МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторного практикуму та
самостійної роботи
з дисципліни
МІКРОПРОЦЕСОРНІ ПРИСТРОЇ КЕРУВАННЯ ТА ОБРОБКИ ІНФОРМАЦІЇ
для студентів спеціальностей
152 „Метрологія та інформаційно-вимірювальна техніка“
153 „Мікро- та наносистемна техніка“
денної й заочної форм навчання

2019

Методичні вказівки до лабораторного практикуму та самостійної роботи з дисципліни «Мікропроцесорні пристрої керування та обробки інформації» для студентів спеціальності 152 „Метрологія та інформаційно-вимірювальна техніка“ 153 „Мікро- та наносистемна техніка“ денної й заочної форм навчання / Укл.: В.І.Рева. – Запоріжжя: ЗНТУ, 2019. – 90 с.

Укладач: В.І.Рева, доц., канд. фіз.-мат. наук

Рецензент: О.В. Василенко, доц., канд.техн.наук

Відповідальний за випуск: А.В. Коротун, доц., канд.фіз.-мат.наук

Затверджено
на засіданні кафедри
мікро- та наноелектроніки

Протокол №6
від “ 20 “ березня 2019 р.

Рекомендовано до видання
НМК ФРЕТ
Протокол №7
від “ 21 “ березня 2019 р.

ЗМІСТ

Вступ 6
Підготовка до проведення лабораторної роботи та оформлення звіту 7
1. Лабораторна робота №1 “Знайомство з платформою Arduino на прикладі контролера Arduino uno” 8
1.1. Загальні відомості про Arduino UNO 8
1.2 Середовище Arduino IDE 12
1.3. Програмування на мові Arduino IDE 13
1.4. Хід роботи 14
1.4.1. Перша програма в Arduino IDE 14
1.4.2. Зібрати світлофор, використовуючи для індикації світлодіоди 15
1.4.3. Зібрати більш складну систему включає в себе 2 пов'язаних світлофора 17
1.4.4. Додамо в цю систему управління 19
1.4.5. Підвищення відмовостійкісті 21
1.5. Контрольні запитання 23
2. Лабораторна робота №2 “Робота з Com–портом та переферичними пристроями” 23
2.1. Короткі теоретичні відомості 23
2.1.1. Потенціометр 23
2.1.2. Знайомство з COM-портом 24
2.2. Короткі теоретичні відомості 25
2.2.1. Відправка команд з ПК 25
2.2.2. Управління пристроєм через COM-порт (частина 1) . . . 27
2.2.3. Управління пристроєм через COM-порт (частина 2) . . . 29
2.2.4. Повернення рядки відправленої через COM-порт 29
2.2.5. Управління пристроєм через COM-порт (частина 3) за допомогою команд 30
2.2.6. Управління пристроєм через COM-порт (частина 4) управління світлодіодом 32
2.2.7. Управління пристроєм через COM-порт (частина 5). Управління світлодіодом 33
2.2.8. Передача цифрового значення датчика через COM порт 35
2.2.9. Передача значення напруги через COM порт 35

2.3. Контрольні запитання 36
3. Лабораторна робота №3 „Керування Arduino через Com-порт” 36
3.1. Короткі теоретичні відомості 36
3.2. Хід роботи 38
3.2.1. Управління пристроєм через COM-порт (частина 1) за допомогою команд 38
3.2.2. Управління пристроєм через COM-порт (частина 2) управління світлодіодом 39
3.2.3. Управління пристроєм через COM-порт (частина 3) управління світлодіодом 40
3.2.4. Повертати серводвигун за і проти годинникової стрілки на 180 градусів 42
3.2.5. Керувати кутом повороту сервоприводу за допомогою потенціометра 44
3.2.6. Керувати кутом повороту сервоприводу за допомогою потенціометра 45
3.3. Контрольні запитання 47
4. Лабораторна робота №4 “Робота з LCD -індикаторами на базі контролера HD44780” 47
4.1. Організація індикації у МПС на базі HD44780 47
4.1.1. Програмування і управління LCD на базі HD4470 51
4.2. Хід роботи 56
4.2.1. Вивести на екран текст: My name is 56
4.2.2. Вивести на екран текст: Hello world і час роботи програми 57
4.2.3. Вивести на екран текст: Hello world і зворотній відлік часу від 1000с 58
4.2.4. Вивести на екран повідомлення прийняте з COM-порту. Схема залишається колишньою 59
4.2.5. Вивести на екран повідомлення прийняте з COM-порту. Залежно від змісту. Схема залишається колишньою . . 59
4.2.6. Вивести значення температури з датчика 61
4.3. Контрольні запитання 62
5. Лабораторна робота №5 “Керування кроковим двигуном” . 63
5.1. Теоретичні відомості 63
5.1.1 Крековий двигун 63
5.1.2. Схема підключення 65

5.1.3. Створення програмного коду 66
5.1.4. Скетч запуску крокового двигну без використання бібліотек 67
5.2. Хід роботи 69
5.2.1. Керування сервоприводом зі зміною його кута повороту, а також з можливістю зміни швидкості 69
5.3. Контрольні запитання 70
6. Лабораторна робота №6 “Підключення RFID мітки до Arduino” 71
6.1. Загальні теоретичні відомості 71
6.2. Хід роботи 72
6.2.1. Зчитування коду карти (RFID мітки) 72
6.2.2. Організація доступу з індикацією 75
6.3. Контрольні запитання 81
7. Лабораторна робота №6 “Підключення модуля годиника реального часу DS1302 I LCD екрана до контролера Arduino” 82
7.1. Загальні теоретичні відомості 82
7.1.1. Короткий опис модуля DS1302 82
7.2. Хід роботи 84
7.2.1. Виведення на екран поточний часу і дати, заданого з комп'ютера 84
7.3 Контрольні запитання 88
Додаток А Приклад оформлення титульної сторінки 90

ВСТУП

Мікропроцесори (МП) і мікропроцесорні системи (МПС) є в даний час найбільш масовими засобами обчислювальної техніки. Після появи перших МП у 1971 р., число нових розробок мікропроцесорних БІС продовжує безупинно збільшуватися.

Досвід розробки різних систем показав, що очікувана від МП універсальність є в значній мірі обмеженою. Розмаїтість мікропроцесорних БІС, що виражається в їх різній базовій технології, архітектурі, технічних характеристиках поставило розроблювачів цифрових систем перед складними проблемами, а саме: які особливості цього класу пристроїв і тенденції його розвитку, чим відрізняється використання могутніх мікро-ЕОМ від застосування міні-ЕОМ, як проектувати системи на основі МП, у чому полягають особливості підходів до розробки програмного забезпечення мікро-ЕОМ і багато інших.

Рішення цих проблем вимагає від розробника систем глибоких знань цифрової обчислювальної техніки (ОТ), включаючи технічні й алгоритмічні питання, представлення особливостей МП БІС, а головне досвіду розробки систем на основі МП.

Курс "Мікропроцесорні пристрої керування та обробки інформації" вивчається протягом одного семестру.

В результаті вивчення дисципліни студент повинний знати:

− характеристики і параметри, умовні графічні зображення і принцип дії основних різновидів мікропроцесорів і мікроконтроле-рів;

− принципи побудови мікро-ЕОМ на основі різних мікропроцесорних комплектів;

− методику аналізу і вибору оптимального технічного рішення;

− системи команд мікропроцесорів і правила складання програмного забезпечення РЕА на основі мікропроцесорних систем;

− сучасний стан мікропроцесорної і комп'ютерної техніки і перспективи їхнього розвитку.

ПІДГОТОВКА ДО ПРОВЕДЕННЯ ЛАБОРАТОРНОЇ РОБОТИ ТА ОФОРМЛЕННЯ ЗВІТУ

Лабораторна робота включає самостійне опрацювання теоретичного матеріалу, вивчення документації з апаратних засобів, методик проведення розрахунків, програмування і моделювання, обробку й інтерпретацію результатів. При проведенні лабораторного практикуму необхідно:

- підготуватися до експрес-опитування з теоретичного матеріалу, необхідного для виконання роботи (щоб отримати допуск до лабораторної роботи);
- підготувати і оформити план виконання лабораторної роботи;
- після виконання лабораторної роботи провести необхідні розрахунки і оформити звіт, який повинен містити:

1) титульний аркуш (див. Додаток А);
2) мету роботи;
3) опис словами;
4) алгоритм роботи у вигляді блок схеми;
5) необхідні математичні розрахунки роботи пристрою;
6) текст розробленої програми;
7) основні результати моделювання пристрою;
8) стислі висновки по роботі, в яких:

- вказати, що досліджувалось;
- вказати методику (чи метод) дослідження;
- вказати особливості алгоритму роботи пристрою;
- визначити характер роботи пристрою при моделюванні;
- відзначити подібність і відмінність результатів моделювання від теоретичних даних;
- здати звіт з виконаної лабораторної роботи та відповісти на запитання викладача (отримати залік за виконану роботу).

1 ЛАБОРАТОРНА РОБОТА №1 “ЗНАЙОМСТВО З ПЛАТФОРМОЮ ARDUINO НА ПРИКЛАДІ КОНТРОЛЕРА ARDUINO UNO”

Мета роботи – ознайомитися з базовими можливостями контроллера Arduino UNO та середовища Arduino IDE.

1.1 Загальні відомості про Arduino UNO

Різні пристрої та прилади підключаються до мікроконтролеру через порти введення-виведення. Мікроконтролер, який є ядром Arduino, − це 8-бітний мікроконтролер, і тому його порти введення-виведення також є 8-бітними. Це загальний принцип, і хоча теоретично порт введення-виведення 8-бітного мікроконтролера може бути і 16-ти, і 32-х, і 64-бітовим, але практичного сенсу оснащувати 8-бітний мікроконтролер не 8-бітовим портом вводу-виводу немає, оскільки для читання або запису даних в такий порт довелося б використовувати кілька команд асемблера, тобто операція обміну даними з таким портом відбувалася б тільки за кілька тактів.

Порт введення-виведення звичайного призначення (GPIO − General Purpose Input / Output) являє собою кілька виводів (пинов) мікроконтролера по числу біт, тобто в даному випадку це 8 виводів. Якщо подивитися на плату Arduino, то з одного боку знаходяться дві групи по 8 контактів, позначені як цифрові (Digital), а з іншого боку дві групи по 6 контактів, одна з яких - це контакти напруг живлення і пін скидання (Reset), а друга - це 6 контактів, позначених як входи для аналогових сигналів (Analog In).

З точки зору програми для мікроконтролера кожен порт − це кілька спеціальних регістрів (змінних), виробляючи читання або запис даних в які можна змінювати стан або режим роботи виводів мікроконтролера. Оскільки кожному виводу порту звичайного призначення відповідає 1 біт, то напруга на відповідному виводі може змінюватися, приймаючи логічне значення: або високий рівень (HIGH), або низький рівень (LOW).

Як правило високому рівню напруги відповідають напруга, рівне або трохи менше, ніж напруга живлення мікроконтролера (для Arduino це +5 вольт), а низькому рівню відповідає або рівень землі (0

вольт), або невелика напруга (наприклад, +0,2 або + 0,6 вольт). Половина напруги високого рівня називається високим нулем (Hi-Z).

Для логічних мікросхем існує ряд електричних стандартів напруг, заснованих на технології, по якій дана мікросхема виконана. Наприклад, TTL, CMOS і т.п. Якщо дві мікросхеми з однаковою напругою живлення виконані за різними технологіями (наприклад, одна з їх CMOS, а інша TTL), то вони як правило логічно сумісні між собою, якщо мова йде тільки про сигнали 1 або 0. Але рівень сигналу Hi-Z у них може відрізнятися, що може потребувати їх електричного узгодження.

Наприклад, рівень Hi-Z мікросхеми, яка отримує сигнал дорівнює +2В, а рівень Hi-Z мікросхеми, яка сигнал то приймає, то передає +2,4В). Таким чином друга мікросхема передає сигнал першої, і потім переводить порт в стан введення даних, тобто у неї на виведенні рівень Hi-Z +2.4В. А перша читає порт, а там логічна одиниця.

Але на практиці з подібними проблемами можна зіткнутися, хіба що намагаючись передати сигнал з виходу мікросхеми з напругою живлення +3,3 на вхід мікросхеми з напругою живлення + 5В. Якщо у другий рівень Hi-Z становить 2В, то електричного узгодження не буде потрібно, а якщо 2,4 В, то потрібно поставити між цими виводами схему узгодження рівнів на польовому транзисторі.

Проте, при підключенні мікросхем один до одного це слід враховувати, оскільки мені доводилося стикатися наприклад з тим, що мікросхема живиться від напруги +5В, а логічна одиниця у неї +3,2В, і хоча вона при цьому працює, але нагрівається під час роботи градусів до 60 за Цельсієм. Тобто напруга живлення +5 В, а порти введення-виведення, хоч і сумісні з 5-вольтової логікою, однак 3.3-вольтові, тобто для роботи з такою мікросхемою краще використовувати мікроконтролер з напругою живлення +3,3 (прим. Це так званий електричний стандарт інтерфейсу PCI).

Мікроконтролер Arduino, до речі, цілком може працювати на частоті 8МГц і з напругою живлення +3,3, але в конструктиві Arduino така можливість не передбачена, тому до Arduino краще підключати мікросхеми 5-вольтової логіки.

Всього мікроконтролер ATmega328p (як і ATmega168) в корпусі PDIP має три порти введення-виведення. Вони позначені, як порти B, C і D:

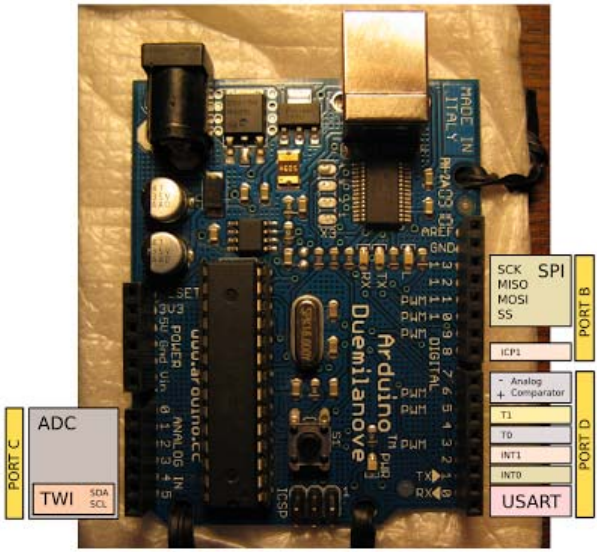


Рисунок 1.1 – Зображення контролера Arduino UNO

• Порт D - це група виводів Digital з номерами від 0 до 7.

• Порт B - це 5 виводів Digital з номерами від 8 до 13. Їм відповідають 5 молодших біт порту B (старші 2 виводи використовуються для підключення кварцового резонатора 16 МГц).

• Порт C - це група виводів Analog In. Їм відповідають 6 молодших біт порту C (7-й біт використовується як сигнал Reset, 8-й біт не використовується).

Таким чином, всього Arduino має 20 контактів введення-виведення звичайного призначення (GPIO).

Крім портів GPIO до мікроконтролеру можна підключати пристрої через спеціальні інтерфейси. Наприклад, мікроконтролер Arduino оснащений наступними інтерфейсами:

• USART - універсальний синхронний і асинхронний послідовний приймач і передавач (Universal Synchronous and Asynchronous serial Receiver and Transmitter). Виводи Digital з номерами 0 (приймач) і 1 (передавач). Він використовується, наприклад, для підключення Arduino до комп'ютера через USB, але не безпосередньо, а за допомогою мікрохеми перехідника USB-to-serial.

• Master / Slave SPI - послідовний периферійний інтерфейс (Serial Peripheral Interface). Виводи Digital з номерами від 10 до 13. У режимі slave (підлеглий) він дозволяє програмувати Arduino через ICSP за допомогою програматора. У режимі master (майстер) він дозволяє підключати до Arduino різні пристрої з інтерфейсом SPI – наприклад використовувати Arduino як програматор або підключити до Arduino флешку формату SD Card або MMC.

• TWI – двопровідний інтерфейс (Two-Wire Interface), сумісний з інтерфейсом I2C. Виводи Analog In з номерами 4 і 5. Він дозволяє підключати до Arduino, наприклад LCD-дисплеї або мікросхеми пам'яті EEPROM.

• ADC - аналого-цифровий перетворювач або АЦП (Analog-to-Digital Converter). Він дозволяє підключати до Arduino аналогові прилади. Наприклад, потенціометри, датчики (сенсори), мікрофон тощо Всього у мікроконтролера Arduino шість 10-бітних аналогових входів - група контактів Analog In.

