МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ ДЕРЖАВНИЙ ПОЛІТЕХНІЧНИЙ КОЛЕДЖ

Для спеціальностей:
123 Комп'ютерна інженерія

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт
з дисциплін: "Мікропроцесорні системи"

Харків 2019

Методичні вказівки до виконання лабораторних робіт з дисциплін: “Мікропроцесорні системи” для студентів спеціальності 123 Комп’ютерна інженерія;

Галузь знань: 12 Інформаційні технології
Спеціальність: 123 Комп’ютерна інженерія

Спеціалізація: 5.123.1 Обслуговування комп’ютерних систем і мереж

Відділення: Автоматизації, комп’ютерної інженерії та будівництва

Укладачі: В.О. Величко, А.А. Дігтяр, М.В. Величко - Харків:ХДПК, 2019, 58 с.

Розглянуто цикловою комісією інформаційних технологій
Протокол №____ від _____2019 р.
Голова циклової комісії _____М.М. Бочарніков

Схвалено методичною радою коледжу
Протокол від _____2019 р. №____
Голова методичної ради _____

Лабораторна робота №1. Ознайомлення з інтегрованою середою програмування мікроконтролерів. Дослідження роботи з портами

Мета роботи: Ознайомитися з інтегрованою середою програмування на ПК

Розробка власних програм на базі плат, сумісних з архітектурою Arduino, здійснюється в офіційному безкоштовному середовищі програмування Arduino IDE. Середовище налаштовано для написання, компіляції та завантаження власних програм в пам'ять мікроконтролера, встановленого на платі Arduino-сумісного пристрою. Основою середовища розробки є мова Processing / Wiring - це фактично звичайний C++, доповнений простими і зрозумілими функціями для керування введенням / виводом на контактах. Існують версії середовища для операційних систем Windows, Mac OS і Linux.

Останню версію середовища Arduino можна скачати зі сторінки завантаження офіційного сайту www.arduino.cc.

ПІДКЛЮЧЕННЯ Ардуіно (ARDUINO)

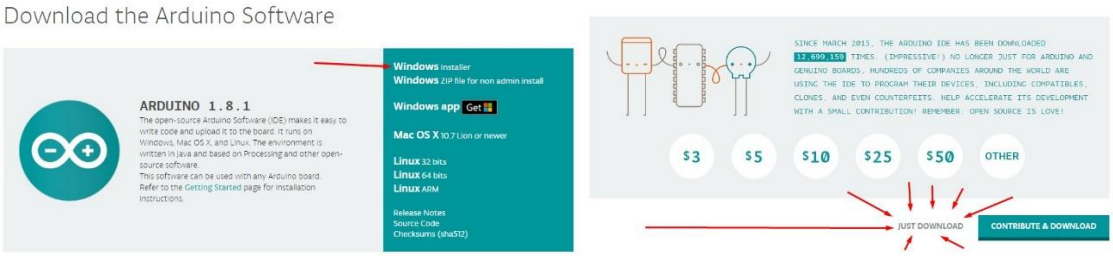
1. Качаємо і встановлюємо Arduino IDE

Качаємо і встановлюємо Arduino IDE * з сайту розробника. При скачуванні можна відмовитися від пожертвування, натиснувши JUST DOWNLOAD (тільки завантажити). При установці Arduio IDE повинні автоматично встановитися драйвера, тобто при появі віконця «погодитися чи на установку драйверів» натиснути так.

* Це програма для написання скетчів і прошивки Arduino

Качаємо і встановлюємо JRE (Java Runtime Environment) * з сайту розробника.

* Arduino IDE працює на Java, тобто потрібно завантажити і встановити безкоштовний пакет, без якого неможлива робота ніяких програм, написаних на Java.



2. Встановлюємо драйвера

Для китайської Arduino NANO завантажити і встановити драйвер CH341 *, посилання нижче, див. Перший скріншот.

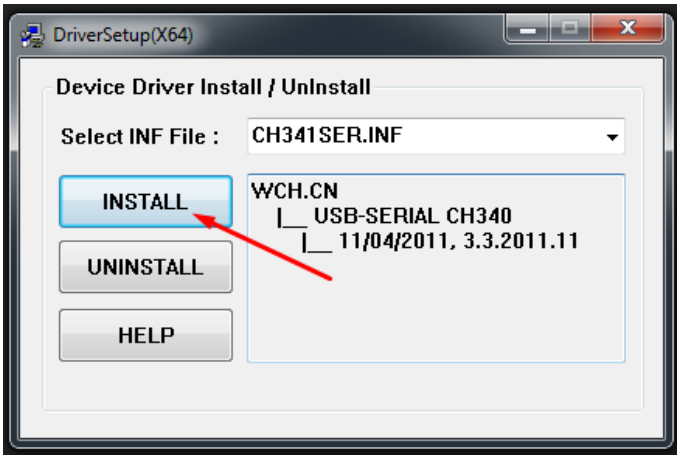
* На китайських НАНАХ стоять USB контролери CH340 / CH341, для правильної роботи потрібен спеціальний драйвер. Це єдина відмінність китайських Ардуіно від оригінальних.

