієнт температурної похибки та велику стійкість до змін, що робить його ідеальним для вимірювання температури в різних застосуваннях, таких як контроль температури відповідальних систем, домашніх приладів, термометрів та інших пристроїв.

Рисунок 6.3 – Датчик температури TMP36

Створимо новий проєкт у середовищі Proteus. Зберемо схему, яка відповідатиме наступному завданню:

– коли температура менше 0 ^{0}C світиться синій світлодіод (3 пін);

– коли температура від 0 ^{0}C до 50 ^{0}C світиться помаранчевий світлодіод (4 пін);

– коли температура більше 50 ^{0}C світиться червоний світлодіод (5 пін).

Для схеми необхідно додати елементи, які показані на рис. 6.4, потім додаємо Ground та Power. Після цього отримуємо схему, як показано на рис. 6.5.

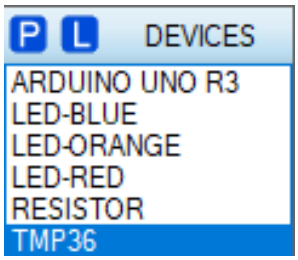


Рисунок 6.4 – Додавання до схеми елементів

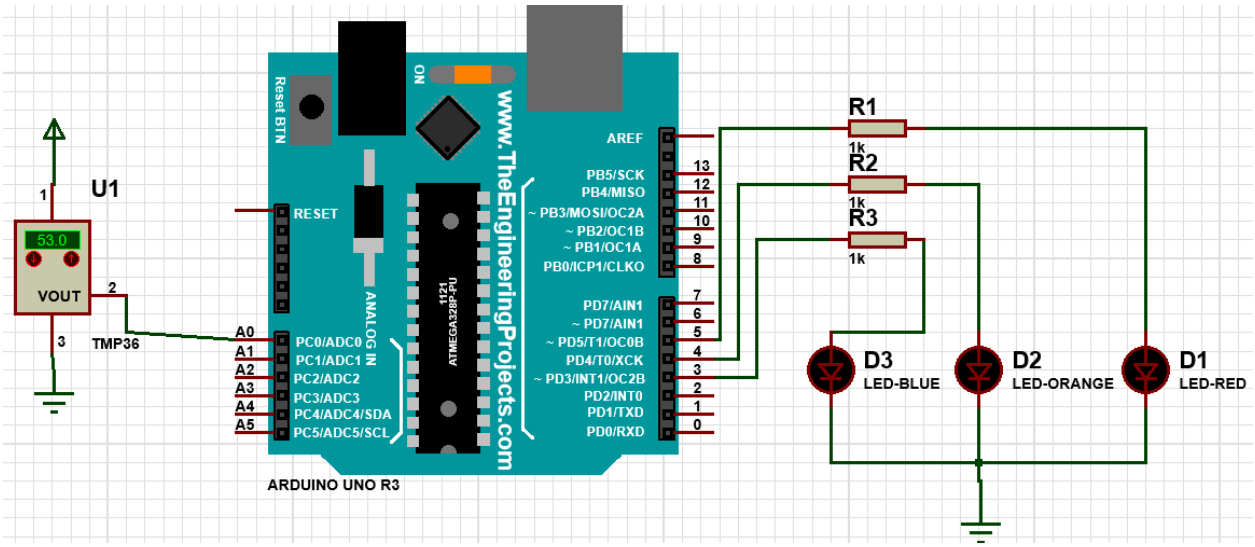
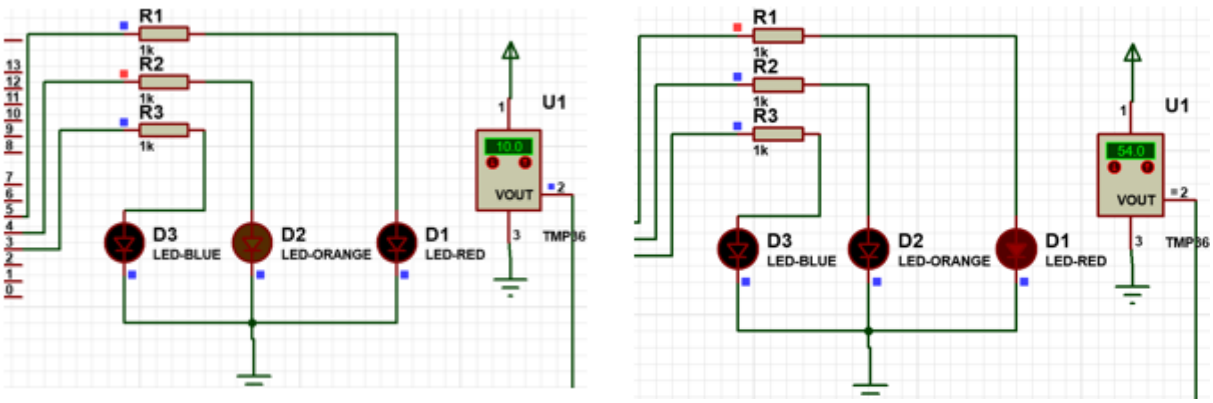