• Аналоговий компаратор (AIN). Виводи Digital з номерами 6 (позитивний вхід) і 7 (негативний вхід). Якщо напруга на позитивному вході компаратора перевищує напруга на негативному вході, то мікроконтролер може генерувати спеціальне переривання, або використовувати цей сигнал для детектора фронтів на 16-бітному таймері. Як негативний входу замість виведення Digital 7 може бути використаний будь-який з виводів групи Analog In. Дозволяє Arduino вимірювати частоту і рівень аналогового сигналу.

• PWM - широтно-імпульний модулятор або ШІМ (Pulse Width Modulation). Виводи Digital с номерами 3, 5, 6, 9, 10, 11. Дозволяє микроконтроллеру Arduino генерувати аналогові сигнали з розрядністю від 2 біт з частотою 250 кГц до 8 біт з частотою 15625 кГц. Всього мікроконтролер Arduino має 6 ШІМ-модуляторів, по 2 на кожен з 3 таймерів.

• ICP - модуль захоплення введення (Input Capture Unit). Вивід Digital з номером 8. Дозволяє Arduino вимірювати частоту і цикл заповнення цифрового сигналу.

• T0 і T1 - Джерела зовнішнього тактового сигналу для таймерів-лічильників 0 і 1. Виводи Digital з номерами 4 і 5. Дозволяють підключати до Arduino, наприклад, мікросхеми годинника реального часу або синхронізувати Arduino з пристроями, що

працюють на частотах, які не кратні системній частоті Arduino 16 МГц (наприклад, 5 МГц або 1.8432 МГц).

Таким чином, кожен пін Arduino може використовуватися або як вивід звичайного призначення (GPIO), або як один з виводів спеціального призначення. Після натискання на кнопку скидання або відразу після включення, всі спеціальні функції виводів відключені, а самі виводи знаходяться в стані Hi-Z, тобто при скиданні вони стають електрично і логічно нейтральними.

Тобто для того, щоб задіяти для того чи іншого вивода мікроконтролера його спеціальну функцію, його слід відповідним чином програмно ініціювати. Це робиться записом потрібних значень у відповідні регістри.

1.2 Середовище Arduino IDE

Для програмування плат Arduino існує спеціальне інтегроване середовище розробки IDE Arduino. Розглянемо зовнішній вигляд програми (рис.1.2).

Рисунок 1.2 – Вікно середовища Arduino IDE

Вгорі, під рядком заголовка знаходиться рядок Меню, з наступними пунктами: Файл, Правка, Скетч, Налаштування, Сервіс.

Під рядком меню розташовується рядок інструментів, на яку винесені часто використовувані команди:

 – Перевірити

 – Загрузити

 – Створити

 – Зберегти

 – Відкрити

 – Монітор порту.

Нижче розташовується рядок з назвами відкритих вкладок. В кінці рядка знаходиться піктограма, при натисканні на яку відкривається меню, що відноситься до вкладок.

У центрі вікна знаходиться робоча область, в якій пишеться код програми. Під робочою областю знаходиться вікно повідомлень, куди виводяться повідомлення про результат компіляції, завантаження і т.д.

1.3 Програмування на мові Arduino IDE

Програма, що написана в IDE Arduino, називається скетчем. Кожен скетч повинен складатися як мінімум з двох обов'язкових функцій:

Лістинг 1.1 – Необхідні блоки скетча

```
1 void setup() {/*відкриваюча дужка на початку програми ініціалізації*/
2 /*Ця програма виконується 1 раз і призначена для приведення
системи у робочий стан, сюди можна записувати функції
ініціалізації портів і периферійних пристроїв*/
3 }/* закриваюча дужка в кінці програми ініціалізації*/
```

```
4 void loop() {/* відкриваюча дужка на початку
  основного циклу*/
  /* по суті нескінченний цикл, це основний
5 цикл програми, всередині
  якого виконується її основна частина(крім
  початкової ініціалізації)*/
6 }/* закриваюча дужка в кінці основного циклу
  */
```

Функція – структурна одиниця програми, яка має ім'я і стримає в собі деяку послідовність дій.

Більшість записів в програмі повинні закінчуватися ";", виключення складають оператори циклів, вибору, умов, а також опис прототипів функцій. З цими та іншими конструкціями ми познайомимося в процесі вивчення курсу.

На початку програми, перед функцією `setup`, зазвичай, оголошуються змінні і визначення. Після включення живлення плати першої виконується функція `setup`. Вона виконується тільки один раз. Зазвичай в ній не започатковано режими роботи портів: порти, до яких підключені різні датчики, встановлюються як входи, а порти з виконавчими пристроями як виходи.

Код, написаний в функції `loop()` починає виконуватися після виконання функції `setup()`, і виконується в нескінченному циклі знову і знову. У цій функції виконується основна робота: різні обчислення, отримання значень датчиків, висновок значень на порти.

1.4 Хід роботи

1.4.1 Перша програма в Arduino IDE

На платі Arduino UNO до 13 висновку підключений світлодіод "L" для його використання не потрібно подключасть до плати додаткових елементів. Їм і скористаємося.

В Arduino IDE напишемо наступний скетч:

Лістинг 1.2 – Програма для миготіння діодом

```
void setup()
{
```

```
  pinMode(13, OUTPUT);       // переключаємо
цифровий вивід в режим виходу
}
void loop()
{
  digitalWrite(13, HIGH);    // вмикаємо світлодіод
  delay(1000);               // чекаємо 1 секунду
  digitalWrite(13, LOW);     // вимикаємо
світлодіод
  delay(1000);               // чекаємо 1 секунду
}
```

Для завантаження скетчу в пам'ять, потрібно попередньо вибрати відповідний порт.

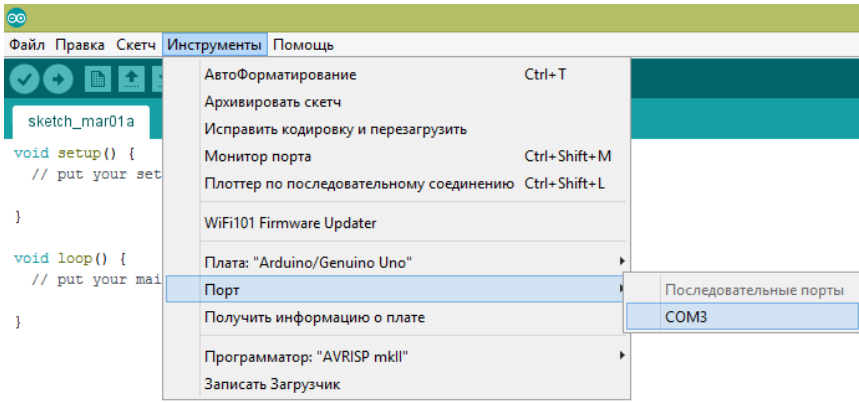


Рисунок 1.3 – Вибір порту у середовищі Arduino IDE

1.4.2 Зібрати світлофор, використовуючи для індикації світлодіоди.

Алгоритм работы светофора следующий:

1. Запалити червоний світлодіод
2. Почекати 5 сек
3. Погасити червоний світлодіод
4. Запалити жовтий світлодіод
5. Почекати 1 сек

6. Погасити жовтий світлодіод
7. Запалити зелений світлодіод
8. Почекати 5 сек
9. Погасити зелений світлодіод
10. Запалити жовтий світлодіод
11. Почекати 2 сек
12. Погасити жовтий світлодіод
13. Чи повернуться до кроку 1

Крок 1. Зберемо схему.

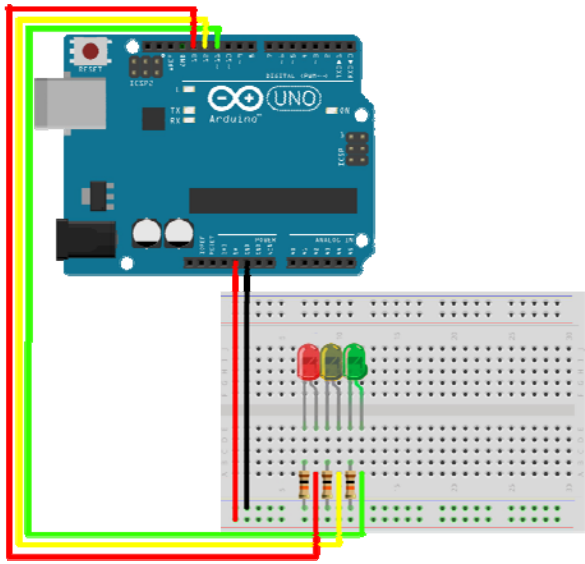


Рисунок 1.4 – Схема підключення діодів до плати Arduino UNO

Крок 2. У середовищі IDE Arduino напишемо наступний код:

Лістинг 1.3 – Програма для керування моделю світлофора

```
#define RED 13
#define YELLOW 12
#define GREEN 11
void setup() {
pinMode(RED, OUTPUT);
pinMode(YELLOW, OUTPUT);
pinMode(GREEN, OUTPUT);
```

```
}
void loop() {
digitalWrite(RED, HIGH);
delay(5000);
digitalWrite(RED, LOW);
digitalWrite(YELLOW, HIGH);
delay(1000);
digitalWrite(YELLOW, LOW);
digitalWrite(GREEN, HIGH);
delay(5000);
digitalWrite(GREEN, LOW);
digitalWrite(YELLOW, HIGH);
delay(2000);
digitalWrite(YELLOW, LOW);
}
```

1.4.3 Зібрати більш складну систему включає в себе 2 пов'язаних світлофора.

Ця схема є аналогом світлофорів на перехрестях, один для машин, інший для пішоходів. Вони повинні регулювати рух таким чином щоб перешкоджати ДТП.

Крок 1. Зберемо схему.

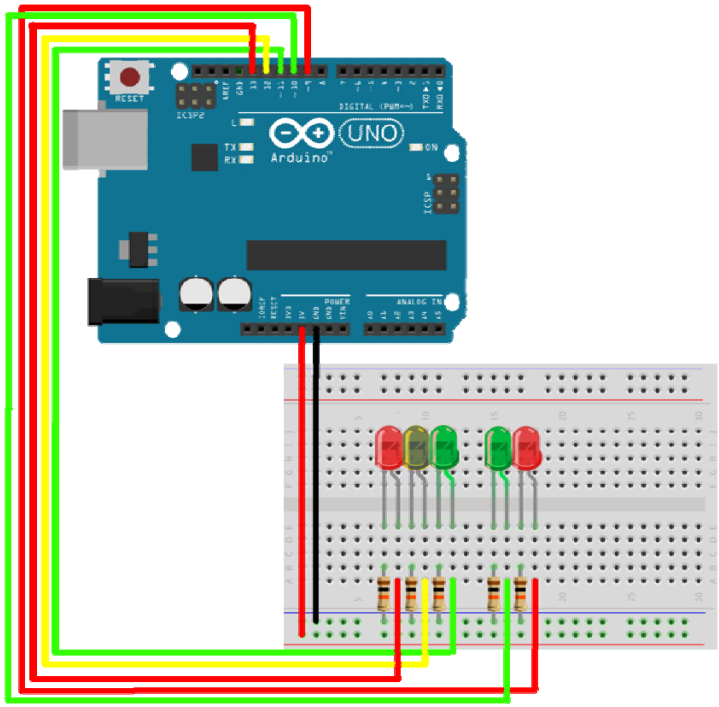


Рисунок 1.5 – Схема підключення діодів до плати Arduino UNO

Крок 2. У середовищі IDE Arduino напишемо наступний код:

Лістинг 1.4 – Програма для керування моделю світлофора

```
#define RED_1 13
#define YELLOW_1 12
#define GREEN_1 11
#define RED_2 9
#define GREEN_2 10
void setup() {
pinMode(RED_1, OUTPUT);
pinMode(YELLOW_1, OUTPUT);
pinMode(GREEN_1, OUTPUT);
pinMode(RED_2, OUTPUT);
pinMode(GREEN_2, OUTPUT);
}
```

```
void loop() {
digitalWrite(RED_1, HIGH);
digitalWrite(GREEN_2, HIGH);
delay(5000);
digitalWrite(RED_1, LOW);
digitalWrite(YELLOW_1, HIGH);
delay(1000);
digitalWrite(YELLOW_1, LOW);
digitalWrite(GREEN_2, LOW);
digitalWrite(GREEN_1, HIGH);
digitalWrite(RED_2, HIGH);
delay(5000);
digitalWrite(GREEN_1, LOW);
digitalWrite(YELLOW_1, HIGH);
delay(2000);
digitalWrite(YELLOW_1, LOW);
digitalWrite(RED_2, LOW);
}
```

1.4.4 Додамо в цю систему управління

Ситуація: пішохідний перехід яким рідко користуються, щоб не переривати марно дорожній рух, до системи додають управління, якщо пішохід натискає на кнопку, система світлофорів починає перемикання, пропускає його і повертається в режим очікування.

Крок 1. Зберемо схему.

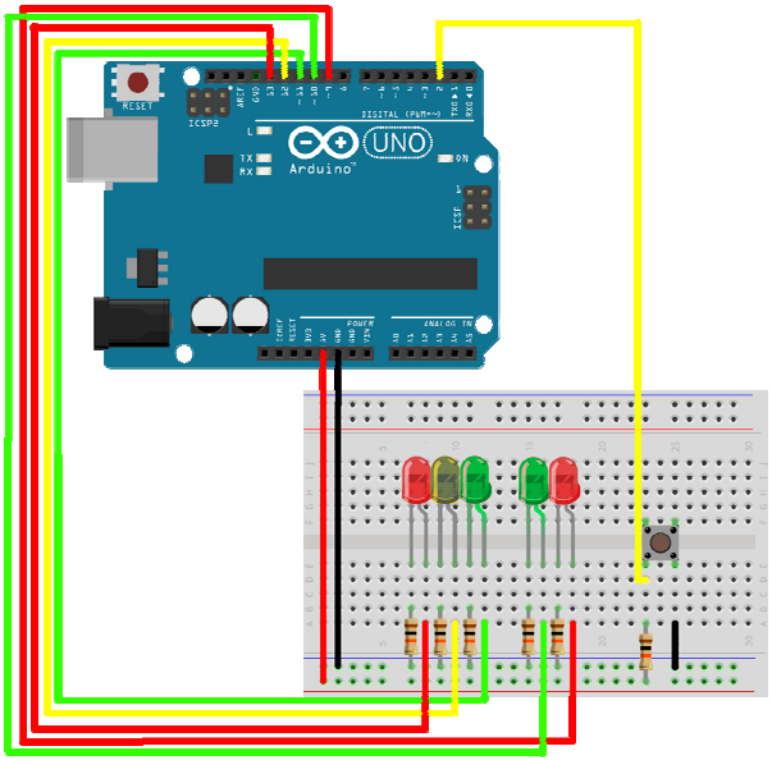


Рисунок 1.6 – Схема підключення діодів до плати Arduino UNO

Крок 2. У середовищі IDE Arduino напишемо наступний код:

Лістинг 1.5 – Програма для керування моделю світлофора

```
#define RED_1 13
#define YELLOW_1 12
#define GREEN_1 11
#define RED_2 9
#define GREEN_2 10
#define BUTTON 2
void setup() {
pinMode(RED_1, OUTPUT);
pinMode(YELLOW_1, OUTPUT);
pinMode(GREEN_1, OUTPUT);
pinMode(RED_2, OUTPUT);
```

```
pinMode(GREEN_2, OUTPUT);
}
void loop() {
if(!digitalRead(BUTTON))
{
digitalWrite(YELLOW_1, HIGH);
delay(1000);
digitalWrite(YELLOW_1, LOW);
digitalWrite(GREEN_2, HIGH);
digitalWrite(RED_2, LOW);
digitalWrite(RED_1, HIGH);
delay(4000);
digitalWrite(RED_1, LOW);
digitalWrite(YELLOW_1, HIGH);
delay(1000);
digitalWrite(YELLOW_1, LOW);
digitalWrite(GREEN_2, LOW);
}
else
{
digitalWrite(GREEN_1, HIGH);
digitalWrite(RED_2, HIGH);
}
}
```

1.4.5 Підвищення відмовостійкісті

У реалізації попередніх завдань є недолік, уявімо ситуацію, коли людина підходить до стовпа і постійно навмисно натискає на кнопку, або присутній постійний потік людей, для того щоб позбутися від ситуації завжди відображувати проїзду машин в програму потрібно додати блоки затримки.