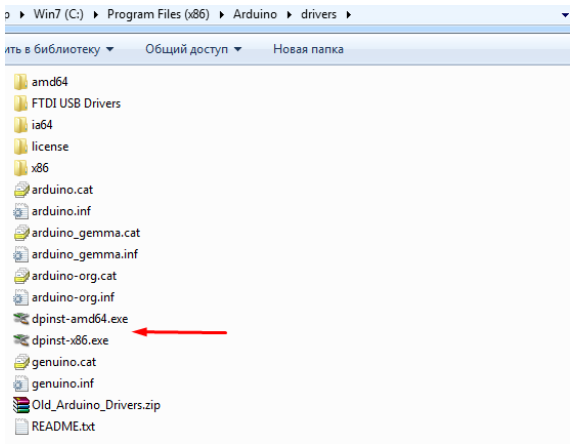
При установці Arduio IDE повинні автоматично поставить драйвера.

Якщо цього не відбулося, встановити драйвера Arduino з папки з Arduino IDE (C: \\ Program files ...), см. Другий скріншот.

Підключити Arduino до комп'ютера, почекати, поки Windows її розпізнає і запам'ятає (перше підключення).

P.S. Вилізе віконце, яке повідомить, що пристрій упізнано і підключено до COM порту з певним номером (2, 3, 6, 9 ...)





3. Налаштовуємо Arduino IDE

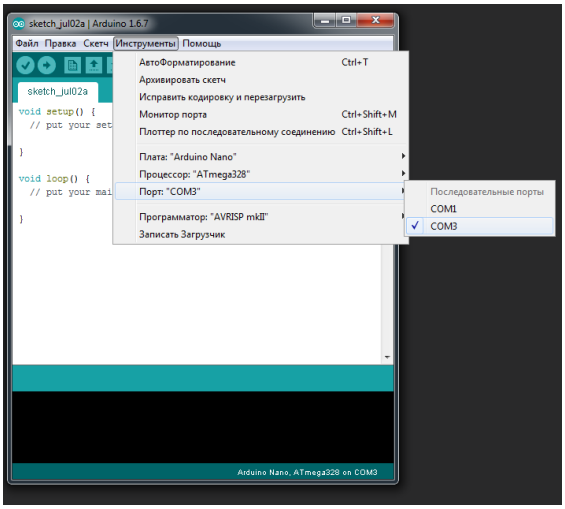
Запустити Arduino IDE, вибрати плату (Інструменти \ плата \ "ваша плата»). Див. Перший скріншот.

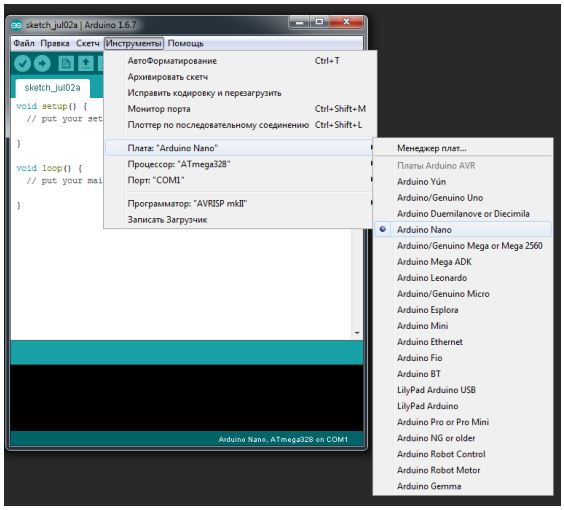
Вибрати порт: інструменти \ порт \ "COM відмінний від COM1, наприклад COM3, COM5 ...» Див. Другий скріншот. Який саме порт ви могли бачити при першому підключенні Ардуіно до комп'ютера.

Примітка: якщо у вас тільки COM1 - значить або не встали драйвера, або здохла плата.

Готові прошивки просто відкриваються подвійним кліком. Щоб завантажити прошивку, тисніть кнопку ЗАВАНТАЖИТИ на верхній панелі інструментів, вона у вигляді стрілочки.

УВАГА, РАДА! У ШЛЯХУ ДО папку зі скачати скетч НЕ ПОВИННО БУТИ КИРИЛИЦІ! СТВОРІТЬ НА ДИСКУ ПАПКУ ARDUINO, І ПРАЦЮЙТЕ В НІЙ!





4. Установка бібліотек Arduino

Припустимо, завантажили бібліотеку. Її потрібно розпакувати і покласти в папку:

C: \ Program Files (x86) \ Arduino \ libraries \ (Windows x64)

C: \ Program Files \ Arduino \ libraries \ (Windows x86)

Як приклад - бібліотека для дисплея на чіпі TM1637, дивіться скріншот

В папці libraries повинна з'явитися папка TM1637, в якій є папка examples, і два файли з розширеннями .h і .cpp. Ці два файли повинні бути в кожній бібліотеці.

УВАГА, ЧАСТА ПОМИЛКА ПРИ ВСТАНОВЛЕННЯ БІБЛІОТЕК! Подивіться уважно на скріншот зверху: бібліотека завжди містить файли з розширеннями .h і .cpp (або вони знаходяться в папці src), а також папку прикладів examples і іноді файл keywords. Так ось! Ці файли бібліотеки повинні знаходитися в папці, яка знаходиться в папці libraries, а не в папці!