Рисунок 6.5 – Схема проєкту у середовищі Proteus вимірювача на датчику температури TMP36

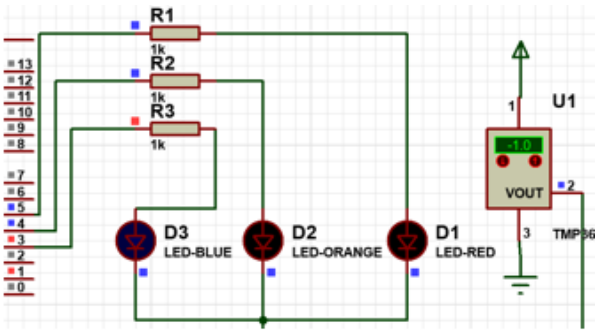
Далі наведено код програми найпростішого прикладу, який задовольняє умові завдання:

```
void setup() {
  pinMode(A0, INPUT); // встановлення режиму вхідного піна A0
  pinMode(3, OUTPUT); // встановлення режиму вихідного піна 3
  pinMode(4, OUTPUT); // встановлення режиму вихідного піна 4
  pinMode(5, OUTPUT); // встановлення режиму вихідного піна 5
  Serial.begin(9600); // встановлення швидкості передачі даних
}
void loop() {
  int tmp = analogRead(A0); // зчитування даних з датчика
  float voltage = (tmp * 5.0) / 1024; // перетворення 10-бітного числа у вимірювання напруги
  float milliVolt = voltage * 1000; // це множиться на 1000 для перетворення його у мілівольти
  float tmpCel = (milliVolt - 500) / 10; // для датчика TMP36. Діапазон (-40 °C до +125 °C)
  float tmpFer = (((tmpCel * 9) / 5) + 32); // використовується для перетворення градусів Цельсія -> Фаренгейта
  digitalWrite(3, LOW); // встановлення вихідного піна 3 в рівень LOW
  digitalWrite(4, LOW); // встановлення вихідного піна 4 в рівень LOW
  digitalWrite(5, LOW); // встановлення вихідного піна 5 в рівень LOW
  if (tmpCel <= 0) {
    digitalWrite(3, HIGH); // встановлення вихідного піна 3 в рівень HIGH
    delay(1000); // затримка на 1 секунду
    digitalWrite(3, LOW); // встановлення вихідного піна 3 в рівень LOW
    delay(1000); // затримка на 1 секунду
  } else if (tmpCel > 0 && tmpCel <= 50) {
    digitalWrite(4, HIGH); // встановлення вихідного піна 4 в рівень HIGH
    delay(1000); // затримка на 1 секунду
    digitalWrite(4, LOW); // встановлення вихідного піна 4 в рівень LOW
    delay(1000); // затримка на 1 секунду
  } else {
    digitalWrite(5, HIGH); // встановлення вихідного піна 5 в рівень HIGH
    delay(1000); // затримка на 1 секунду
    digitalWrite(5, LOW); // встановлення вихідного піна 5 в рівень LOW
    delay(1000); // затримка на 1 секунду
  }
}
```

Створимо бінарний файл та перевіримо результат, як показано на рис. 6.6.