Лістинг 1.6 – Програма для керування моделю світлофора

```
#define RED_1 13
#define YELLOW_1 12
#define GREEN_1 11
#define RED_2 9
```

```
#define GREEN_2 10
#define BUTTON 2
void setup() {
pinMode(RED_1, OUTPUT);
pinMode(YELLOW_1, OUTPUT);
pinMode(GREEN_1, OUTPUT);
pinMode(RED_2, OUTPUT);
pinMode(GREEN_2, OUTPUT);
}
void loop() {
if(!digitalRead(BUTTON))
{
  delay(1000);
  for(int i=0;i<3;i++)
  {
  digitalWrite(GREEN_1, HIGH);
  delay(500);
  digitalWrite(GREEN_1, LOW);
  delay(500);
    }
digitalWrite(YELLOW_1, HIGH);
delay(1000);
digitalWrite(YELLOW_1, LOW);
digitalWrite(GREEN_2, HIGH);
digitalWrite(RED_2, LOW);
digitalWrite(RED_1, HIGH);
delay(4000);
digitalWrite(RED_1, LOW);
digitalWrite(YELLOW_1, HIGH);
delay(1000);
digitalWrite(YELLOW_1, LOW);
digitalWrite(GREEN_2, LOW);
delay(5000);
}
else
{
digitalWrite(GREEN_1, HIGH);
digitalWrite(RED_2, HIGH);
```

```
}
}
```

1.5 Контрольні запитання

1 Що таке Arduino UNO?

2. Що таке Arduino IDE?;

3. Який контролер використовується на платі Arduino UNO?

4. Яка мова програмування використовується для написання програм у Arduino IDE?

5. Якими пристроями оснащена плата Arduino UNO?

6. Перахуйте можливі канали зв'язку Arduino UNO з іншими пристроями в т.ч. з комп'ютером.

2 ЛАБОРАТОРНА РОБОТА №2 "РОБОТА З COM–ПОРТОМ ТА ПЕРЕФЕРИЧНИМИ ПРИСТРОЯМИ"

Мета роботи – ознайомитися з принципом взаємодії із пристроями за допомогою послідовного порту.

2.1 Короткі теоретичні відомості

2.1.1 Потенціометр

Потенціометр ще називають змінний резистор. Це такий вид резистора, опір якого можна регулювати вручну. У потенціометра три ноги: перший харчування, другий вихід (визначає положення), третій земля. На малюнку 2.1 показаний зовнішній вигляд потенціометра.

Рисунок 2.1 – Потенціометр

2.1.2 Знайомство з COM-портом

Більшість мікроконтролерів мають безліч портів введення-виведення. Для зв'язку з ПК найбільш придатний з них протокол UART. Це протокол послідовної асинхронної передачі даних. Для його перетворення в інтерфейс USB на платі є конвертор USB-RS232 - FT232RL.

Для перевірки роботи програми необхідно, щоб на комп'ютері була термінальна програма, яка приймає дані з COM-порту. В Arduino IDE вже вбудована така. Для її виклику виберіть в меню Сервіс> Монітор порту. Вікно цієї утиліти дуже просте:

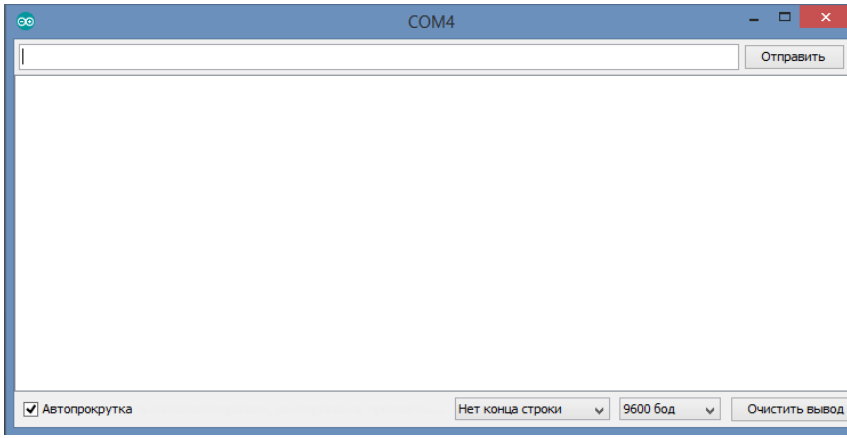


Рисунок 2.2 – Вікно монітора послідовного порту

Для запуску робота порту UART служить функція Serial.begin(). Єдиний її параметр − це швидкість. Про швидкості необхідно домовлятися на передавальної і приймальні стороні заздалегідь, так як протокол передачі асинхронний. У розглянутому прикладі швидкість 9600біт / с.

Для запису значення в порт використовуються три функції:

– Serial.write() – записує в порт дані в двійковому вигляді.
– Serial.print() може мати багато значень, але всі вони служать для виведення інформації в зручній для людини формі. Наприклад, якщо інформація, зазначена як параметр для передачі, виділена лапками - термінальна програма виведе її без зміни. Якщо ви хочете вивести якесь значення в певній системі числення, то необхідно додати службове слово: BIN-двоичная, OCT - восьмерична, DEC - десяткова, HEX - шістнадцяткова. Наприклад, Serial.print (25, HEX).
– Serial.println() робить те ж, що і Serial.print(), але ще переводить рядок після виведення інформації.

2.2 Короткі теоретичні відомості

2.2.1 Відправка команд з ПК

Перш ніж цим займатися, необхідно отримати уявлення щодо того, як працює COM-порт.

В першу чергу весь обмін відбувається через буфер пам'яті. Тобто коли ви відправляєте щось з ПК пристрою, дані поміщаються в певний спеціальний розділ пам'яті. Як тільки пристрій готовий - воно вичитує дані з буфера. Перевірити стан буфера дозволяє функція Serial.avaliable(). Ця функція повертає кількість байт в буфері. Щоб прочитати ці байти необхідно скористатися функцією Serial.read(). Розглянемо роботу цих функцій на прикладі:

Лістинг 2.1 – Пиклад роботи з послідоаним портом

```
int val = 0;

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

```
  if (Serial.available() > 0) {
    val = Serial.read();
    Serial.print("I received: ");
    Serial.write(val);
    Serial.println();
  }
}
```

Після того, як код буде завантажений в пам'ять мікроконтролера, відкрийте монітор COM-порту. Введіть один символ і натисніть Enter. В поле отриманих даних ви побачите: "I received: X", де замість X буде введений вами символ.

Програма нескінченно крутиться в основному циклі. У той момент, коли в порт записується байт функція Serial.available() приймає значення 1, тобто виконується умова Serial.available()> 0. Далі функція Serial.read() вичитує цей байт, тим самим очищаючи буфер. Після чого за допомогою вже відомих вам функцій відбувається висновок.

Використання вбудованого в Arduino IDE монітора COM-порту має деякі обмеження. При відправці даних з плати в COM-порт висновок можна організувати в довільному форматі. А при відправці з ПК до плати передача символів відбувається відповідно до таблиці ASCII. Це означає, що коли ви вводите, наприклад символ "1", через COM-порт відправляється в двійковому вигляді "00110001" (тобто "49" в десятковому вигляді).

Трохи змінимо код і перевіримо це твердження:

Лістинг 2.2 – Пиклад роботи з послідоаним портом

```
int val = 0;

void setup() {
  Serial.begin(9600);
}
void loop() {
  if (Serial.available() > 0) {
    val = Serial.read();
    Serial.print("I received: ");
    Serial.println(val,BIN);
```

```
    }
}
```

Після завантаження, в моніторі порту при відправці "1" ви побачите в відповідь: "I received: 110001". Можете змінити формат виведення і переглянути, що приймає плата при інших символах.

2.2.2 Управління пристроєм через COM-порт (частина 1)

Очевидно, що за командами з ПК можна керувати будь-якими функціями мікроконтролера. Завантажте програму, яка керує роботою світлодіода:

Лістинг 2.3 – Управління діодом через послідовний порт

```
#define LED 13
int val = 0;
void setup() {
        Serial.begin(9600);
        pinMode(LED,OUTPUT);
}
void loop() {
  if (Serial.available() > 0) {
    val = Serial.read();
    if (val=='H')
    {
      digitalWrite(LED,HIGH);
      Serial.print("The LED is ON\n");
      }
    if (val=='L')
    {
      digitalWrite(LED,LOW);
      Serial.print("The LED is OFF\n");
    }
  }
}
```

При відправці в COM-порт символу "H" відбувається запалювання світлодіода на тринадцятому виведення, а при відправці "L" світлодіод буде гаснути.

Якщо за результатами прийому даних з COM-порту ви хочете, щоб програма в основному циклі виконувала різні дії, можна виконувати перевірку умов в основному циклі.

Наприклад:

Лістинг 2.4 – Управління діодом через послідовний порт

```
#define LED 13
int val = '0';
void setup() {
        Serial.begin(9600);
        pinMode(LED,OUTPUT);
}
void loop() {
  if (Serial.available() > 0)
  {
    val = Serial.read();
    if (val=='1') {
      Serial.print("The diode flashes at a
frequency of 5 Hz\n");
    }
    if (val=='0') {
      Serial.print("The diode flashes at a
frequency of 1 Hz\n");
    }
    }
    if (val=='1') {
      digitalWrite(LED,HIGH); delay (100);
      digitalWrite(LED,LOW); delay (100);
    }
    if (val=='0') {
      digitalWrite(LED,HIGH); delay (500);
      digitalWrite(LED,LOW); delay (500);
    }
}
```

Якщо в моніторі порту відправити значення "1" світлодіод буде блимати з частотою 5Гц. Якщо відправити "0" - частота зміниться на 1 Гц.

2.2.3 Управління пристроєм через COM-порт (частина 2)

Самостійно зберіть схему і напишіть програму для керування запалюванням і гасінням 3 світлодіодів будь-якого кольору, що підключені до будьяких виводів.

2.2.4 Повернення рядки відправленої через COM-порт

Для коректної роботи програми потрібно встановити в налаштуваннях терміналу додавання символу кінця рядка.

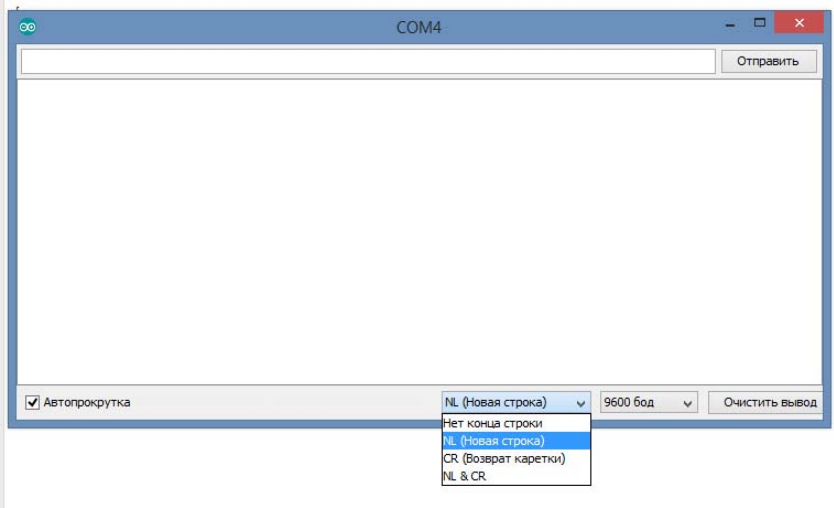


Рисунок 2.3 – Вікно монітора послідовного порту

Пошук по документації Arduino IDE дає два варіанти, що таке string. Це сам string як рядок char'ів, і String, що є об'єктом. Що таке об'єкт? Згідно Вікіпедії, це "деяка сутність у віртуальному просторі, що володіє певним станом і поведінкою, що має задані значення властивостей (атрибутів) і операцій над ними (методів)". Іншими словами - змінна з вбудованими функціями, що роблять щось з цією змінною. Щоб почати працювати з цим об'єктом, напишемо що-небудь такого виду:

Лістинг 2.5 – Спілкування з Arduino UNO через послідовний порт

```
String message;
```

```
char incomingChar = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
while(Serial.available()&&(incomingChar!=10)) {
       incomingChar = Serial.read();
       message += incomingChar;
     }

if(incomingChar==10){
Serial.print("I received: " + message);
message = "";
incomingChar = 0;
}
}
```

Дана програма спочатку зчитує символи надходять через послідовний інтерфейс, а потім повертає повідомлення, модифікуючи його шляхом додавання "I received:" в початок.

2.2.5 Управління пристроєм через COM-порт (частина 3) за допомогою команд

Напишемо програму обробки рядка, її завдання полягає в тому щоб порівняти перше слово з потрібним, і якщо так то видати друге слово назад а послідовний порт, якщо немає то видати відповідне повідомлення:

Лістинг 2.6 – Спілкування з Arduino UNO через послідовний порт

```
String message, data;
char incomingChar = 0;
void setup()
{
  Serial.begin(9600);
}
```

```
void loop()
{
while(Serial.available()&&(incomingChar!=10)) {
      incomingChar = Serial.read();
      message += incomingChar;
    }
if(incomingChar==10){
if(message.startsWith("START")){
data = message.substring(5);
data.trim();
Serial.println(data);
}
else
{
  Serial.println("Incorrect command");
  }
message = "";
incomingChar = 0;
}
}
```

Код використовує дуже зручні функції, вбудовані в об'єкт String. Це startsWith(), яка повертає одиницю, якщо рядок починається з того, що записано в дужках, substring(), яка повертає шматок рядка, що починається в даному випадку з 5-го символу (вважається, починаючи з нуля), trim(), який відкидає все зайве по краях рядка.

Про trim() варто написати окремо. Він потрібний не тільки для того, щоб відкинути пробіл на початку рядка, що вийшов, але в першу чергу - щоб позбутися від символів в її кінці. Це службові символи, що додаються при відправці повідомлення - NL (новий рядок) і CR (повернення каретки). Вони потрібні як раз для того, щоб сигналізувати про кінець команди, але можуть і перешкоджати. Тому, незважаючи на те, що в моніторі порту можна вибрати, які з цих символів посилати або не посилати нічого, краще перестрахуватися. Тим більше, що робиться це в один рядок коду.

А ось і список функцій (методів) об'єкта String.

charAt() - повертає символ, що стоїть на зазначеному місці

concat() - функція конкатенації, тобто злиття двох рядків в одну. Правда string1 = string1 + string2 це те ж саме, що і string1.concat(string1, string2), а записується простіше і зрозуміліше.

equals() - повертає одиницю, якщо рядок посимвольний дорівнює тому, що написано в дужках. Є ще equalsIgnoreCase(), який ігнорує регістр (верхній або нижній)

endsWith() - який працює аналогічно startsWith()

indexOf() - повертає місце в рядку символу (або рядка) в дужках. Шукає з кінця і повертає -1, якщо не знайдено.

length() - видає довжину рядка

setCharAt() - вимагає місце і символ, який треба поставити на це місце, наприклад: string1.setCharAt (3, 'd') поставить d третім символом в рядку замість того, що там стояло.

2.2.6 Управління пристроєм через COM-порт (частина 4) управління світлодіодом

Напишемо програму обробки рядка, її завдання полягає в тому щоб порівняти перше слово з потрібним, управляти статусом діода.

Лістинг 2.7 – Керування Arduino UNO через послідовний порт

```
#define LED 13
String message, data;
char incomingChar = 0;
void setup()
{
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}
void loop()
{
while(Serial.available()&&(incomingChar!=10)) {
       incomingChar = Serial.read();
       message += incomingChar;
    }
if(incomingChar==10){
if(message.startsWith("LED")){
data = message.substring(3);
```

```
data.trim();
if(data.startsWith("ON"))digitalWrite(LED,HIGH);
if(data.startsWith("OFF"))digitalWrite(LED,LOW);
}
else
{
  Serial.println("Incorrect command");
  }
message = "";
incomingChar = 0;
}
}
```

2.2.7 Управління пристроєм через COM-порт (частина 5). Управління світлодіодом

Зібрати схему підключивши світлодіод до 9-го вивода, написати програму

Лістинг 2.8 – Керування Arduino UNO через послідовний порт

```
#define LED_PIN 9
// для работы с текстом существуют объекты-
строки (англ. string)
String message;
void setup()
{
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  // передаваемые с компьютера данные
поставляются байт за
  // байтом, в виде отдельных символов (англ.
character). Нам
  // нужно последовательно их обрабатывать пока
(англ. while)
```