Простий приклад:

C:\Program Files (x86)\Arduino\libraries\GyverButton\(файлы библиотеки) — ПРАВИЛЬНО

C:\Program Files (x86)\Arduino\libraries\GyverButton\GyverButton-master\(файлы библиотеки) — НЕПРАВИЛЬНО

5. Основні помилки при прошивці Arduino (FAQ)

5.1 Помилка компіляції

Виникає на етапі складання і компіляції прошивки. Помилки компіляції викликані проблемами в коді прошивки, тобто проблема суто софтварная. Зліва від кнопки «завантажити» є кнопка з галочкою - перевірка. Під час перевірки проводиться компіляція прошивки і виявляються помилки, якщо такі є. Ардуіно в цьому випадку може бути взагалі не підключено до комп'ютера.

У деяких випадках помилка виникає при наявності кирилиці (російських букв) в шляху до папки зі скетчем. Рішення: завести для скетчів окрему папочку в корені диска з англійською назвою.

У чорному віконці в самому низу Arduino IDE можна прочитати повний текст помилки.

У викачаних з інтернету готових скетчах часто виникає помилка з описом <назва файлу> no such file or directory. Це означає, що в скетчі використовується бібліотека <назва файлу>, і потрібно покласти її в Program Files / Arduino / libraries. Також бібліотеки завжди можна пошукати в гуглі по <назва файлу>.

При використанні якихось особливих методів і функцій помилкою може стати неправильно обрана плата в «Інструменти / плата».

Якщо прошивку пишете ви, то будь-які синтаксичні помилки в коді будуть підсвічені, а знизу в чорному віконці можна прочитати більш детальний опис, в чому власне помилка. Зазвичай вказується рядок, в якій зроблена помилка, також цей рядок підсвічується червоним.

5.2 Помилка завантаження

Виникає на етапі, коли прошивка зібрана, скомпільована, в ній немає критичних помилок, і проводиться завантаження в плату по кабелю. Помилка може виникати як унаслідок несправностей заліза, так і з-за налаштувань програми та драйверів.

USB кабель, яким підключається Arduino, повинен бути Data-кабелем. Існують кабелі, призначені тільки для зарядки, у них всередині 2 дроти. Data кабель має 4 дроти, два з яких потрібні для передачі даних. Таким кабелем підключаються до комп'ютера плеєри та смартфони.

Причиною помилки завантаження є невстановлені драйвера CH340, якщо у вас китайська NANO.

Також буде помилка, якщо Ви не оберете COM порт, до якого підключена Arduino. Якщо крім COM1 інших портів немає - читай два пункти над цим, або спробуй інший USB порт, або взагалі інший комп'ютер.

Більшість проблем при завантаженні, викликаних «зависанням» Ардуіно або завантажувача, лікуються повним відключенням Ардуіно від харчування. Потім вставляємо USB і заново прошиває.

Якщо в описі помилки зустрічається слово averdude або bootloader is not responding - з імовірністю 95% здох завантажувач, наприклад при випадковому короткому замиканні проводи на плату. Решта 5% - завантажувач «злетів», і його можна прошити заново програматором або інший Ардуіно. Детальніше про це можна почитати в гуглі по «як перепрошити завантажувач на Ардуіно».

Розглянемо установку Arduino IDE на комп'ютері з операційною системою Windows. Вирушаємо на сторінку www.arduino.cc, вибираємо версію для операційної системи Windows і викачуємо архівний файл. Він містить все необхідне, в тому числі і драйвери. Після закінчення завантаження розпаковуємо скачаний файл в зручне для себе місце.

Тепер необхідно встановити драйвери. Підключаємо Arduino до комп'ютера. На контролері повинен засвітитися живлення - зелений світлодіод. Windows починає спробу установки драйвера, яка закінчується повідомленням «Можливо, драйвер не було встановлено». Відкриваємо Диспетчер пристроїв. У складі пристроїв знаходимо значок Arduino Uno - пристрій відзначено знаком оклику.

Клацаємо правою кнопкою миші на значку Arduino Uno і у вікні, вибираємо пункт - Оновити драйвери і далі пункт Виконати пошук драйверів на цьому комп'ютері. Вказуємо шлях до драйвера - ту папку на комп'ютері, куди вони витягли скачаний архів. Нехай це буде папка drivers каталогу установки Arduino -

наприклад, C:\arduino-1.0\drivers. Ігноруємо всі попередження Windows і отримуємо в результаті повідомлення - Оновлення програмного забезпечення для даного пристрою завершено успішно. У заголовку вікна буде вказано і COM-порт, на який встановлено пристрій.

Тепер можна запускати Arduino IDE.

Середовище розробки Arduino (див. рис. 1) складається з:

• редактора програмного коду;

• області повідомлень;

• вікна виведення тексту;

• панелі інструментів з кнопками часто використовуваних команд;

• декількох меню.

Скетч пишеться в текстовому редакторі, який має колірне підсвічування створюваного програмного коду. Під час збереження і експорту проекту в області повідомлень з'являються пояснення і інформація про помилки. Вікно виведення тексту показує повідомлення Arduino, що включають повні звіти про помилки та іншу інформацію.

Кнопки панелі інструментів дозволяють перевірити і записати програму, створити, відкрити і зберегти скетч, відкрити моніторинг послідовної шини.