а) *б)*



в)

Рисунок 6.6 – Результат роботи проєкту у середовищі Proteus для вимірювача на датчику температури TMP36

Повторимо ті самі дії у середовищі Tinkercad (рис. 6.7 – 6.8).

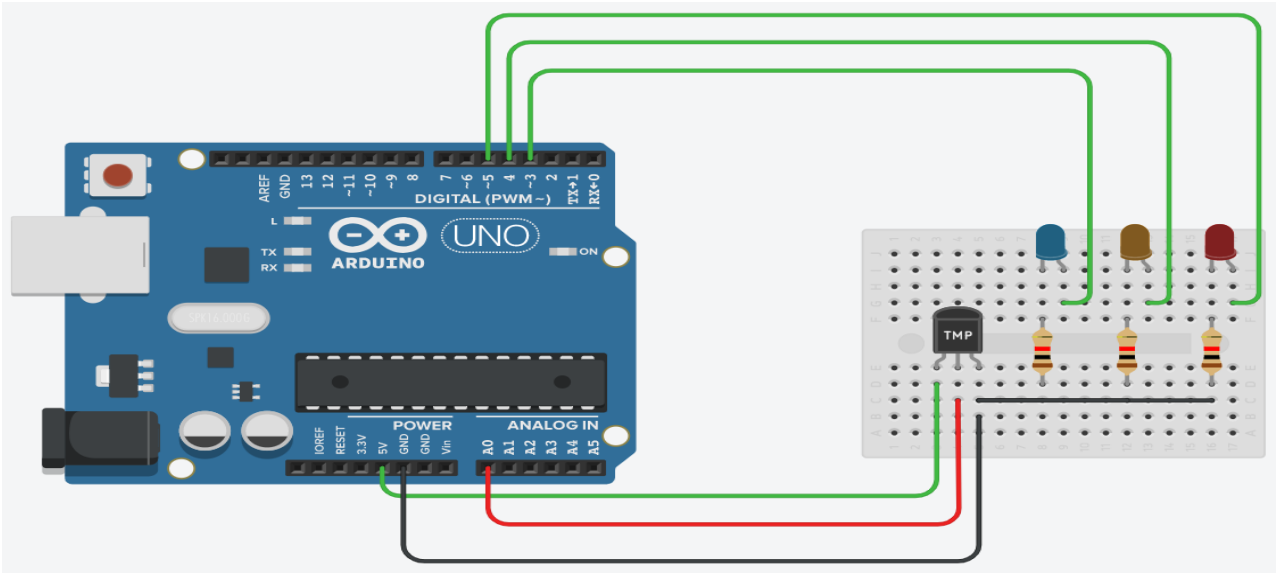


Рисунок 6.7 – Схема проєкту вимірювача на датчику температури TMP36 у середовищі TinkerCad