```
    // в порту доступны (англ. available) новые
данные
    while (Serial.available()) {
      // считываем (англ. read) пришедший символ в
переменную
      char incomingChar = Serial.read();
      //Serial.print(incomingChar,HEX);
      // не стоит путать целые числа и символы.
Они соотносятся
      // друг с другом по таблице, называемой
ANSII. Например
      // '0' — это 48, '9' — 57, 'A' — 65, 'B' —
66 и т.п. Символы
      // в программе записываются в одинарных
кавычках
      if (incomingChar >= '0' && incomingChar <=
'9') {
        // если пришёл символ-цифра, добавляем его
к сообщению
        message += incomingChar;
        } else if (incomingChar == 10) {
          Serial.print(message);
        // если пришёл символ новой строки, т.е.
enter, переводим
        // накопленное сообщение в целое число
(англ. to integer).
        // Так последовательность символов '1',
'2', '3' станет
        // числом 123. Результат выводим на
светодиод
        analogWrite(LED_PIN, message.toInt());
        // обнуляем накопленное сообщение, чтобы
начать всё заново
        message = "";
      }
    }
    // посылайте сообщения-числа с компьютера
через Serial Monitor
```

}

2.2.8 Передача цифрового значення датчика через COM порт

Зібрати схему:

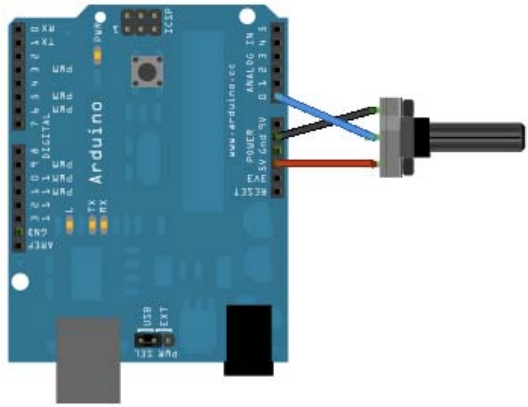


Рисунок 2.4 –Схема підключення потенціометра

Написати скетч и загрузити його в плату:

Лістинг 2.9 – Отримання данних з Arduino UNO через послідовний порт

```
void setup() {
 Serial.begin(9600);
}
void loop() {
 int sensorValue = analogRead(A0);
 Serial.println(sensorValue);
  delay(1);
}
```

2.2.9 Передача значення напруги через COM порт

Написати скетч и загрузити його в плату:

Лістинг 2.9 – Отримання данних з Arduino UNO через послідовний порт

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

```
}
void loop() {
int sensorValue = analogRead(A0);
float voltage = sensorValue * (5.0 / 1023.0);
Serial.println(voltage);
}
```

2.3 Контрольні запитання

1 Принцип передачі данних по послідовному порту.

2 Як здійснити конфігурацію COM–порту на платі Arduino UNO?

3 Функції роботи зі строками.

4 Алгоритм передачі через послідовний порт одного символу.

5 Алгоритм передачі через послідовний порт строки.

6 Керування Arduino UNO через послідовний порт.

3 ЛАБОРАТОРНА РОБОТА №3 „КЕРУВАННЯ ARDUINO ЧЕРЕЗ COM-ПОРТ”

Мета роботи – ознайомитися можливостями керування Arduino через COM-порт.

3.1 Короткі теоретичні відомості

Сервопривід (серводвигун або сервомотор) − це різновид мотора, для якого можна легко задавати кут повороту осі. Фактично це мотор, який має додаткові елементи для управління, зворотний зв'язок і обмежений кут повороту. У серводвигунів встановлений редуктор (набір зубчастих коліс), що визначає зусилля і швидкість обертання осі. Також в сервомоторами присутній датчик, який визначає положення вихідного вала і відповідно кут повороту серводвигуна. На малюнку 3.1 представлені складові частини серводвигуна.

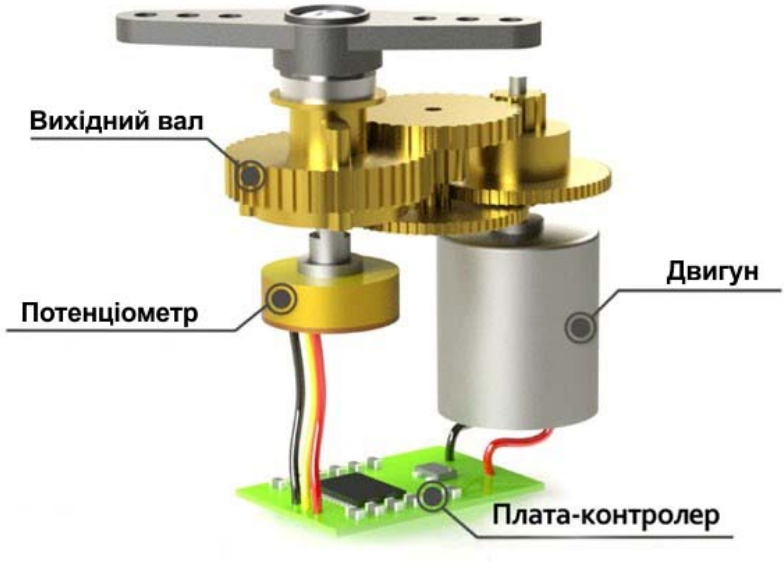


Рисунок 3.1 – Структура сервопривода

Серводвигуни можна використовувати в тих випадках, коли необхідно точно регулювати положення, наприклад, маніпулятора.

Зовнішній вигляд сервоприводу представлений на малюнку 3.2.

Рисунок 3.2 – Сервопривод

Від сервоприводу відходять три дроти: коричневий - земля, червоний - плюс, помаранчевий - сигнальний (підключається до цифрового виходу).

3.2 Хід роботи

3.2.1 Управління пристроєм через COM-порт (частина 1) за допомогою команд

Напишемо програму обробки рядка, її завдання полягає в тому щоб порівняти перше слово з потрібним, і якщо так то видати друге слово назад а послідовний порт, якщо немає то видати відповідне повідомлення:

Лістинг 3.1 – Управління Arduino UNO через послідовний порт

```
String message, data;
char incomingChar = 0;
void setup()
{
  Serial.begin(9600);
}
void loop()
{
while(Serial.available()&&(incomingChar!=10)) {
      incomingChar = Serial.read();
      message += incomingChar;
    }
if(incomingChar==10){
if(message.startsWith("START")){
data = message.substring(5);
data.trim();
Serial.println(data);
}
else
{
  Serial.println("Incorrect command");
  }
message = "";
incomingChar = 0;
```

```
}
}
```

Код використовує дуже зручні функції, вбудовані в об'єкт String. Це startsWith(), яка повертає одиницю, якщо рядок починається з того, що записано в дужках, substring(), яка повертає шматок рядка, що починається в даному випадку з 5-го символу (вважається, починаючи з нуля), trim(), який відкидає все зайве по краях рядка.

Про trim() варто написати окремо. Він потрібний не тільки для того, щоб відкинути пробіл на початку вийшла рядки, але в першу чергу - щоб позбутися від символів в її кінці. Це службові символи, що додаються при відправці повідомлення - NL (новий рядок) і CR (повернення каретки). Вони потрібні як раз для того, щоб сигналізувати про кінець команди, але можуть і перешкодити. Тому, незважаючи на те, що в моніторі порту можна вибрати, які з цих символів посилати або не посилати нічого, краще перестрахуватися. Тим більше, що робиться це в один рядок коду.

А ось і список функцій (методів) об'єкта String.

charAt() - повертає символ, що стоїть на зазначеному місці

concat() - функція конкатенації, тобто злиття двох рядків в одну. Правда string1 = string1 + string2 це те ж саме, що і string1.concat (string1, string2), а записується простіше і зрозуміліше.

equals() - повертає одиницю, якщо рядок посимвольний дорівнює тому, що написано в дужках. Є ще equalsIgnoreCase(), який ігнорує регістр (верхній або нижній)

endsWith() - який працює аналогічно startsWith()

indexOf() - повертає місце в рядку символу (або рядка) в дужках. Шукає з кінця і повертає -1, якщо не знайдено.

length() - видає довжину рядка

setCharAt() - вимагає місце і символ, який треба поставити на це місце, наприклад: string1.setCharAt (3, 'd') поставить d третім символом в рядку замість того, що там стояло.

3.2.2 Управління пристроєм через COM-порт (частина 2) управління світлодіодом

Напишемо програму обробки рядка, її завдання полягає в тому щоб порівняти перше слово з потрібним, управляти статусом діода.

Лістинг 3.2 – Управління Arduino UNO через послідовний порт

```
#define LED 13
String message, data;
char incomingChar = 0;
void setup()
{
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
}
void loop()
{
while(Serial.available()&&(incomingChar!=10)) {
      incomingChar = Serial.read();
      message += incomingChar;
    }
if(incomingChar==10){
if(message.startsWith("LED")){
data = message.substring(3);
data.trim();
if(data.startsWith("ON"))digitalWrite(LED,HIGH);
if(data.startsWith("OFF"))digitalWrite(LED,LOW);
}
else
{
  Serial.println("Incorrect command");
  }
message = "";
incomingChar = 0;
}
}
```

3.2.3 Управління пристроєм через COM-порт (частина 3) управління світлодіодом

Зібрати схему підключивши світлодіод до 9-му висновку написати програму

Лістинг 3.3 – Управління Arduino UNO через послідовний порт

```
#define LED_PIN 9
// для работы с текстом существуют объекты-строки (англ. string)
String message;
void setup()
{
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
}
void loop()
{
  // передаваемые с компьютера данные поставляются байт за
  // байтом, в виде отдельных символов (англ. character). Нам
  // нужно последовательно их обрабатывать пока (англ. while)
  // в порту доступны (англ. available) новые данные
  while (Serial.available()) {
    // считываем (англ. read) пришедший символ в переменную
    char incomingChar = Serial.read();
    //Serial.print(incomingChar,HEX);
    // не стоит путать целые числа и символы. Они соотносятся
    // друг с другом по таблице, называемой кодировкой. Например
    // '0' — это 48, '9' — 57, 'A' — 65, 'B' — 66 и т.п. Символы
    // в программе записываются в одинарных кавычках
    if (incomingChar >= '0' && incomingChar <= '9') {
      // если пришёл символ-цифра, добавляем его к сообщению
      message += incomingChar;
```

```
            } else if (incomingChar == 10) {
              Serial.print(message);
            // если пришёл символ новой строки, т.е.
enter, переводим
            // накопленное сообщение в целое число
(англ. to integer).
            // Так последовательность символов '1',
'2', '3' станет
            // числом 123. Результат выводим на
светодиод
            analogWrite(LED_PIN, message.toInt());
            // обнуляем накопленное сообщение, чтобы
начать всё заново
            message = "";
          }
        }
        // посылайте сообщения-числа с компьютера
через Serial Monitor
    }
```

3.2.4 Повертати серводвигун за і проти годинникової стрілки на 180 градусів

Крок 1. Зберемо схему.

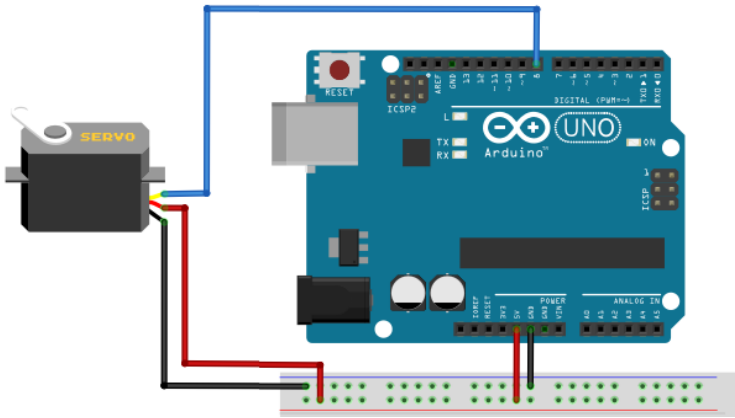


Рисунок 3.3 – Схема підключення серводвигуна

Крок 2. Складемо програму. Спочатку створимо змінну «кут», в якій буде зберігатися поточний стан серводвігателя. Її значення буде змінюватися спочатку від 0 до 180, а потім від 180 до 0.

Лістинг 3.4 – Управління серводвигуном за допомогою Arduino UNO

```
#include <Servo.h>
#define servoPin 8
Servo servo;
int angle = 0; // угол сервы в градусах
void setup()
{
servo.attach(servoPin);
}

void loop()
{
// инкремент от 0 до 180 градусов
for(angle = 0; angle < 180; angle++)
{
servo.write(angle);
delay(15);
}
// теперь в обратном направлении от 180 до 0
градусов
for(angle = 180; angle > 0; angle--)
{
servo.write(angle);
delay(15);
}
}
```

3.2.5 Керувати кутом повороту сервоприводу за допомогою потенціометра

Крок 1. Зберемо схему.

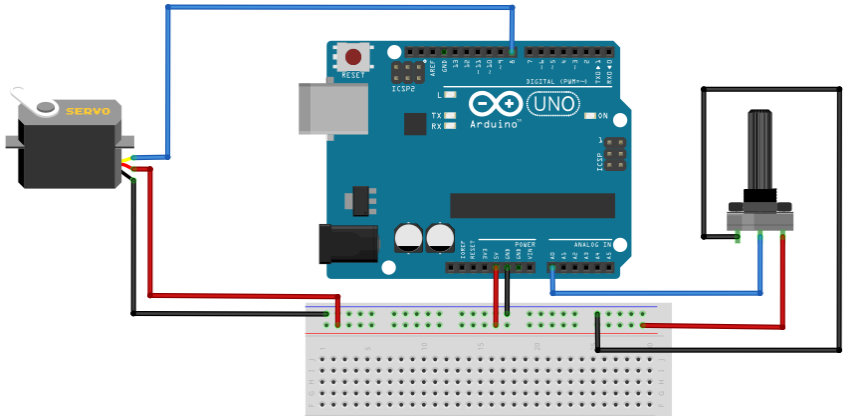


Рисунок 3.4 – Схема підключення серводвигуна та потенціометра

Крок 2. Складемо програму. Обчислюємо кут повороту потенціометра в залежності від значення, отриманого з потенціометра. Встановлюємо сервопривід на розрахунковий кут.

Лістинг 3.5 – Управління серводвигуном за допомогою Arduino UNO

```
#include <Servo.h>
int potPin = 0;
#define servoPin 8
Servo servo;
void setup()
{
servo.attach(servoPin);
}
void loop()
{
int reading = analogRead(potPin); // от 0 до
1023
int angle = reading / 6; // от 0 до 180
servo.write(angle);
}
```

3.2.6 Керувати кутом повороту сервоприводу за допомогою потенціометра

Шаг 1. Соберем схему.

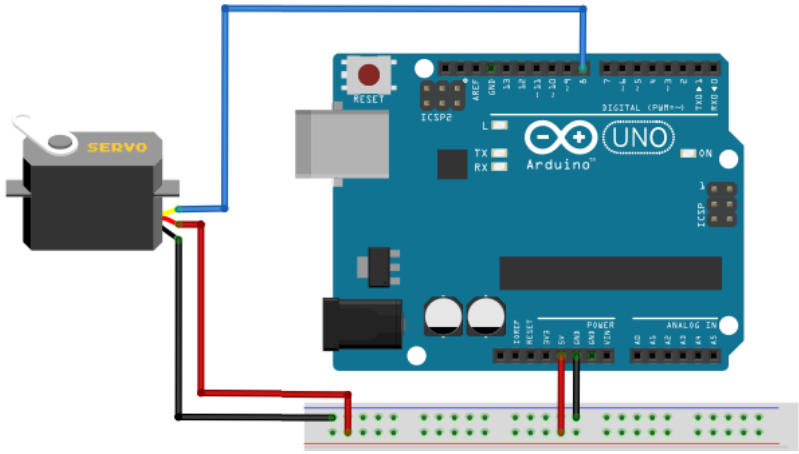


Рисунок 3.5 – Схема підключення серводвигуна

Шаг 2. Составим программу.

Лістинг 3.6 – Управління серводвигуном за допомогою Arduino UNO з компёютера через послыдовний порт

```
#include <Servo.h>
#define servoPin 8
Servo servo;
String message, data;
char incomingChar = 0;
int angle = 0; // угол сервы в градусах
void setup()
{
  Serial.begin(9600);
  servo.attach(servoPin);
}
void loop()
{
while(Serial.available()&&(incomingChar!=10))
{
        incomingChar = Serial.read();
```