Скетчам, що розробляються, може бути додана додаткова функціональність за допомогою бібліотек, що представляють собою спеціальним чином оформлений програмний код, що реалізовує деякий функціонал, який можна підключити до створюваного проекту. Спеціалізованих бібліотек існує безліч. Зазвичай бібліотеки пишуться так, щоб спростити вирішення тієї чи іншої задачі і приховати від розробника деталі програмно-апаратної реалізації.

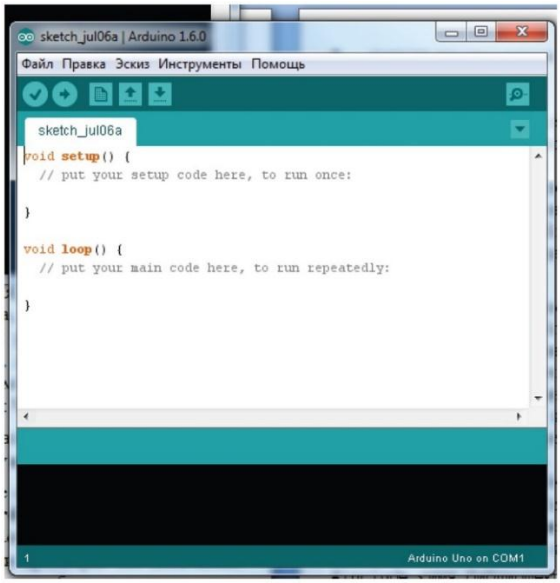


Рис. 1 - Средовище Arduino IDE.

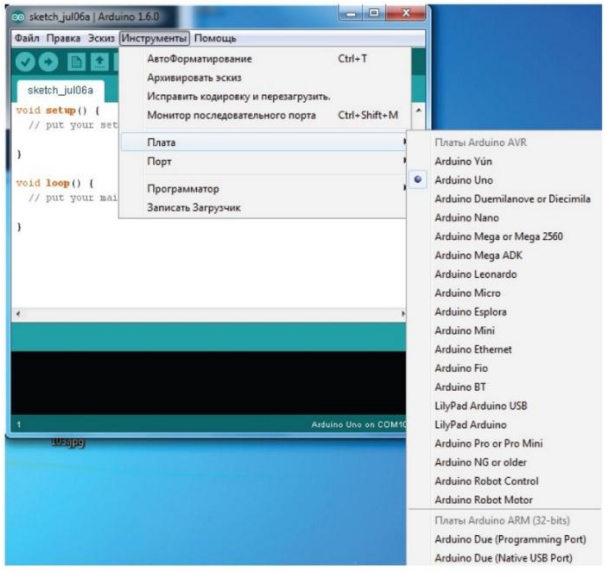


Рис. 2 - Вибір Arduino плати.

Програма, написана в середовищі Arduino, носить назву скетч.

Середа Arduino IDE поставляється з набором стандартних бібліотек. Вони знаходяться в підкаталозі libraries каталогу установки Arduino. Необхідні бібліотеки можуть бути також завантажені з різних ресурсів. Якщо бібліотека встановлена правильно, то вона з'являється в меню Ескіз | Імпорт бібліотек. Вибір бібліотеки в меню призведе до додавання до початкового коду рядка #include <ім'я бібліотекі.h> Ця директива підключає заголовки з описом об'єктів, функцій і констант бібліотеки, які тепер можуть бути використані в проекті. Середа Arduino буде компілювати створюваний проект разом із зазначеною бібліотекою.

Перед завантаженням скетчу потрібно задати необхідні параметри в меню Інструменти | Плата (Tools | Board) (рис. 3) і порт | Послідовний порт.

Сучасні платформи Arduino перезавантажуються автоматично перед завантаженням. На старих платформах необхідно натиснути кнопку перезавантаження. На більшості плат під час процесу завантаження будуть мигати світлодіоди RX і TX. При завантаженні скетчу використовується завантажувач (bootloader) Arduino - невелика програма, що завантажується в мікроконтролер на платі. Вона дозволяє завантажувати програмний код без використання додаткових апаратних засобів. Робота завантажувача розпізнається по миганню світлодіода на цифровому виводі D13.

Монітор послідовного порту (Serial Monitor) відображає дані, що їх посилають в платформу Arduino (плату з інтерфейсом USB або послідовної шини). Тепер, коли ми трохи дізналися про Arduino і середовище програмування Arduino IDE, перейдемо до лабораторних занять – створення проектів.

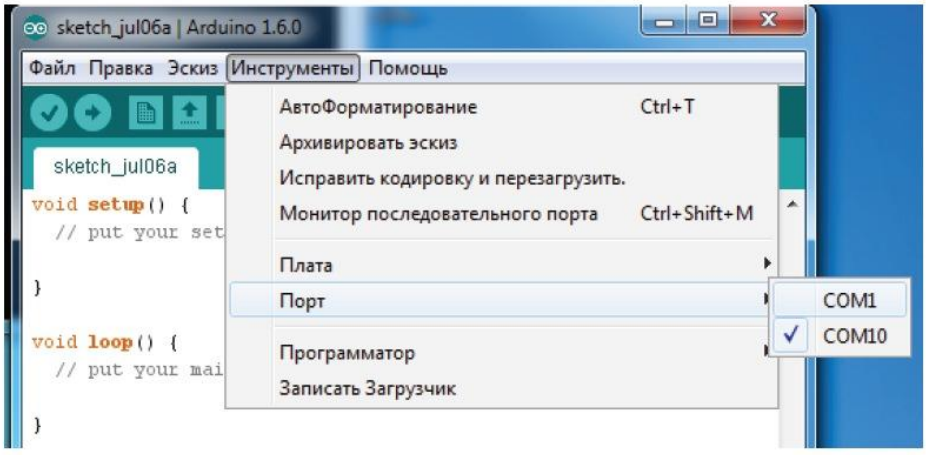


Рис. 3 - Вибір порту підключення плати Arduino

Останнім часом з'явилося багато моделей контролерів Arduino, в яких в якості USB-чіпсета використовуються мікросхеми серії CH340.