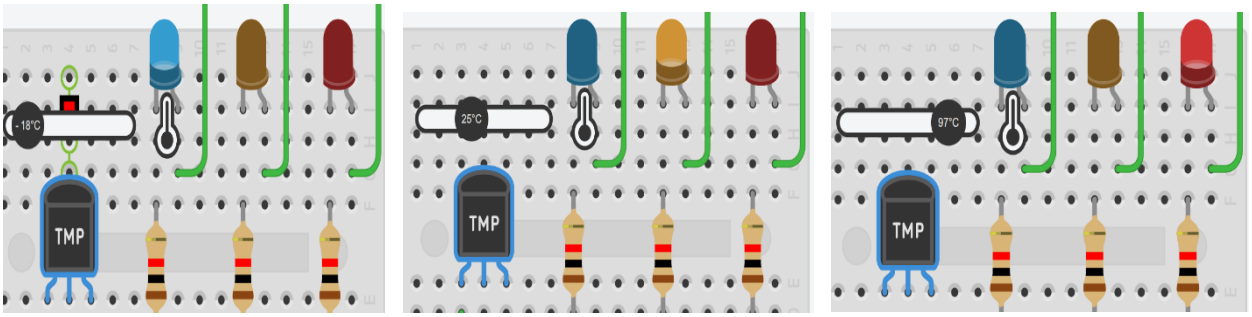


Рисунок 6.8 – Результат роботи проєкту у середовищі TinkerCad для вимірювача на датчику температури TMP36

3. Підключення LCD індикаторів до мікропроцесора

Як було зазначено в практичній роботі 3, рідкокристалічні дисплеї, що відображають символьну інформацію, наприклад текст і числа, є найбільш недорогими і простими у використанні серед усіх LCD та називаються індикаторами. Схема підключення LCD 1602 до Arduino UNO наведена на рис. 6.9.

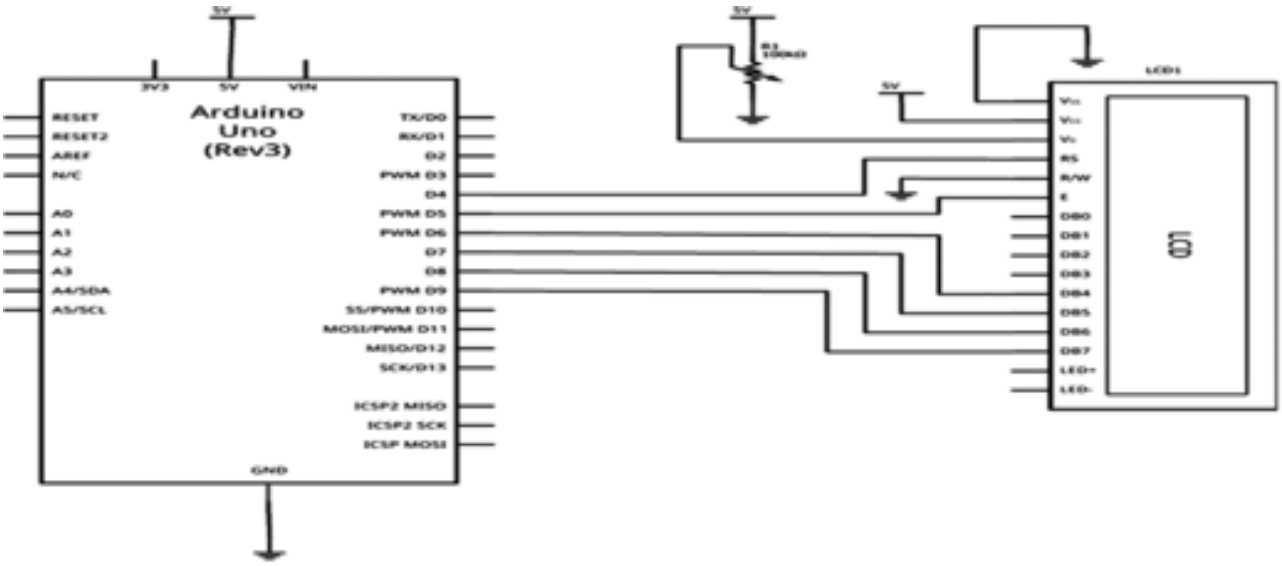


Рисунок 6.9 – Схема підключення LCD 1602 до Arduino UNO

Схему підключення LCD 1602 до Arduino UNO у середовищі TinkerCad наведено на рис. 6.10.