```
          message += incomingChar;
        }
    if(incomingChar==10){
    if(message.startsWith("SET ANGLE")){
    data = message.substring(9);
    data.trim();
    int angle = data.toInt();
    if((angle!=0))
    {
      if((angle>0)&&(angle<180))
      {
      Serial.println("Angle is set at "+ data + "
degrees");
      servo.write(angle);
      }
      else
      {
        Serial.println("Incorrect angle");
        }
      }
    else
    {
      Serial.println("Incorrect command");
      }
    }
    else
    {
      Serial.println("Incorrect command");
      }
    message = "";
    incomingChar = 0;
    }
    }
```

Далі управляємо сервоприводом за допомогою відправки в COM-порт відповідних команд в форматі "SET ANGLE" + кут в градусах, значення кута помилково лежати в межах від 0 до 180 градусів

3.3 Контрольні запитання

1 Що таке серводвигун?

2. Які бібліотеки використовуються у Arduino IDE для роботи з серводвигуном?;

3. Функції роботи зі строками.

4. Чим серводвигун відрізняється від інших двигунів?

5. Управління Arduino UNO через послідовний порт.

6. Перахуйте можливі канали зв'язку Arduino UNO з іншими пристроями в т.ч. з комп'ютером.

4 ЛАБОРАТОРНА РОБОТА №4 "РОБОТА З LCD-ІНДИКАТОРАМИ НА БАЗІ КОНТРОЛЕРА HD44780"

Мета роботи – ознайомитися із загальними принципами організації системи виводу інформації в мікропроцесорних системах за допомогою сучасних знакосинтезуючих дисплеїв на базі контролеру HD44780.

4.1 Організація індикації у МПС на базі HD44780

Контролер HD44780 виробництва Hitachi є найбільш поширеним контролером управління алфавітно-цифровим модулем. Майже всі провідні виробники - Epson, Sanyo, Toshiba, Samsung, Philips випускають аналоги цього контролера або сумісні з ним по інтерфейсу і командній мові мікросхеми або РКІ на базі цих контролерів. Практично HD44780 є промисловим стандартом. Модулі з цим контролером застосовують у вимірювальних приладах, промисловому, технологічному і медичному устаткуванні, офісній техніці.

Контролер підтримує розмір символу 5x7 точок і 5x10 точок. HD44780 може управляти двома рядками по 40 символів.

Підключення до МК здійснюється так. Контролер з'єднується з РКІ через паралельну синхронну шину (8 або 4 ліній даних -

вибирається програмно), через лінію вибору операцій (R/W), лінію вибору регістра (RS), лінію стробування і синхронізації (E) (рис. 4.1).

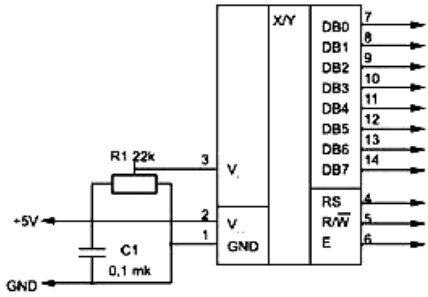


Рисунок 4.1 – Інтерфейс підключення контролера HD44780

Підстроювальний резистор R1 плавно змінює напругу живлення РКІ, що дозволяє виставляти необхідну контрастність індикатора при необхідному куті огляду. Модуль може під'єднуватися за допомогою 4- або 8-мирозрядної шини (рис. 4.2). При цьому обмін можна організувати або з системною шиною, або через порти вводу-виводу програмними засобами.

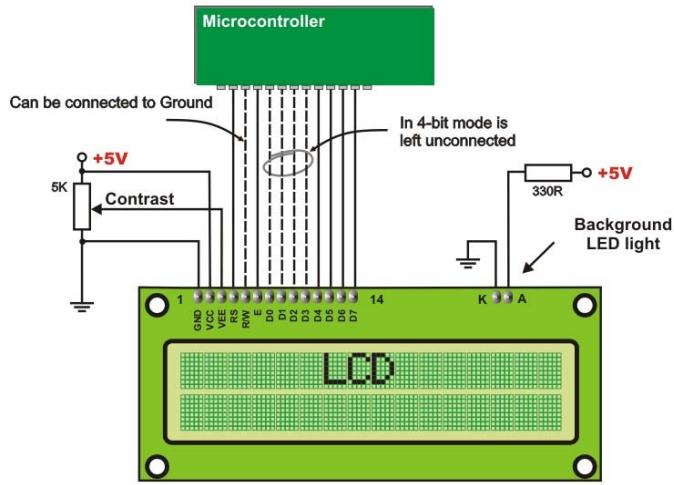


Рисунок 4.2 – Схема підключення РКІ до МК

На часовій діаграмі на рис. 4.3 показані стани керуючих сигналів і шини даних під час читання і запису.

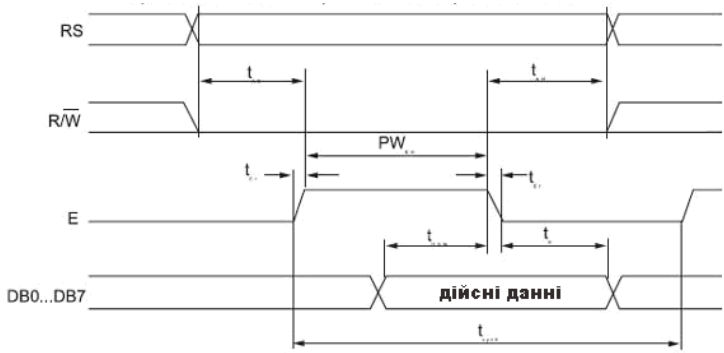


Рисунок 4.3 – Часова діаграма запису даних у РКІ

У вихідному стані E=0, R/W=0 значення сигналу RS довільне, шина даних DB0...DB7 перебуває в стані високого імпедансу. У проміжках між операціями обміну E і R/W також мають дорівнювати 0, у цей момент шина даних вільна і може використовуватися в мультиплексному режимі для інших цілей. У табл.4.1 наведено послідовність дій при виконанні операцій читання-запису. Час виконання кожного кроку не менше 250 нс.

Таблиця 4.1 – Послідовність дій при читання та запису

Операції запису для 8-мі розрядної шини	
1	Установка значення лінії RS
2	Виведення значення байта даних на шину DB0...DB7
3	Установка лінії E=1
4	Установка лінії E=0
5	Установка шини DB0...DB7 у стан HI
Операції читання для 8-мі розрядної шини	
1	Установка значення лінії RS
2	Установка лінії R/W=1
3	Установка лінії E=1
4	Зчитування байта даних з шини DB0...DB7
5	Установка лінії E=0
6	Установка лінії R/W=0
Операції запису для 4-х розрядної шини	

1	Установка значення лінії RS
2	Виведення значення старшої тетради байта даних DB4...DB7
3	Установка лінії E=1
4	Установка лінії E=0
5	Виведення значення молодшої тетради байта даних DB4...DB7
6	Установка лінії E=1
7	Установка лінії E =0
8	Установка шини DB4...DB7 в стан HI
Операції читання для 4-х розрядної шини	
1	Установка значення лінії RS
2	Установка лінії R/W=1
3	Установка лінії E=1
4	Зчитування значення старшої тетради байта даних DB4...DB7
5	Установка лінії E=0
6	Установка лінії E=1
7	Зчитування значення молодшої тетради байта даних DB4...DB7
8	Установка лінії E=0
9	Установка лінії R/W=0

Описані операції читання-запису байта є базовими для здійснення обміну даними з РКІ. Процес обміну по 4-х і 8-мі розрядною шиною розрізняється лише реалізацією цих операцій. Ці дві операції можуть бути реалізовані апаратно, коли модуль підключений до системної шини, або програмно, коли він взаємодіє з портами МК.

У таблиці 4.2 наведені значення часових інтервалів сигналів.

Таблиця 4.2 – Часові характеристики

Параметр	Позн.	Мін., нс	Макс., нс
Операція читання			
Період сигналу E	t_{cycE}	500	-
Позитивний напівперіод сигналу E	PW_{EH}	230	-
Фронт/спад сигналу E	t_{Ef}, t_{Er}	-	20
Встановлення адреси	t_{AS}	40	-
Утримання адреси	t_{AH}	10	-
Встановлення даних	t_{DSW}	80	-
Утримання даних	t_{DSH}	10	-
Операція запису			

Період сигналу E	t_{cycE}	500	-
Позитивний напівперіод сигналу E	PW_{EH}	230	-
Фронт/спад сигналу E	t_{Ef}, t_{Er}	-	20
Встановлення адреси	t_{AS}	40	-
Утримання адреси	t_{AH}	10	-
Встановлення даних	t_{DSW}	-	160
Утримання даних	t_{DHW}	5	-

4.1.1 Програмування і управління LCD на базі HD4470

На рис. 4.4 наведено основні елементи контролера HD4470, які безпосередньо взаємодіють з керуючою програмою - регістр даних (DR), регістр команд (IR), відеопам'ять (DDRAM), ОЗП знакогенератора (CGRAM), лічильник адреси пам'яті (AC), прапор зайнятості контролера.

Управляється контролер через керуючий інтерфейс системи. Основні об'єкти взаємодії - регістри DR і IR. Вибір регістра, що адресується, здійснюється лінією RS, якщо RS=0 - адресується регістр команд (IR), якщо RS=1 - регістр даних (DR). Дані через регістр DR можуть розміщатися або зчитуватися у відеопам'ять (DDRAM), чи в ОЗП знакогенератора (CGRAM) за поточною адресою, на яку вказує лічильник адреси (AC). Інформація в регістрі IR інтерпретується пристроєм виконання команд як керуюча послідовність. Зчитування регістра IR повертає в 7-мі молодших розрядах поточне значення лічильника AC, а в старшому розряді прапор зайнятості (BF). Відеопам'ять має загальний об'єм 80 байтів і призначена для зберігання коду символів.

У таблиці 4.3 показаний набір різних прапорів для HD44780. Прапори визначають режими роботи різних елементів контролера. У таблиці 4.4 наведені значення управляючих прапорів, безпосередньо після подачі на РКІ-модуль напруги живлення. Перевизначення значень прапорів виробляється спеціальними командами, записуваними в регістр IR, при цьому комбінації старших бітів визначають групу прапорів або команду, а молодші містять власне прапори.

Таблиця 4.3 – Управляючі прапори контролера HD44780

I/D	Режим зсуву лічильника адреси AC, 0 - зменшення, 1 - збільшення
S	Прапор режиму зсуву вмісту екрану. 0 – зсуву немає, 1 - після запису в DDRAM чергового коду екран зсувається в напрямі, який визначається прапором I/D: 0 - вправо, 1 - вліво. При зсуві не змінюється вміст DDRAM. змінюються лише внутрішні покажчики розташування видимого початку рядка в DDRAM
S/C	Прапор-команда, яка виробляє разом з прапором R/L операцію зсуву вмісту екрану (так само, як і у попередньому випадку, без змін в DDRAM) або курсору. Визначає об'єкт зсуву: 0 - зсувається курсор, 1 - зсувається екран
R/L	Прапор-команда, яка виробляє разом з прапором S/C операцію зсуву екрану або курсору. Уточнює напрям зсуву: 0 - вліво, 1 - вправо
D/L	Прапор, що визначає ширину шини даних: 0 - 4 розряди, 1 - 8 розрядів
N	Режим розгортки зображення на ЖКИ: 0 - один рядок, 1 - два рядки
F	Розмір матриці символів: 0 - 5 x 8 точок, 1 - 5 x 10 точок
D	Наявність зображення: 0 - вимкнено, 1 - включено
C	Курсор у вигляді підкреслення: 0 - вимкнений, 1 - включений
B	Курсор у вигляді мерехтливого знакомісця: 0 - вимкнений, 1 - включений

Таблиця 4.4 – Значення прапорів після подачі живлення

I/D=1	Режим збільшення лічильника на 1
S=0	Без зсуву зображення
D/L=1	8-мирозрядна шина даних
N=0	Режим розгортки одного рядка
F=0	Символи з матрицею 5 x 8 точок
D=0	Відображення вимкнене
C = 0	Курсор у вигляді підкреслення вимкнений
B = 0	Курсор у вигляді мерехтливого знакомісця вимкнений

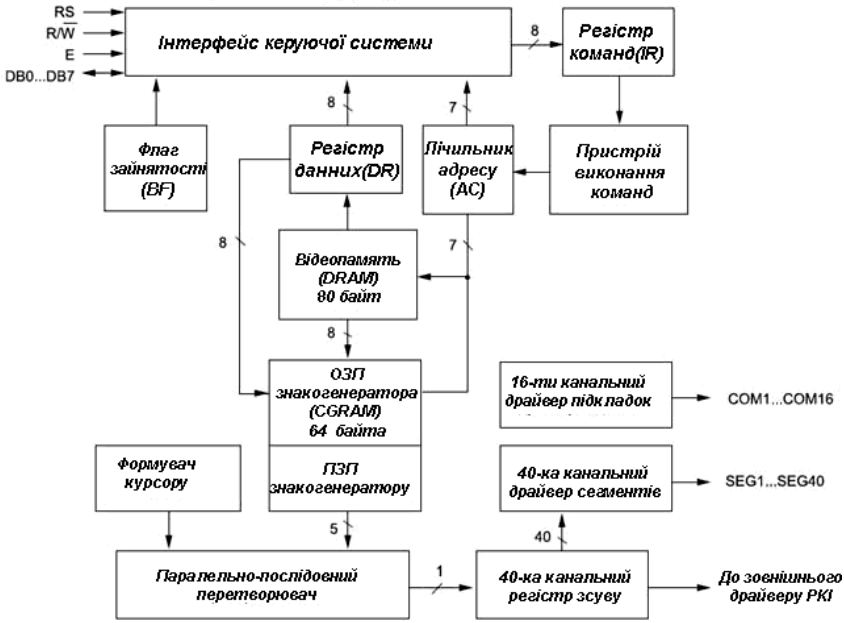


Рисунок 4.4 –Спрощена структурна схема контролера HD4470

Список комбінацій бітів регістра IR і виконувані ними управляючі команди наведені в табл.4.5. Оскільки на момент включення РКІ-модуль нічого не відображує (прапор D=0), то для того, щоб вивести будь-який текст, необхідно, як мінімум, включити відображення, встановивши прапор D=1. Наприклад, широко поширена послідовність для ініціалізації модуля: $OC, 6 (знак $ перед числом вказує на шістнадцяткове представлення) встановлює режим відображення 2-х рядків з матрицею 5 х 8 точок і робота з 8-мирозрядною шиною даних; $OC включає відображення на екрані модуля без відображення курсорів; 6 встановлює режим автоматичного переміщення курсору зліва-направо після виведення кожного символу.

Таблиця 4.5 - Комбінації бітів регістра IR

Призначення	D7	D6	D5	D4	D3	D2	D1	D0
Очищення екрану, AC = 0, адресація AC на DDRAM	0	0	0	0	0	0	0	1
AC=0, адресація на DDRAM, скинуті зсуви, початок рядка адресується на початку DDRAM	0	0	0	0	0	0	1	-
Вибирається напрям зсуву курсору чи екрану	0	0	0	0	0	1	I/D	S
Вибирається режим відображення	0	0	0	0	1	D	C	B
Команда зсуву курсору/екрану	0	0	0	1	S/C	R/L	-	-
Визначення параметрів розгортки і ширини шини даних	0	0	1	DL	N	F	-	-
Присвоєння лічильнику AC адреси в області CGRAM	0	1	AG	AG	AG	AG	AG	AG
Присвоєння лічильнику AC адреси в області DDRAM	1	AD	AD	AD	AD	AD	AD	AD

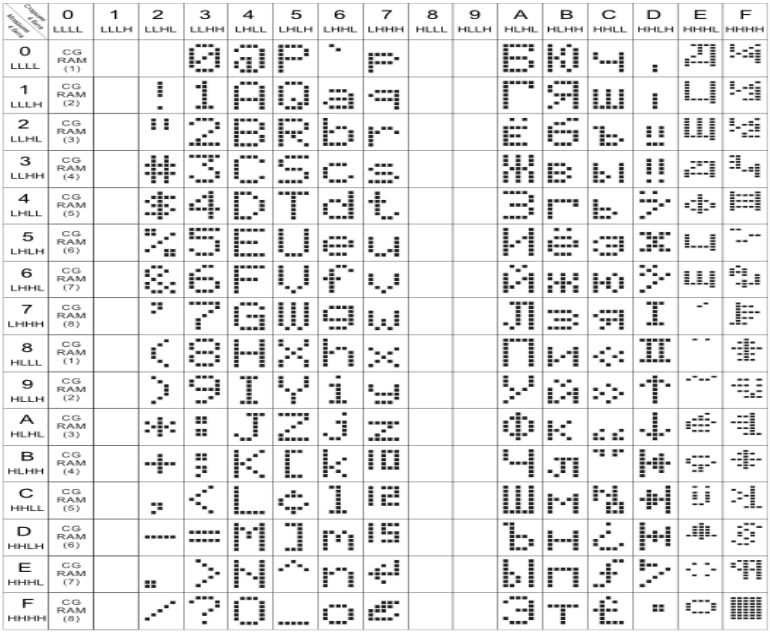
Контролер HD44780 підтримує як операції запису, так і операції читання. Читання регістра DR призводить до завантаження вмісту DDRAM або CGRAM залежно від поточного режиму, при цьому курсор зміщується на одну позицію, як і при записі. Читання регістра IR повертає 8 значущих розрядів, причому в 7-ми молодших міститься поточне значення лічильника AC (7 розрядів, якщо адресується DDRAM, і 6 - якщо CGRAM), а в старшому - прапор зайнятості BF. Цей прапор має значення 1 коли контролер зайнятий і 0 - коли вільний. Необхідно враховувати, що більшість операцій, які виконуються контролером, займають значний час, близько 40 мкс, а час виконання деяких доходить до одиниць мілісекунд, тому цикл чекання зняття прапора BF має бути обов'язково присутнім в програмах драйвера РКІ-модуля і передувати здійсненню будь-якої операції (природно, окрім операції перевірки прапора BF).

Після здійснення операції запису або читання DDRAM і появи після неї ознаки готовності (BF=0), прочитане в цьому ж циклі (разом з прапором BF) значення AC швидше за все не буде достовірним, оскільки між появою ознаки готовності і обчисленням контролером нового значення ACA існує деякий часовий інтервал близько 4 мкс при тактовій частоті контролера 270 кГц. Тому якщо необхідно набути

дійсного значення АСА, потрібно зробити повторну операцію прочитання IR після не менше ніж 4 мкс (якщо контролер працює на частоті 270 кГц час чекання необхідно пропорційно збільшити).

Виведення на екран символу здійснюється шляхом запису його коду (табл. 4.6) в регістр DR. Наприклад, код символу *0* за таблицею є *$30*. При цьому символ розміщується в DDRAM за поточною адресою, що вказується в АС, а значення АС збільшується чи зменшується на 1.

Таблиця 4.6 – Коди символів для відображення на екрані



Виробник контролера рекомендує виконувати наступну послідовність дій для ініціалізації. Витримати паузу не менше 15 мс між встановленням робочої напруги живлення (більше 4,5 В) і виконанням будь-яких операцій з контролером. Першою операцією виконати команду, яка вибирає розрядність шини (це має бути команда незалежно від того, якої розрядності інтерфейс ви збираєтеся використовувати надалі), причому перед виконанням цієї операції не перевіряти значення прапора BF. Далі знову витримати паузу не менше 4,1 мс і повторити команду вибору розрядності шини, причому перед поданням команди знову не перевіряти прапор BF. Наступний

крок - знов витримати паузу, цього разу 100 мкс, і втретє повторити команду встановлення розрядності шини, знов без перевірки BF. Ці три операції призначені для ініціалізації і покликані вивести контролер у вихідний режим роботи (тобто перевести в режим роботи з 8-мирозрядною шиною) з будь-якого стану. Слідом за ними нормальним порядком (без витримки пауз, але з перевіркою прапора BF) виконується ініціалізація режимів роботи з видачею послідовності для ініціалізації, аналогічній вказаній раніше (що містить у тому числі команду вибору необхідної розрядності шини).

Слід пам'ятати, що при виборі режиму роботи з 4-х розрядною шиною, тобто видачі команд, це звичайно відбувається з 8-ми розрядного режиму, який встановлюється автоматично після подачі напруги живлення, а отже ви не зможете адекватно оголосити необхідне значення прапорів N і F, розташованих в молодшій тетраді команди установки розрядності шини. Тому команду необхідно повторити у вже сталому 4-х розрядному режимі послідовною передачею двох тетрад.

4.2 Хід роботи

4.2.1 Вивести на екран текст: My name is ....

Крок 1. Зібрати схему.

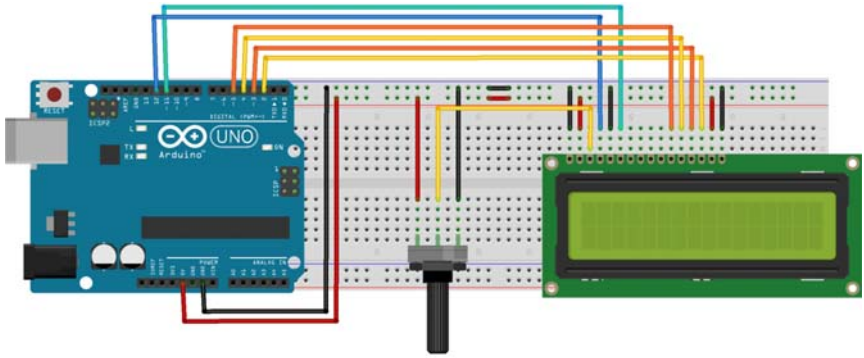


Рисунок 4.5 –Схема підключення контролера HD4470

Потенціометр використовується в схемі для регулювання контрастності дисплея.

Крок 2. Написати програму. Коментарі годі й переписувати.

Лістинг 4.1 – Ініціалізація контролера HD4470 платою Arduino UNO