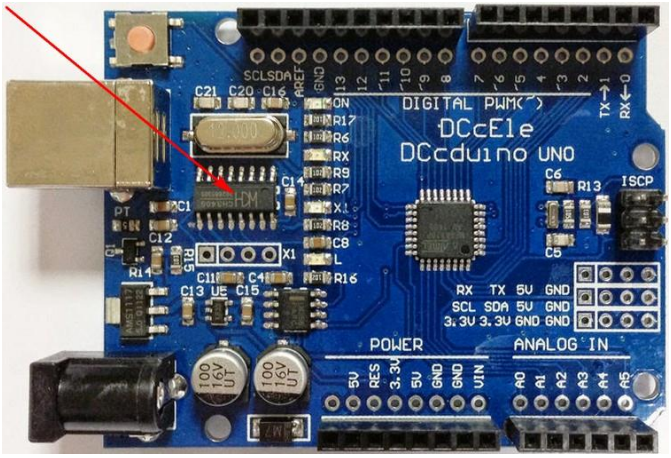


Рис. 4 - Плата Arduino UNO.

Якщо при першому підключенні Arduino, комп'ютер не зміг визначити новий пристрій Вам буде достатньо завантажити і встановити останню версію драйвера USB-SERIAL CH340.

Контрольні запитання:

1) Що таке Arduino?

2) Що таке Arduino IDE?

3) З чого складається апаратна частина Arduino?

4) Що таке технологія VLIW?

Лабораторна робота№2. Розробка схеми електричної принципової мікропроцесорної системи.

Мета роботи: ознайомитись з редактором схем САПР Fritzing, налаштуванням його опцій, навчитись підключати бібліотеки компонентів, створювати електричні принципові схеми.

Теоретична частина

Початковим етапом розробки довільного радіоелектронного пристрою є опис його роботи на деякому рівні абстракції, в якості якої можуть виступати структурна, функціональна, або електрична принципова схеми. При реалізації проектів друкованих плат робота починається з формування ідеї розробника у вигляді електричної принципової схеми. Редактори схем практично усіх програм даного типу подібні між собою, однак в Fritzing є досить велика кількість опцій. Основною особливістю Fritzing є проектна структура розробки, а також незвична розробникам схем і плат процедура компіляції схеми і проекту.

Fritzing - це програма з відкритим кодом, розроблена для того, щоб полегшити процес прототипирования проектів на базі популярних платформ: Arduino, Raspberry Pi і багатьох інших.

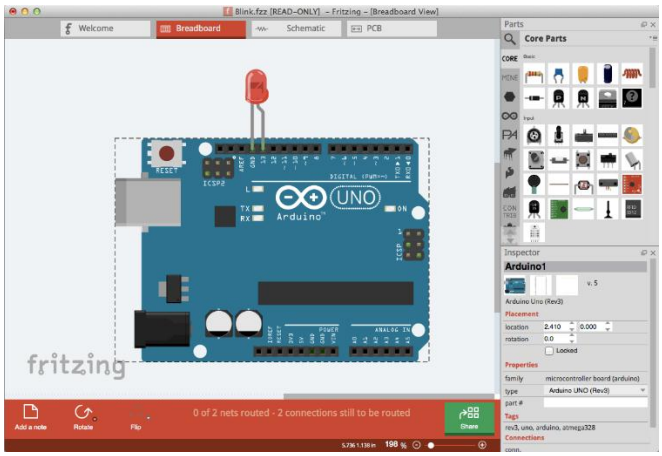


Рисунок - САПР моделювання електронних схем Fritzing

У ній зберігається величезна кількість віртуальних моделей самих різних платформ, компонентів і модулів, які ви можете розставляти на робочому полі і підключати до макетної платі, створюючи таким чином принципову схему вашого майбутнього устрою.

Більш того у Fritzing можна створити навіть макет друкованої плати, щоб в майбутньому її виготовити.

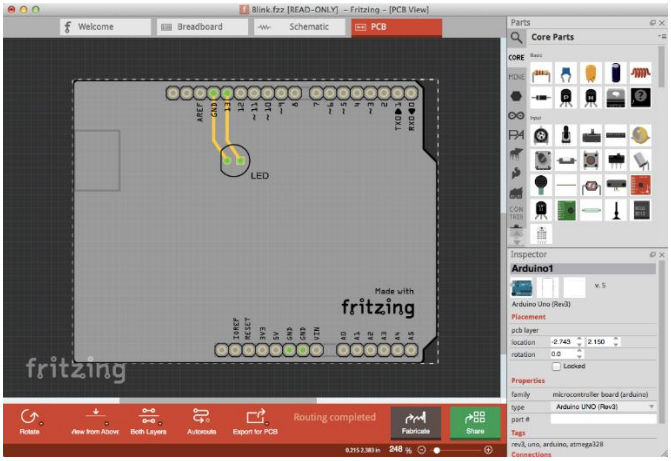


Рисунок - Макет плати

Інструмент безумовно корисний, а головне простий у використанні. Вам необхідно просто підключити всі необхідні елементи, і навіть якщо допустите помилку жоден з віртуальних компонентів не згорить і не прийде в непридатність.