Далі наведено код програми найпростішого прикладу роботи LCD 1602:

```
#include <LiquidCrystal.h>
int seconds = 0;
```

```
LiquidCrystal lcd_1(12, 11, 5, 4, 3, 2);
void setup()
{
  lcd_1.begin(16, 2); // set up the number of columns and rows on the LCD.
  // print a message to the LCD.
  lcd_1.print("hello world!");
}
void loop()
{
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting
  // begins with 0):
  lcd_1.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd_1.print(seconds);
  delay(1000); // wait for 1000 millisecond(s)18b
  seconds += 1;
}
```

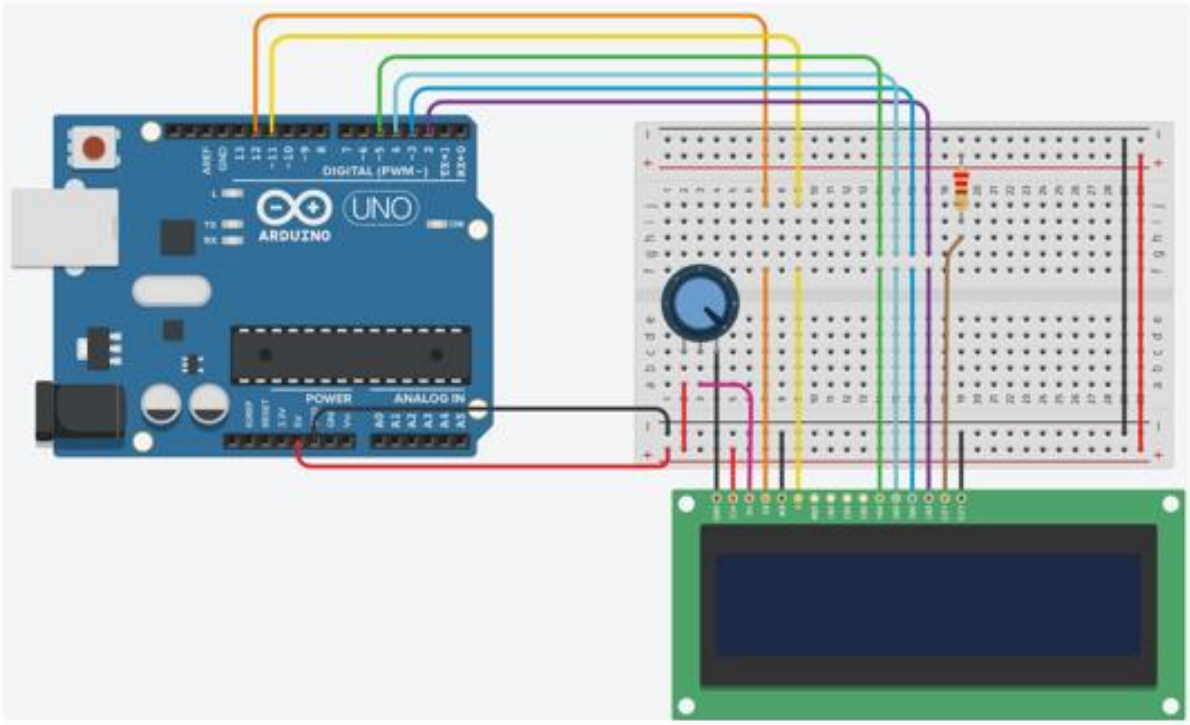


Рисунок 6.10 – Схема підключення LCD 1602 до Arduino UNO у середовищі TinkerCad

4. Індивідуальне завдання

Забезпечити роботу датчику відповідно до завдань, наведених в табл. 6.1.