```
/* Подключаем библиотеку для работы с LCD */
#include <LiquidCrystal.h>
/* Создаём объект LCD-дисплея, используя конструктор класса LiquidCrystal
* с 6ю аргументами. Библиотека по количеству аргументов сама определит,
* что нужно использовать 4-битный интерфейс.
* Указываем, к каким пинам Arduino подключены выводы дисплея:
*   RS, E, DB4, DB5, DB6, DB7
*/
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
 /* Инициализируем дисплей: 2 строки по 16 символов */
 lcd.begin(16, 2);
 /* Выводим на дисплей традиционную фразу */
 lcd.print("My name is ");
  /* Устанавливаем курсор в 1 столбец 2й строки. Нумерация идёт с нуля,
  * первым аргументом идёт номер столбца.
  */
 lcd.setCursor(0, 1);
 lcd.print("Ivan");/*вместо Ivan пишете свое имя в кавычках*/
}
void loop()
{
}
```

4.2.2 Вивести на екран текст: Hello world і час роботи програми

Схема залишається колишньою

Лістинг 4.2 – Програма керування контролером HD4470 платою Arduino UNO

```
#include <LiquidCrystal.h>
/*  Создаём  объект  LCD-дисплея,  используя
конструктор класса LiquidCrystal
* с 6ю аргументами. Библиотека по количеству
аргументов сама определит,
* что нужно использовать 4-битный интерфейс.
* Указываем, к каким пинам Arduino подключены
выводы дисплея:
*   RS, E, DB4, DB5, DB6, DB7
*/
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
 /* Инициализируем дисплей: 2 строки по 16
символов */
 lcd.begin(16, 2);
 /* Выводим на дисплей традиционную фразу */
 lcd.print("Hello, world!");
}
void loop()
{
 /*  Устанавливаем  курсор  в  1  столбец  2й
строки. Нумерация идёт с нуля,
  * первым аргументом идёт номер столбца.
  */
 lcd.setCursor(0, 1);
 /* Выводим на дисплей число секунд, прошедших
с момента старта Arduino */
 lcd.print(millis() / 1000);
}
```

4.2.3 Вивести на екран текст: Hello world і зворотній відлік часу від 1000с (САМОСТІЙНО).

На онові матеріалу передніх пунктів самостійно написати програму для виводиння на еклан тексту Hello world і зворотнього відліку часу від 1000 с.

4.2.4 Вивести на екран повідомлення прийняте з COM-порту. Схема залишається колишньою

Лістинг 4.3 – Програма керування контролером HD4470 платою Arduino UNO через послідовний порт

```
#include <LiquidCrystal.h>
String message;
char incomingChar = 0;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
  Serial.begin(9600);
  lcd.begin(16, 2);
}
void loop()
{
while(Serial.available()&&(incomingChar!=10))
{
      incomingChar = Serial.read();
      message += incomingChar;
    }
if(incomingChar==10){
message.trim();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print(message);
message = "";
incomingChar = 0;
}
}
```

Ця програма отримує дані з COM порту і виводить їх на LCD екран.

4.2.5 Вивести на екран повідомлення прийняте з COM-порту. Залежно від змісту. Схема залишається колишньою

Лістинг 4.4 – Програма керування контролером HD4470 платою Arduino UNO через послідовний порт

```
#include <LiquidCrystal.h>
String message, data;
char incomingChar = 0;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
void setup()
{
  Serial.begin(9600);
  lcd.begin(16, 2);
}
void loop()
{
while(Serial.available()&&(incomingChar!=10))
{
      incomingChar = Serial.read();
      message += incomingChar;
    }
if(incomingChar==10){
if(message.startsWith("line 1:")){
data = message.substring(7);
data.trim();
lcd.clear();
lcd.setCursor(0, 0);
lcd.print("                ");
lcd.setCursor(0, 0);
lcd.print(data);
message = "";
incomingChar = 0;
}
else if(message.startsWith("line 2:")){
data = message.substring(7);
data.trim();
lcd.setCursor(0, 1);
lcd.print("                ");
lcd.setCursor(0, 1);
lcd.print(data);
message = "";
incomingChar = 0;
}
```

```
else
{
  Serial.println("Incorrect command");
message = "";
incomingChar = 0;
  }
}
}
```

Ця програма отримує дані з COM порту і якщо віддана рядок починається з "line 1:" або "line 2:" то виводить на LCD то що йде після «:», в іншому випадку повертає в COM порт повідомлення про помилку.

4.2.6 Вивести значення температури з датчика

Крок 1. Зібрати схему.

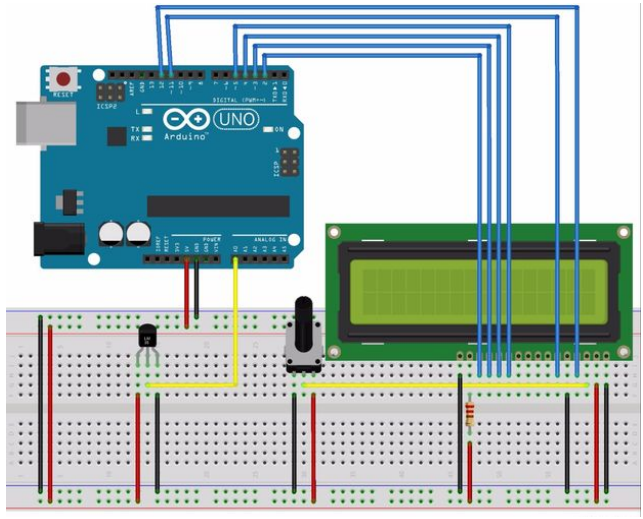


Рисунок 4.6 – Схема підключення контролера HD4470 та датчика температури

Крок 2. Написати програму. Коментарі можна не переписувати

Лістинг 4.5 – Програма для вимірювання температури та відображення її значення на рідкопристалічному індикаторі

```
#include <LiquidCrystal.h>
#define pinTemp A0
LiquidCrystal lcd(12,11,5,4,3,2);
float temperatuur;
float temp[30];
float som;
void setup() {
  lcd.begin(16,2);
  lcd.print("Temperatuur ");
  analogReference(INTERNAL);
  lcd.print(analogRead(pinTemp)/9.31);
}
void loop() {
  for(int i = 0; i<30;i++)
  {temp[i]=temperatuur;
  delay(20);
    }
  for(int i=0;i<30;i++){
      som=som+temp[i];
    }
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Temperatuur ");
    lcd.print(analogRead(pinTemp)/9.31);
    lcd.setCursor(0,0);
    som=0;
}
```

4.3 Контрольні запитання

1 Принцип побудови РКІ-модуля з контролером HD44780.
2 Алгоритм роботи по 4-х провідному інтерфейсу.
3 Алгоритм роботи по 8-ми провідному інтерфейсу.
4 Переваги і недоліки модулів з контролером HD44780.
5 Схема підключення РКІ-модуля до МК.

5 ЛАБОРАТОРНА РОБОТА №5 “КЕРУВАННЯ КРОКОВИМ ДВИГУНОМ”

Мета роботи – ознайомитися із загальними принципами керування кроковим двигуном.

5.1 Теоретичні відомості

5.1.1 Крокoвий двигун

Крокові двигуни - основа точної робототехніки. На відміну від двигунів постійного обертання, один оборот складається з безлічі мікропереміщень, які і називають кроками. Іншими словами, ми можемо повернути вал двигуна рівно на 90 градусів, і зафіксувати його в цьому положенні. Грубим аналогом крокової двигуна є серводвигун.

Принцип роботи такого механізму полягає в наступному: завдяки електронному комутатору виникають імпульси напруги, які згодом передаються на обмотки управляння, що розташовані на статорі КД. В залежності від послідовності сприйняття обмоток управляння, проходить певна дискретна зміна магнітного поля в робочому зазорі двигуна. Дискретне переміщення КД виконується завдяки спеціальній формі ротора та двох обмоток. У результаті чергувань направлень напруги в обмотках, можна досягнути того, що ротор буде по черзі займати фіксоване значення.

Зазвичай у крокового двигуна на 1 оберт валу, виходить біля 200 кроків. Кількість кроків може залежати від моделі та конструкції самого двигуна. Дискретний електропривід в дуеті з кроковим двигуном поєднуються з числовими пристроями, а це дозволяє успішно управляти верстатами ЧПУ (верстати з числовим програмним управлянням) та іншим обладнанням. Такий механізм як крокoвий двигун застосовується в електроприводах потужністю до декількох кіловат.

Зауважимо, що крокові двигуни бувають двох типів біополярні та уніполярні. Біополярні мають чотири виходи та дві обмотки. Уніполярні шість виходів та вміщають в себе дві обмотки, кожна з яких має відвід із середини рис 5.1.

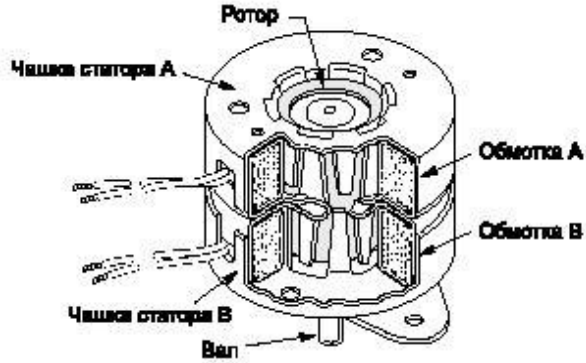


Рисунок 5.1 − схематичне зображення крокового двигуна

Кроковідвигуни застосовують там, де потрібна дуже точно дозувати переміщення. Найбільш очевидний приклад − робот маніпулятор. Щоб механічна рука торкнулася робочим інструментом потрібної точки, необхідно щоб кожен з вузлів повернувся на строго заданий кут. Похибка в частки градуса в підставі руки, призведе до величезної похибки на ефектори.

Іншим відомим прикладом може служити ЧПУ верстат. Взяти той же 3D-принтер. Для точного переміщення друкуючої головки застосовують саме крокові двигуни. У старих дисководах кроковиках використовувалися для переміщення магнітної головки. А в сучасних фотоапаратах мікро-мініатюрні крокові двигуни (рис 5.2) переміщують лінзи.

Рисунок 5.2 − Кроковий двигун 28BYj-48

За допомогою контролера Ардуіно Уно запустимо популярний крокový двигун 28BYj-48. Цей мініатюрний двигун має вбудований редуктор, який дозволяє здійснювати дуже точні переміщення вихідного валу.

Так, в 4-крокovому режимі двигун робить 2048 кроків за один оберт. У 8-крокової - 4096. Напруга живлення - 5 Вольт. Струм - 160мА. А значить, для експерименту нам буде достатньо штатного харчування від USB.

Як драйвера для двигуна використовуємо мікросхему ULN2003, яку часто продають в парі з 28BYj-48. Ось так виглядає плата драйвера рис 5.3:

Рисунок 5.3 – Драйвер для крокового двигуна

На платі є 4 входи для мікроконтролера: IN1..IN4. П'ять виходів на двигун, і два контакти харчування. Також є перемичка, що розриває ланцюг живлення двигуна.

5.1.2 Схема підключення

Як правило, провідники двигуна 28BYj-48 вже мають роз'єм з ключем, який вставляється в плату тільки в правильному положенні. В іншому випадку, при підключенні необхідно слідувати колірній схемі (див. Малюнок). Контакти IN1..IN4 можна підключити до будь-яких цифрових виходів Ардуіно Уно.

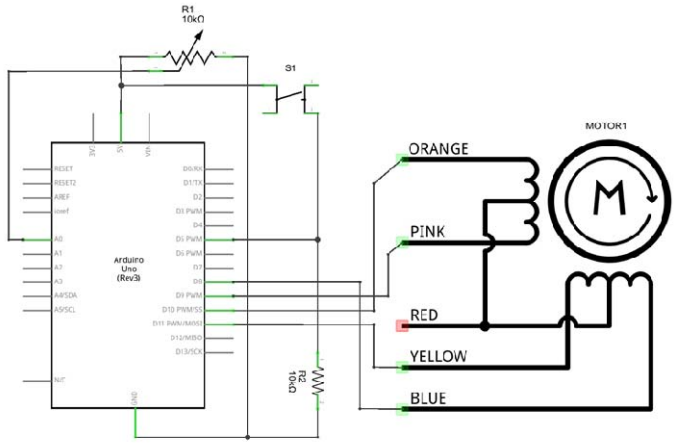


Рисунок 5.4 – Структурна схема підключення крокового двигуна

Схема підключення безпосередньо до ArduinoUNO

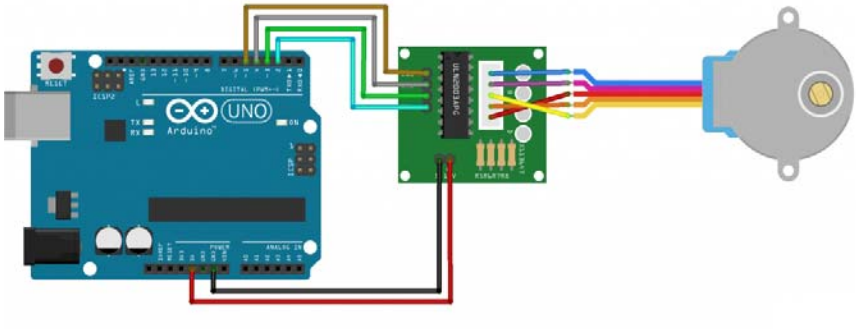


Рисунок 5.5 – Блок схема підключення крокового двигуна

5.1.3 Створення програмного коду

Для запуску крокового двигуна потрібно подати напругу на його обмотки у певній визначеній послідовності. Ми реалізуємо одну з декількох схем комутації. Для цього, звернемо увагу на таблицю комутації:

Таблиця 5.1 – Таблиця комутації для керування кроковим двигуном

Крок	A	B	A\	B\
0	1	1	0	0
1	0	1	0	0
2	0	1	1	0
3	0	0	1	0
4	0	0	1	1
5	0	0	0	1
6	1	0	0	1
7	1	0	0	0

У цій таблиці, колонка A відповідає котушці, керованої сигналом IN3. Колонка B - IN4. A \ і B \ - управляються через IN1 і IN2, відповідно.

Виходить така от нехитра програма. Мінлива dl в ній - це час, між сусідніми комутаціями.

5.1.4 Скетч запуску крокового двигну без використання бібліотек

Лістинг 5.1 – Програма для керівання кроковим двигуном