Робота у Fritzing. Коли перший раз відкриваєте проект під Fritzing, перед вами з'явиться таке віконце

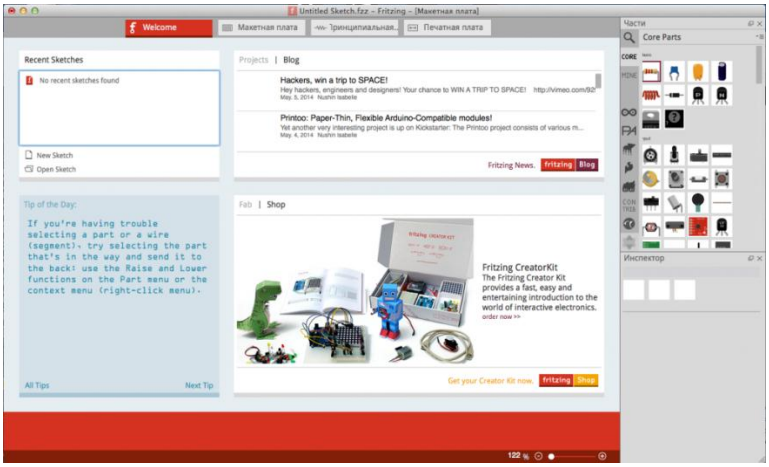


Рисунок - Запуск САПР моделювання електронних схем Fritzing

Переключившись на вкладку Макетна плата то побачимо наступний екран.

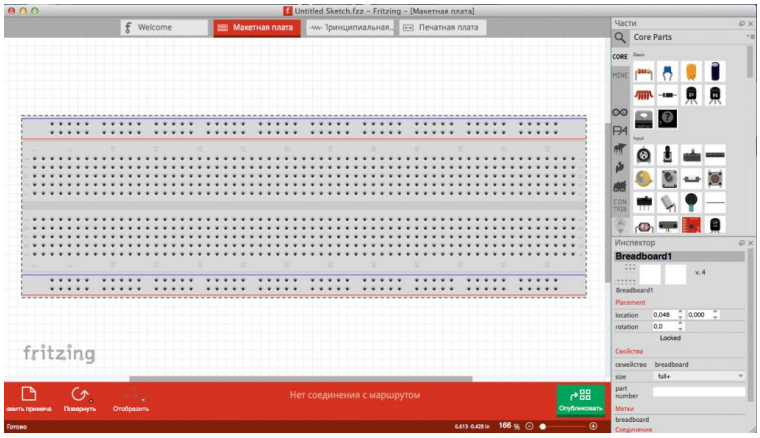


Рисунок - Вибір електронних компонентів

У правій частині екрана знаходиться панель інструментів з усіма елементами і опціями. Якщо компонент налаштовується, то в нижній частині панелі інструментів відображаються настроюються параметри для цього компонента.

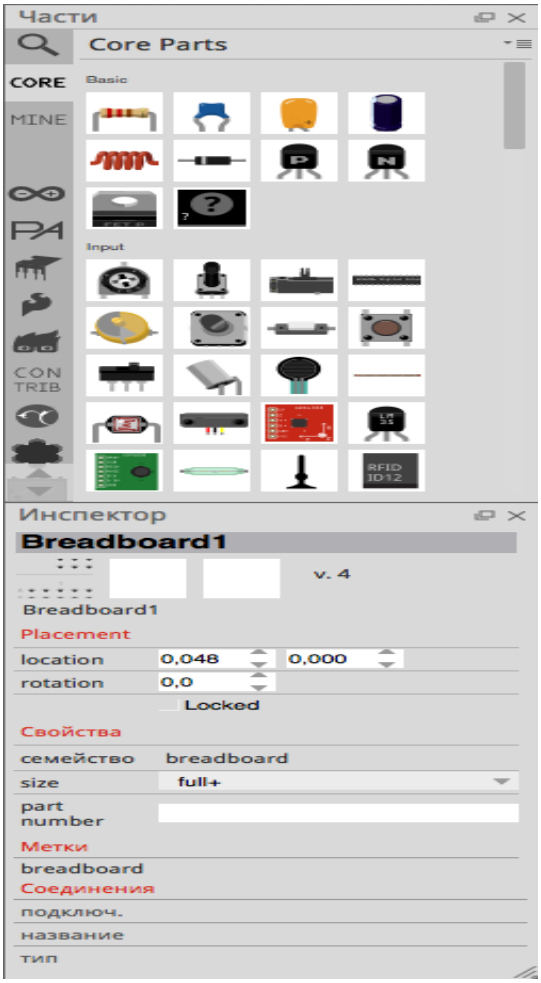


Рисунок - Бібліотека електронних компонентів

Для виділеного елемента ми можемо налаштувати його параметри в нижній частині панелі інструментів для зміни значення його опору, допуску (tolerance) і відстань між висновками.