Таблиця 6.1 – Варіанти завдань

№ вар.	Завдання
1	Створити проєкти «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **10** в САПР Proteus. Під'єднати датчик температури **TMP36** до піна **A0** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Цельсію.
2	Створити проєкт «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **9**. Під'єднати датчик температури **TMP36** до порту **A1** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Фаренгейту.
3	Створити проєкт «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **8**. Результат вимірювання має виводитися на LCD у градусах по Цельсію.
4	Створити проєкт «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **7**. Під'єднати датчик температури **TMP36** до порту **A2** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Фаренгейту.
5	Створити проєкт «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **6**. Під'єднати датчик температури **TMP36** до порту **A3** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Цельсію.
6	Створити проєкт «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **5**. Під'єднати датчик температури **TMP36** до піна **A4** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Фаренгейту.
7	Створити проєкт «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **5**. Під'єднати датчик температури **TMP36** до піна **A5** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Цельсію.
8	Створити проєкт «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **4**. Під'єднати датчик температури **TMP36** до піна **A0** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Фаренгейту.
9	Створити проєкт «Вимірювання температури» у середовищі Tinkercad та Proteus. Під'єднати датчик температури **DS18B20** до піна **3**. Під'єднати датчик температури **TMP36** до піна **A1** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Цельсію.
10	Створити проєкт «Вимірювання температури». Під'єднати датчик температури **DS18B20** до піна **2**. Під'єднати датчик температури **TMP36** до піна **A2** в Tinkercad. Результат вимірювання має виводитися на LCD у градусах по Фаренгейту.

Порядок виконання роботи

1. Ознайомитися з основними принципами підключення датчиків до мікропроцесора ATMega328.

2. Розробити програму вводу інформації з датчиків та індикації результатів.

3. Розробити програму роботи датчика згідно з індивідуальним завданням.

4. Розробити модель схеми згідно з індивідуальним завданням у середовищі Tinkercad та перевірити її роботу.

5. Розробити модель схеми згідно з індивідуальним завданням у середовищі PROTEUS та перевірити її роботу.

6. Зібрати схему на монтажній платі згідно з індивідуальним завданням, ввести програму в Arduino UNO R3, запустити та перевірити її роботу.

7. Перевірити правильність функціонування програми у середовищах моделювання та на реальній схемі.

8. Оформити звіт про роботу.

Зміст звіту

1. Тема лабораторної роботи.

2. Мета роботи.

3. Індивідуальне завдання.

4. Програми і спрощені блок-схеми алгоритмів організації процесу вводу інформації з датчиків та індикації результатів за допомогою мікропроцесора ATMega328 та LCD індикатора згідно індивідуального завдання.

5. Скріншот та посилання на працюючу модель схеми згідно індивідуального завдання у середовищі Tinkercad.

6. Працююча модель схеми згідно індивідуального завдання у середовищі Proteus.

7. Фото працюючої зібраної схеми на монтажній платі згідно індивідуального завдання.

8. Висновки.

Список літератури

1. Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Програмування мікропроцесорів» для студентів денної та заочної форм навчання за спеціальністю «Комп'ютерна інженерія» / А. О. Подорожняк, С. Г. Межерицький, Г. В. Гейко. – Харків : НТУ «ХПІ». – 2020. – 36 с.
2. Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Структура та функціонування мікропроцесорів» для студентів денної та заочної форм навчання за спеціальністю «Комп'ютерна інженерія» / А. О. Подорожняк, С. Г. Межерицький, Г. В. Гейко, В. В. Лимаренко. – Харків : НТУ «ХПІ». – 2020. – 56 с.
3. Методичні вказівки до самостійної роботи студентів з навчальної дисципліни «Структура та функціонування мікропроцесорів» для студентів денної та заочної форм навчання за спеціальністю «Комп'ютерні інженерія» / А. О. Подорожняк, Г. В. Гейко. – Харків : НТУ «ХПІ». – 2020. – 20 с.
4. Методичні вказівки до самостійної роботи студентів з навчальної дисципліни «Програмування мікропроцесорів» для студентів денної та заочної форм навчання за спеціальністю «Комп'ютерні інженерія» / А. О. Подорожняк, Н. Ю. Любченко. – Харків : НТУ «ХПІ». – 2021. – 15 с.
5. Проектування та аналіз електричних схем в програмному середовищі Proteus VSM. Методичні вказівки до самостійної роботи студентів з курсу «Проектування мікропроцесорних систем керування технологічними процесами» / Медвідь В.Р., Пісьціо В.П. – Тернопіль: ТНТУ, 2018. – 26 с.
6. Методичні вказівки до лабораторних робіт з навчальної дисципліни «Електроніка та мікропроцесорні системи» для здобувачів вищої освіти першого (бакалаврського) рівня за спеціальністю 015.10 «Професійна освіта. Комп'ютерні технології» денної та заочної форм навчання / Василець С.В., Василець К.С. – Рівне: НУВГП, 2019. – 134 с.
7. Мікропроцесорні та мікроконтролерні системи: Частина 2. Проектування мікропроцесорних систем: Лабораторний практикум: навч. посіб. для студ. освітньої програми «Інтегровані інформаційні системи» спеціальності 126