```
int in1 = 2;
int in2 = 3;
int in3 = 4;
int in4 = 5;
const int dl = 5;
void setup() {
    pinMode(in1,OUTPUT);
    pinMode(in2,OUTPUT);
    pinMode(in3,OUTPUT);
    pinMode(in4,OUTPUT);
}
void loop() {
    digitalWrite( in1,HIGH );
    digitalWrite( in2,HIGH );
    digitalWrite( in3,LOW );
```

```
    digitalWrite( in4,LOW );
    delay(dl);
    digitalWrite( in1,LOW );
    digitalWrite( in2,HIGH );
    digitalWrite( in3,HIGH );
    digitalWrite( in4,LOW );
    delay(dl);
    digitalWrite( in1,LOW );
    digitalWrite( in2,LOW );
    digitalWrite( in3,HIGH );
    digitalWrite( in4,HIGH );
    delay(dl);
    digitalWrite( in1,HIGH );
    digitalWrite( in2,LOW );
    digitalWrite( in3,LOW );
    digitalWrite( in4,HIGH );
    delay(dl);
}
```

Щоб змусити двигун рухатися швидше або повільніше, необхідно буде змінити змінну dl. Збільшуємо паузу між комутаціями - двигун обертається повільніше. Зменшуємо паузу - крутиться швидше.

5.2 Хід роботи

5.2.1 Керування сервоприводом зі зміною його кута повороту, а також з можливістю зміни швидкості

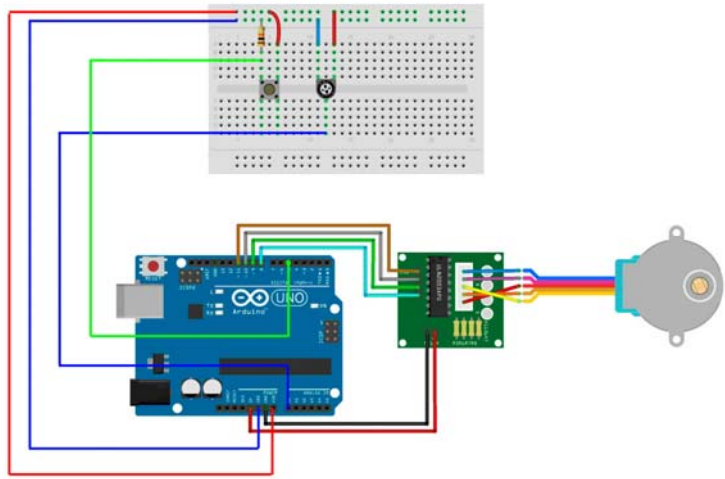


Рисунок 5.6 – Схема підключення

Лістинг 5.2 – Програма для керівання кроковим двигуном

```
int pins[] = {8, 9, 10, 11}; //Задаем пины по
порядку
int buttonPin = 5;
int potPin = A0;
int phases =4; // для шагового режима установить 4
bool motorPhases[8][4] = {
 {              1, 1, 0, 0},
  {              0, 1, 0, 0},
  {              0, 1, 1, 0},
  {              0, 0, 1, 0},
  {              0, 0, 1, 1},
  {              0, 0, 0, 1},
  {              1, 0, 0, 1},
  {              1, 0, 0, 0}
};
voidsetup() {
```

```
  for (int i = 0; i < 4; i++)
pinMode(pins[i],OUTPUT);
  pinMode(buttonPin,INPUT);
  pinMode(potPin,INPUT);
  //Serial.begin(9600);
}
int phase = 0;
int _step = 1; // Если у шага поменять знак, на -1
- изменится направление вращения.
voidloop() {
  int delaytime =map(analogRead(potPin), 0, 1023,
1000, 16000);
  bool isButtonDown =digitalRead(buttonPin);
  //Serial.println(isButtonDown);
  if (isButtonDown) {
    _step =-_step;
    delay(500); // Делаемпаузу, чтобыотсечдребезг
  }
  phase += _step;
  if (phase > 7) phase = 0;
  if (phase < 0) phase = 7;
  for (int i = 0; i < 4; i++) {
    digitalWrite(pins[i], ((motorPhases[phase][i]
== 1) ?HIGH:LOW));
  }
    delayMicroseconds(delaytime);
}
```

5.3 Контрольні запитання

1. Що таке кроковий двигун?
2. Чим кроковий двигун відрізняється від інших типів двигунів?
3. Як відбудається керування кроковим двигуном?
4 Як організувати керування кроковим двигуном за допомогою мікроконтроллера?

6 ЛАБОРАТОРНА РОБОТА №6 “ПІДКЛЮЧЕННЯ RFID МІТКИ ДО ARDUINO”

Мета роботи – ознайомитися із загальними принципами роботи RFID пристроїв за допомогою готових модулів Arduino.

6.1 Загальні теоретичні відомості

RFID (англ. Radio Frequency IDentification, радіочастотна ідентифікація) - спосіб автоматичної ідентифікації об'єктів, в якому за допомогою радіосигналу зчитуються або записуються дані, що зберігаються в так званих транспондерах, або RFID-мітках. Будь-яка RFID-система складається з пристрою, що зчитує (зчитувач, рідер або інтеррогатор) і транспондера (він же RFID-мітка, іноді також застосовується термін RFID-тег).

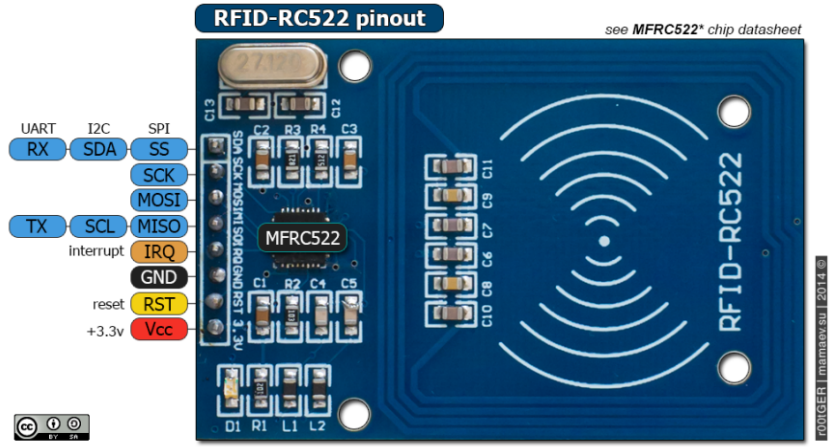


Рисунок 6.1 – Зчитувач RFID міток

Рисунок 6.2 – RFID мітка

Модуль RC має 8 висновків (написані по порядку розташування на модулі):

VCC - Живлення. Необхідно 3.3V;

RST - Reset. Лінія скидання. Ні в якому разі не підключати до піну RESET на CraftDuino! Даний пін чіпляється на цифровий порт з PWM;

GND - Ground. Земля =));

MISO - Master Input Slave Output - дані від веденого до ведучого, SPI;

MOSI - Master Output Slave Input - дані від ведучого до веденого, SPI;

SCK - Serial Clock - тактовий сигнал, SPI;

NSS - Slave Select - вибір веденого, SPI;

IRQ - лінія переривань.

6.2 Хід роботи

6.2.1 Зчитування коду карти (RFID мітки)

Схема підключення:

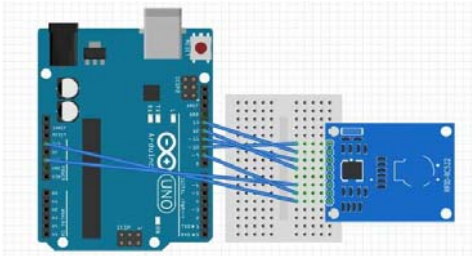


Рисунок 6.3 – Схема підключення

Лістинг 6.1 – Програма отримання RFID ідентифікатора ключа та відправка його через послідовний порт

```
#include <SPI.h>
#include <RFID.h>
#define SS_PIN 10
#define RST_PIN 9
RFID rfid(SS_PIN, RST_PIN);
// Данные о номере карты храняться в 5 переменных,
будем запоминать их, чтобы проверять, считывали ли
мы уже такую карту
    int serNum0;
    int serNum1;
    int serNum2;
    int serNum3;
    int serNum4;
void setup()
{
  Serial.begin(9600);
  SPI.begin();
  rfid.init();

}
void loop()
{
    if (rfid.isCard()) {
        if (rfid.readCardSerial()) { // Сравниваем
номер карты с номером предыдущей карты
            if (rfid.serNum[0] != serNum0
                && rfid.serNum[1] != serNum1
                && rfid.serNum[2] != serNum2
                && rfid.serNum[3] != serNum3
                && rfid.serNum[4] != serNum4
            ) {
                /* Если карта - новая, то
считываем*/
                Serial.println(" ");
                Serial.println("Card found");
                serNum0 = rfid.serNum[0];
```

```
                    serNum1 = rfid.serNum[1];
                    serNum2 = rfid.serNum[2];
                    serNum3 = rfid.serNum[3];
                    serNum4 = rfid.serNum[4];
                    Serial.println("Cardnumber:");
                    Serial.print("Dec: ");
  Serial.print(rfid.serNum[0],DEC);
                    Serial.print(", ");
  Serial.print(rfid.serNum[1],DEC);
                    Serial.print(", ");
  Serial.print(rfid.serNum[2],DEC);
                    Serial.print(", ");
  Serial.print(rfid.serNum[3],DEC);
                    Serial.print(", ");
  Serial.print(rfid.serNum[4],DEC);
                    Serial.println(" ");
                    Serial.print("Hex: ");
  Serial.print(rfid.serNum[0],HEX);
                    Serial.print(", ");
  Serial.print(rfid.serNum[1],HEX);
                    Serial.print(", ");
  Serial.print(rfid.serNum[2],HEX);
                    Serial.print(", ");
  Serial.print(rfid.serNum[3],HEX);
                    Serial.print(", ");
  Serial.print(rfid.serNum[4],HEX);
                    Serial.println(" ");
                } else {
                  /* Если это уже считанная карта,
просто выводим точку */
                  Serial.print(".");
                }
            }
    }
    rfid.halt();
}
```

У результаті отримуємо у вікні COM порту унікальній код мітки.

6.2.2 Організація доступу з індикацією

Необхідно виконати наступні умови:

1. Створити масив з номером відомої карти (див. пункт 1.1);
2. Вважати серійний номер і записати його в окремий масив;
3. Провести поелементне порівняння фіксований номер зі лічених;
4. В залежності від результату, виконати різні дії.

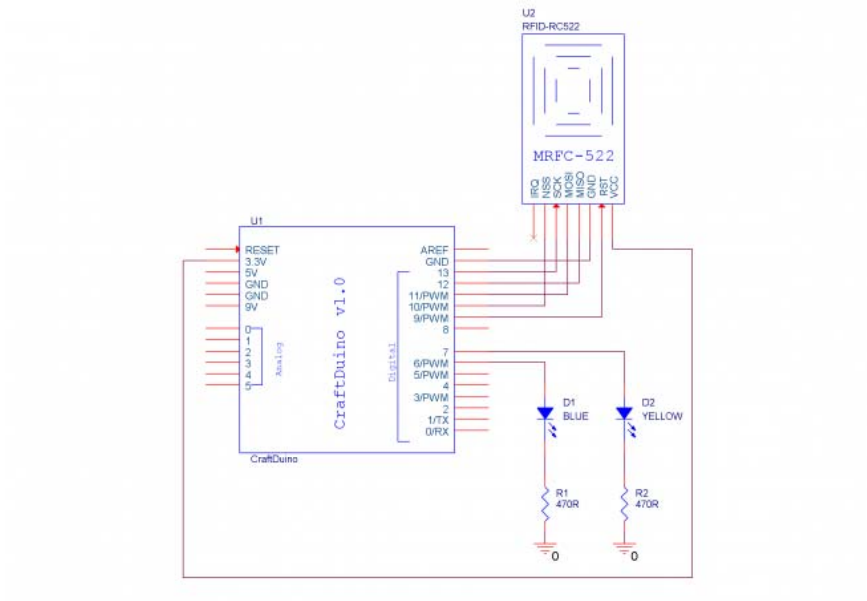


Рисунок 6.4 – Схема підключення

Лістинг 6.2 – Програма для керування системою ідентифікації

```
#include <SPI.h>
#include <RFID.h>
#define SS_PIN 10
#define RST_PIN 9
#define BLUE_LED 6
#define YELLOW_LED 7
RFID rfid(SS_PIN, RST_PIN);
unsigned char reading_card[5];
unsigned char master[5] = {114,13,207,204,124}; //
КОД КАРТИ
```

```
unsigned char i;
void indication(int led);
void allow();
void denied();
void setup()
{
  Serial.begin(9600);
  SPI.begin();
  rfid.init();
  pinMode(BLUE_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
}
void loop()
{
    if (rfid.isCard())
    {
        if (rfid.readCardSerial())
        {
                /* Reading card */
                Serial.println(" ");
                Serial.println("Card found");
                Serial.println("Cardnumber:");
                for (i = 0; i < 5; i++)
                {
                  Serial.print(rfid.serNum[i]);
                  Serial.print(" ");
                  reading_card[i] =
rfid.serNum[i];
                }
                Serial.println();
                //verification
                for (i = 0; i < 5; i++)
                {
                  if (reading_card[i]!=master[i])
                  {
                    break;
                  }
                }
```

```
                    if (i == 5)
                    {
                      allow();
                    }
                    else
                    {
                      denied();
                    }
            }
      }
      rfid.halt();
}

void allow()
{
  Serial.println("Access accept!");
  indication(BLUE_LED);
}
void denied()
{
  Serial.println("Access denied!");
  indication(YELLOW_LED);
}
void indication(int led)
{
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
}
```

Скетч 3 Вивід на LSD імені з подальшим доступом.

Підключення LCD дисплею (див. Лабораторну 4 підключення LCD через I2C).

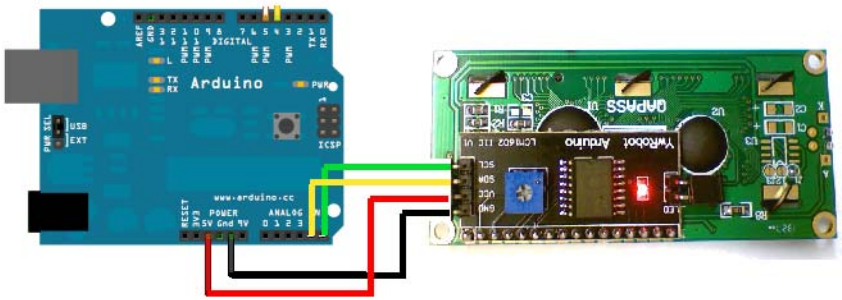


Рисунок 6.5 – Схема підключення рідкокристалічного дисплею до Arduino UNO

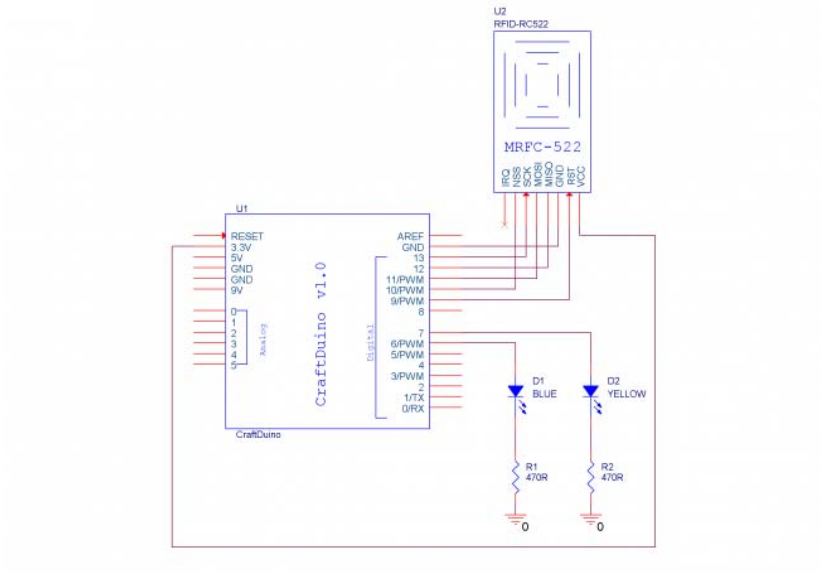


Рисунок 6.6 – Принципова схема підключення рідкокристалічного дисплею до Arduino UNO

Лістинг 6.2 – Програма для керування системою ідентифікації та контролю доступу