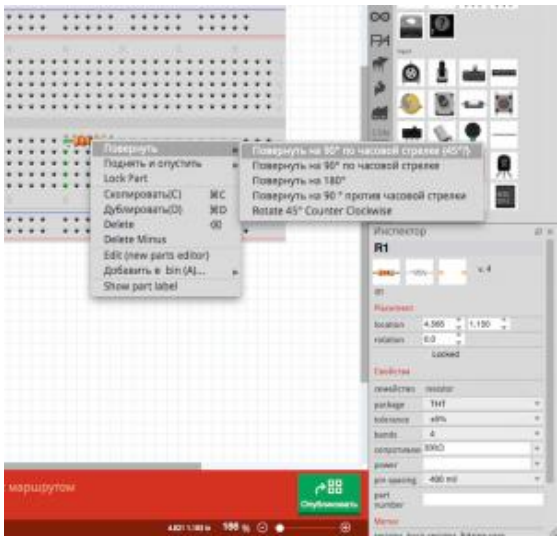


Рисунок - Налаштування електронних компонентів

Так само, як на реальній макетної платі, можемо додати дроти, для підключення необхідних нам елементів. Наведіть курсор миші на отвір на макетної платі і зверніть увагу, що воно стає синім. Це означає, що можна починати вести провід. Клацніть отвір на макетної платі і, не відпускаючи лівої кнопки миші, перетягніть другий кінець дроту в потрібну точку. Підключений позитивний вивід світло діода до верхнього ряду контактів на макетної платі і з'єднаний другий вивід світло діода з резистором.

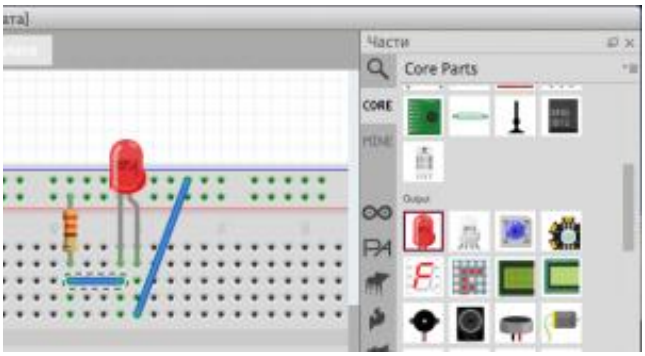


Рисунок - З’єднання електронних компонентів

Завдання:

Створити схему електричну принципову мікропроцесорної системи в САПР Fritzing згідно із вашим завданням

Лабораторна робота №3. Мови програмування. Бітові операції

Мета роботи: Навчитися працювати із бітовими операціями на Arduino

ХІД РОБОТИ

В цьому проекті ми навчимося керувати світлодіодом. Змусимо його блимати.

Для реалізації проекту нам потрібні наступні компоненти:

• контролер Arduino UNO R3;
• плата для прототипування;
• світлодіод;
• резистор 220 Ом;
• провід тато-тато.

Світлодіод - це напівпровідниковий прилад, що перетворює електричний струм безпосередньо в світлове випромінювання.

По-англійськи світлодіод називається light emitting diode, або LED. Кольорові характеристики світлодіодів залежать від хімічного складу використаного в ньому напівпровідника. Світлодіод випромінює у вузькій частині спектру, його колір чистий, що особливо цінують дизайнери. Світлодіод механічно міцний і винятково надійний, його термін служби може досягати 100 тисяч годин, що майже в 100 разів більше, ніж у лампочки розжарювання, і в 5-10 разів більше, ніж у люмінесцентної лампи. Нарешті, світлодіод - низьковольтний електроприлад, а отже, безпечний. Світлодіоди поляризовані, має значення, в якому напрямку підключати їх.

Позитивний висновок світлодіода (довший) називається анодом, негативний - катодом. Як і всі діоди, світлодіоди дозволяють току текти тільки в одному напрямку - від анода до катода. Оскільки струм протікає від позитивного до негативного, анод світлодіода повинен бути підключений до цифрового сигналу 5 В, а катод повинен бути підключений до землі. Ми будемо підключати світлодіод до цифрового контакту D10 Arduino послідовно з резистором. Світлодіоди повинні бути завжди з'єднані послідовно з резистором, який виступає в якості обмежувача струму. Чим більше значення резистора, тим більше він обмежує струм. В цьому експерименті ми використовуємо резистор номіналом 220 Ом. Схема підключення приведена на рис. 1.1. Як підібрати обмежувальний резистор і як впливатиме номінал резистора на яскравість світлодіода, ми розглянемо в експерименті 3.