«Інформаційні системи та технології» / А.О. Новацький. – Київ: КПІ ім. Ігоря Сікорського, 2021. – 268 с.

8. Програмування мікропроцесорів у захищеному режимі: навчально-методичний посібник / І. С. Зиков, С. Г. Межерицький, А. О. Подорожняк, І. П. Хавіна. – Харків: ТОВ «ДІСА ПЛЮС», 2018. – 264 с.

9. Поворознюк А. І. Архітектура комп’ютерів. Методичні вказівки до виконання та оформлення курсового проекту для студентів денної та заочної форми навчання за напрямком 123 «Комп’ютерна інженерія» / А. І. Поворознюк, О. А. Поворознюк, Г. Є. Філатова. – Харків: НТУ «ХПІ», 2022. – 42 с.

10. Рисований О. М. Системне програмування: підручник для студенів напряму «Комп’ютерна інженерія» вищих навчальних закладів / О. М. Рисований. – Харків: НТУ «ХПІ», 2010. – 912 с.

11. Simon Monk. Programming Arduino. Getting Started with Sketches. – McGrow Hill, 2012. – 177 p.

12. Simon Monk. Programming Arduino. Next Steps. Going Further with Sketches. – McGrow Hill, 2014. – 212 p.

13. Proteus Design Suite. Getting Started Guide. – Labcenter Electronics Ltd. – 2020. – 191 p.

14. The Arduino Platform and C Programming. Introduction to Programming the Internet of Things (IOT). [Електронний ресурс] URL: https://www.coursera.org/learn/arduino-platform (дата звернення 11.12.2023).

15. ATmega328p datasheet [Електронний ресурс] URL: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (дата звернення 11.12.2023).

16. СТЗВО-ХПІ-3.01-2021. Текстові документи у сфері навчального процесу. Загальні вимоги до виконання. – Харків: НТУ «ХПІ», 2021. – 47 с.

ЗМІСТ

Навчальне видання

Методичні вказівки
до виконання практичних робіт з навчальної дисципліни
«Архітектура та програмування мікропроцесорів»
для студентів денної та заочної форми навчання
за спеціальністю «Комп’ютерна інженерія»

Укладачі:
ПОДОРОЖНЯК Андрій Олексійович,
ГЕЙКО Геннадій Вікторович,
МЕЖЕРИЦЬКИЙ Сергій Геннадійович,
ЛЮБЧЕНКО Наталія Юріївна

Відповідальний за випуск проф. Олександр ЗАКОВОРОТНИЙ
Роботу до видання рекомендував проф. Микола ЗАПОЛОВСЬКИЙ

В авторській редакції

План 2024 р., поз. 129.
Підп. до друку 13.03.2024. Формат 60x84 1/16.
Папір офсет. Друк ризографічний. Ум. друк. арк. 2,8.

Видавничий центр НТУ «ХПІ»,
вул. Кирпичова, 2, м. Харків, 61002
Свідоцтво суб’єкта видавничої справи ДК № 5478 від 21.08.2017 р.

Електронна версія