```
#include <SPI.h>
#include <RFID.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x3F,16,2);
#define SS_PIN 10
#define RST_PIN 9
#define BLUE_LED 6
#define YELLOW_LED 7
RFID rfid(SS_PIN, RST_PIN);
unsigned char reading_card[5];
unsigned char master[5] = {230, 190, 32, 126, 6};
// КОД КАРТИ
unsigned char i;
void allow();
void denied();
void setup()
{
  Serial.begin(9600);
  SPI.begin();
  rfid.init();
  pinMode(BLUE_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  lcd.init();
  lcd.backlight();// Включаем подсветку дисплея
  lcd.setCursor(3, 0);
  lcd.print("Reed  Card");
  lcd.setCursor(0, 1);
  lcd.print("********************");
}
void loop()
{
    if (rfid.isCard())
    {
        if (rfid.readCardSerial())
        {
               Serial.println(" ");
                Serial.println("Card found");
                Serial.println("Cardnumber:");
                for (i = 0; i < 5; i++)
                {
                  Serial.print(rfid.serNum[i]);
```

```
                    Serial.print(" ");
                    reading_card[i] =
rfid.serNum[i];
                  }
                  Serial.println();
                  //verification
                  for (i = 0; i < 5; i++)
                  {
                    if (reading_card[i]!=master[i])
                    {
                      break;
                    }
                  }
                  if (i == 5)
                  {
                    allow();
                  }
                  else
                  {
                    denied();
                  }
          }
              lcd.clear();
              lcd.backlight();
              lcd.setCursor(3, 0);
              lcd.print("Reed  Card");
              lcd.setCursor(0, 1);
              lcd.print("********************");
     }
     rfid.halt();
}
void allow()
{
  Serial.println("Access accept!");
  digitalWrite(BLUE_LED, HIGH);
  lcd.clear();
  lcd.backlight();
  lcd.print("Access accept");
```

```
  lcd.setCursor(1, 1);
  lcd.print("Name: Swat");
  delay(2000);
  digitalWrite(BLUE_LED, LOW);
}
void denied()
{
  Serial.println("Access denied!");
  digitalWrite(YELLOW_LED, HIGH);
  lcd.clear();
  lcd.backlight();
  lcd.print("Access denied!");
  lcd.setCursor(1, 1);
  lcd.print("Unknown: User");
  delay(2000);
  digitalWrite(YELLOW_LED, LOW);
}
```

6.3 Контрольні запитання

1. Що таке RFID?
2. Які сфери застосування RFID Ви можете назвати?
3. Які програмні засоби є в Arduino IDE для роботи з RFID?
4. Як організувати систему контролю доступу використовуючи RFID?
5. Перерахуйте функції, що використовувалися Вами для роботи з RFID та опишіть їх.

7 ЛАБОРАТОРНА РОБОТА №6 “ПІДКЛЮЧЕННЯ МОДУЛЯ ГОДИНИКА РЕАЛЬНОГО ЧАСУ DS1302 І LCD ЕКРАНА ДО КОНТРОЛЕРА ARDUINO”

Мета роботи – ознайомитися із загальними принципами роботи годиника реального часу DS1302 та його взаємодії з контролером Arduino UNO.

7.1 Загальні теоретичні відомості

Багато проектів або завдання вимагають точного тимчасового виконання. Наприклад, в системі автополиву можуть бути кілька режимів: ранковий полив, денний і вечірній. Значить, для стабільної роботи всієї цієї системи і всього робочого циклу необхідно, щоб система, побудована на Arduino, мала можливість точно визначати поточний час.

Ця функція в платі не була передбачена, тому таку проблему допоможе вирішити RTC модуль годин реального часу. Він допоможе скласти план для Arduino стосовно того, що і в який час має включатися в роботу або проводити інші дії.

Орієнтація в часі дуже корисна не тільки в автополиву, але і в інших системах: включення світла або опалення за розкладом, включення електрочайника за таймером та ін.

Розглянемо роботу модулів годин реального часу на прикладі DS1302. До цього сімейства також можна віднести і інші модулі, наприклад DS1307, DS3231, при цьому схема підключення і написання коду у всіх ідентичні.

7.1.1 Короткий опис модуля DS1302

DS1302 є однойменна мікросхема на невеликій платі з необхідною обв'язкою, що дозволяє серед іншого підключити батарейку на 3-5 В, завдяки чому живлення модуля можна здійснювати, як через Ардуіно, так і самостійно.

Мікросхема DS1302 містить годинник реального часу з календарем і 31 байт статичного ОЗУ. Вона спілкується з мікропроцесором через простий послідовний інтерфейс самостійно.

На що ж здатне цей пристрій?

Рисунок 7.1 - Модуль годинника реального часу DS1302

Дана мікросхема дозволяє рахувати час з точністю до секунд. Відрізняється низьким енергоспоживанням, тому маленької літієвої батарейки може вистачити більш ніж на місяць.

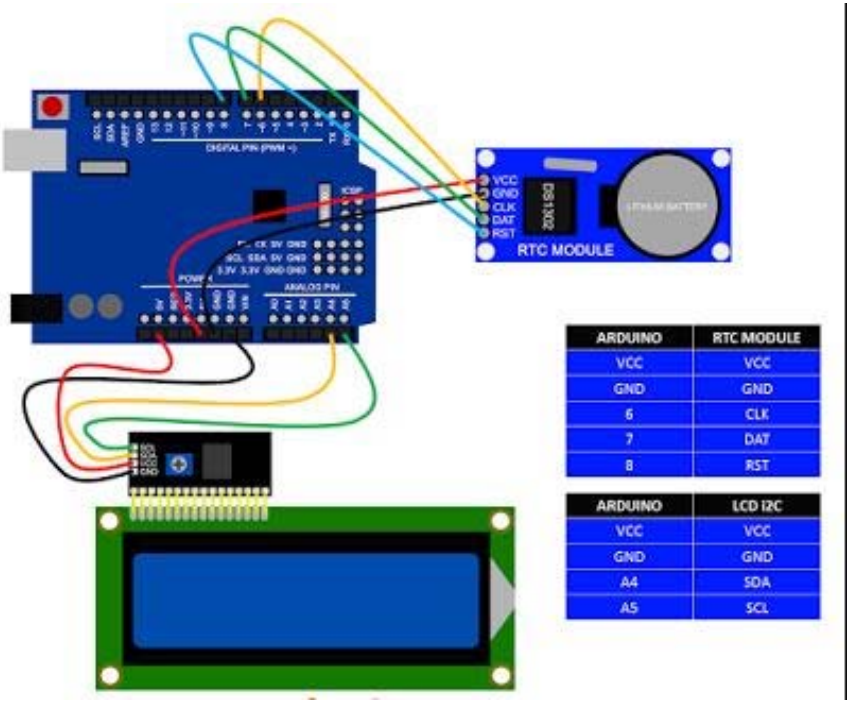
Інформація про реальному часі і календарі представляється в секундах, хвилинах, годинах, дні тижня, дати, місяць і рік. Якщо поточний місяць містить менше 31 дня, то мікросхема автоматично визначить кількість днів у місяці з урахуванням високосного поточного року. Годинники працюють або в 24-годинному або 12-годинному форматі з індикатором AM / PM (до полудня / після полудня). Підключення DS1302 до микропроцесорів спрощено за рахунок синхронного послідовного зв'язку. Для цього потрібно тільки 3 дроти: (1) RST (скидання), (2) I / O (лінія даних) і (3) SCLK (синхронізація послідовного зв'язку).

Дані можуть передаватися по одному байту або послідовністю байтів до 31. DS1302 розроблений, щоб споживати малу потужність і зберігати дані та інформацію годин при споживанні менше 1 мкВт. DS1302 - наступник DS1202. На додаток до основних функцій зберігання часу DS1202, DS1302 має два висновки живлення для підключення основного і резервного джерела живлення, можливість підключення програмованого ланцюга заряду до виводу VCC1 і сім додаткових байтів ОЗУ.

7.2 Хід роботи

7.2.1 Виведення на екран поточний часу і дати, заданого з комп'ютера

Крок 1. Зібрати схему.



ARDUINO	RTC MODULE
VCC	VCC
GND	GND
6	CLK
7	DAT
8	RST

ARDUINO	LCD I2C
VCC	VCC
GND	GND
A4	SDA
A5	SCL

Рисунок 7.2 - Схема підключення модуля DS1302 і LCD екрану

• Вивід VCC і Gnd відповідають відповідно за живлення (подача п'яти вольт на DS1302 і заземлення);

• Контакт CLK підключають до цифрового піну на платі, наприклад, до піну 6;

• Контакт DAT підключають до цифрового піну на платі, наприклад, до піну 7;

• Контакт RESX підключають до цифрового піну на платі, наприклад, до піну 8.

Крок 2. Написати програму. Коментарі годі й переписувати.

Якщо при компіляції НЕ знаходится бібліотеки то необхідно їх підключити

Для бібліотеки LiquidCrystal_I2C.h

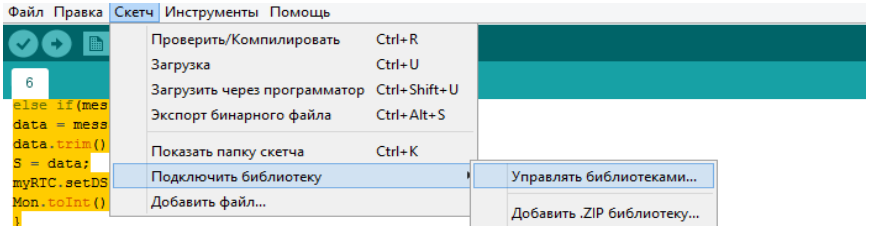


Рисунок 7.3 – Підключення бібліотеки у середовищі Arduino IDE/

В строку пошуку вбити LiquidCrystal_I2C та встановити відповідну бібліотеку

Для virtuabotixRTC.h

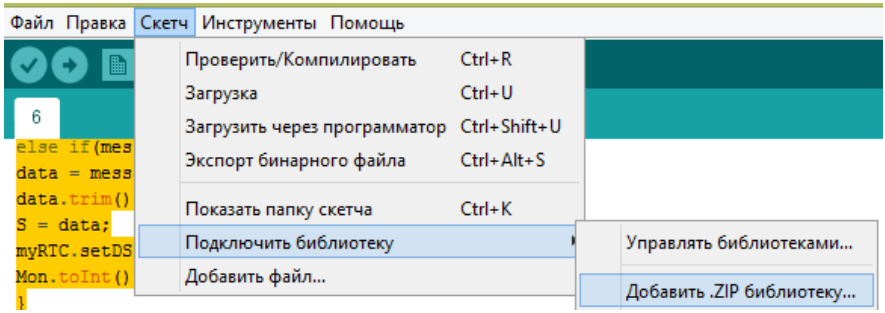


Рисунок 7.4 – Підключення бібліотеки у середовищі Arduino IDE/

Відповідний архів знаходиться в каталозі з лабораторною роботою

Лістинг 7.1 – Програма для керування годинником реального часу та виводом інформації з нього на рідкокристалічний монітор

```
#include<LiquidCrystal_I2C.h>
#include<Wire.h>
#include<virtuabotixRTC.h>
virtuabotixRTC myRTC(6, 7, 8); //CLK, DAT, RST
LiquidCrystal_I2C lcd(0x3F,16,2);
voidsetup() {
Serial.begin(9600);
lcd.init();
lcd.backlight(); // включениеподсветкидисплея
}
```

```
char incomingChar = 0;
String Y,H,M,S,D,Mon,message, data;
voidloop() {
while(Serial.available()&&(incomingChar!=10)) {
incomingChar =Serial.read();
message += incomingChar;}
if(incomingChar==10){
D = myRTC.dayofmonth;
Mon = myRTC.month;
H = myRTC.hours;
M = myRTC.minutes;
S = myRTC.seconds;
Y = myRTC.year;
if(message.startsWith("year:")){
data = message.substring(5);
data.trim();
Y = data;
myRTC.setDS1302Time(S.toInt(),M.toInt(),
H.toInt(), 06, D.toInt(),
Mon.toInt(), Y.toInt());
}
elseif(message.startsWith("month:")){
data = message.substring(6);
data.trim();
Mon = data;
myRTC.setDS1302Time(S.toInt(),M.toInt(),
H.toInt(), 06, D.toInt(),
Mon.toInt(), Y.toInt());
}
elseif(message.startsWith("day:")){
data = message.substring(4);
data.trim();
D = data;
myRTC.setDS1302Time(S.toInt(),M.toInt(),
H.toInt(), 06, D.toInt(),
Mon.toInt(), Y.toInt());
}
elseif(message.startsWith("hours:")){
```

```
data = message.substring(6);
data.trim();
H = data;
myRTC.setDS1302Time(S.toInt(),M.toInt(),
H.toInt(), 06, D.toInt(),
Mon.toInt(), Y.toInt());
}
elseif(message.startsWith("min:")){
data = message.substring(4);
data.trim();
M = data;
myRTC.setDS1302Time(S.toInt(),M.toInt(),
H.toInt(), 06, D.toInt(),
Mon.toInt(), Y.toInt());
}
elseif(message.startsWith("sec:")){
data = message.substring(4);
data.trim();
S = data;
myRTC.setDS1302Time(S.toInt(),M.toInt(),
H.toInt(), 06, D.toInt(),
Mon.toInt(), Y.toInt());
}
else
{
Serial.println("Incorrect command");
}
message ="";
incomingChar = 0;
}
myRTC.updateTime();
lcd.setCursor(0, 0);
D = myRTC.dayofmonth;
Mon = myRTC.month;
lcd.print("date: ");
lcd.print(D.length()==1?"0"+D:D);
lcd.print("/");
lcd.print(Mon.length()==1?"0"+Mon:Mon);
```

```
lcd.print("/");
lcd.print(myRTC.year);
lcd.print(" ");
lcd.setCursor(0, 1);
H=myRTC.hours;
M=myRTC.minutes;
S=myRTC.seconds;
lcd.print("time: ");
lcd.print(H.length()==1?"0"+H:H);
lcd.print(":");
lcd.print(M.length()==1?"0"+M:M);
lcd.print(":");
lcd.print(S.length()==1?"0"+S:S);
lcd.print(" ");
}
```

Задати дату і час можна за допомогою введення відповідних команд через COM-порт:

```
year: XXXX      month: XX       day: XX

hours: XX       min: XX         sec: XX
```

де XX - число

Завдання:

1. Ввести за допомогою команд COM-порту в годинник реального часу дату і час.

2. Додати в програму код, що дозволяє задавати дату і час за допомогою команд з COM-порта:

```
date: ДД/ММ/ГГГГ

time: ЧЧ:ММ:СС
```

7.3 Контрольні запитання

1. Опишіть структуру і функції модулю годинника реального часу DS1302?

2. Схема підключення модулю годинника реального часу DS1302 до контролеру?

3. Підключення додаткових бібліотек в Arduino IDE?

4 Як організувати керування годинником реального часу через послідовний порт?

5. Який канал зв’язку використовувався при роботі з рідкокристалічним дисплеєм у данній роботі?.

Додаток А
Приклад оформлення титульної сторінки

Міністерство освіти та науки України

Запорізький національний технічний університет

Кафедра МіНЕ

ЗВІТ
З ЛАБОРАТОРНОЇ РОБОТИ №__

_____
(тема роботи)
з дисципліни „ МІКРОПРОЦЕСОРНІ ПРИСТРОЇ
КЕРУВАННЯ ТА ОБРОБКИ ІНФОРМАЦІЇ“

Виконав:
студент гр. РТ–__ _____
(Ініціали, Прізвище)

Прийняв:
(посада) _____
(Ініціали, Прізвище)

20___ р.