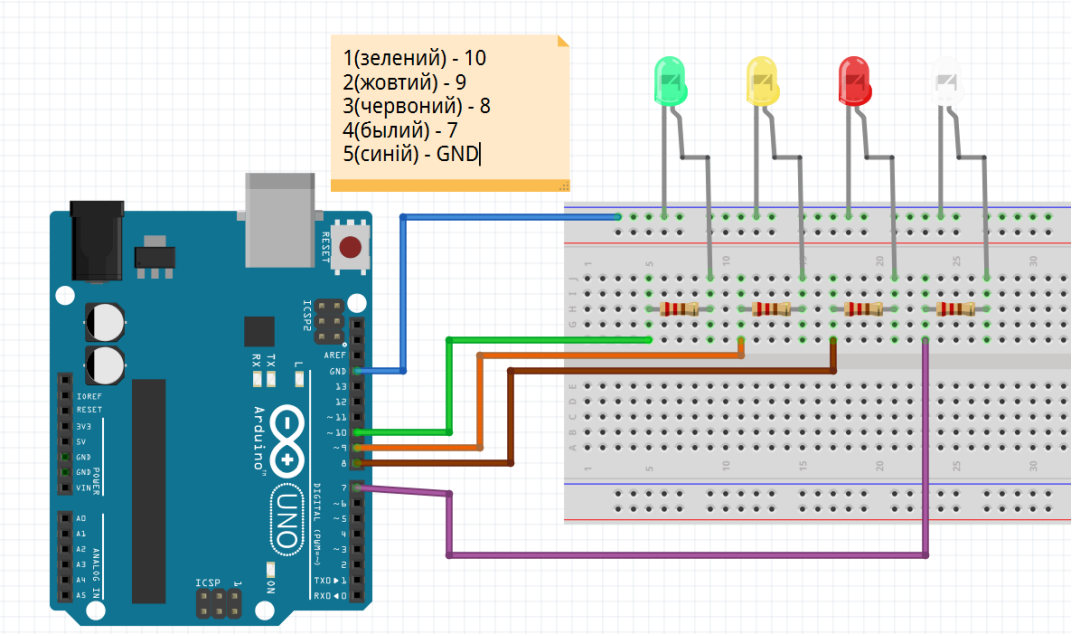


Рис.26 – Структурна схема лабораторної моделі.

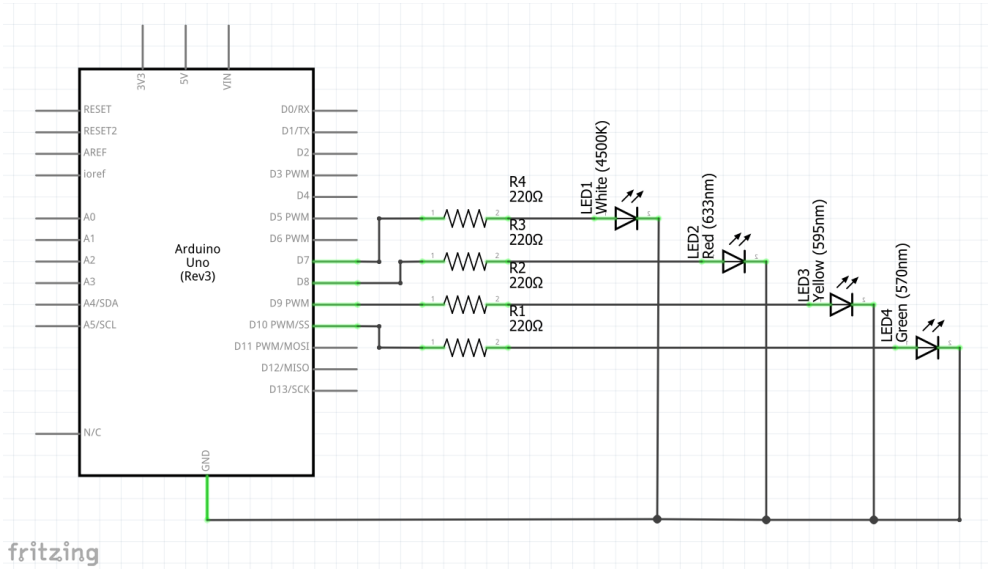


Рис. – Принципова електрична схема лабораторної моделі.

Світлодіод послідовно з резистором підключаємо до цифрового висновку Arduino D10. За замовчуванням всі висновки Arduino налаштовані як входи. Ми збираємося використовувати висновок Arduino як вихід, тому необхідно його переконфигурировать, видавши контролера відповідну команду.

```
pinMode (10, OUTPUT);
```

Для миготіння світлодіода необхідно поперемінно с певним інтервалом подавати на висновок Arduino сигнали HIGH (високий рівень або 1) і LOW (низький рівень або 0). Інтервал зміни сигналу на виході D10 Arduino будемо встановлювати за допомогою команди delay (), що затримує виконання скетчу на заданий час в мілісекундах (мс).

Скетч експерименту наведено в лістингу 1.1.

СКЕТЧ 2

```
const int LED1=10;              // вывод для подключения светодиода 10 (D10)
const int LED2=9;               // вывод для подключения светодиода 9 (D9)
const int LED3=8;               // вывод для подключения светодиода 8 (D8)
const int LED4=7;              // вывод для подключения светодиода 7 (D7)
void setup()
{
                                 // Конфигурируем вывод подключения светодиода
как выход (OUTPUT)
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
```

```
pinMode(LED4, OUTPUT);
}
void loop()
{
                                    // включаем светодиод, подавая на вывод 1 (HIGH)
digitalWrite(LED1,HIGH);
delay(1000);                        // пауза 1 сек (1000 мс)
digitalWrite(LED1,LOW);     // выключаем светодиод, подавая на вывод 0 (LOW)
delay(1000);                       // пауза 1 сек (1000 мс)
digitalWrite(LED2,HIGH);    // включаем светодиод, подавая на вывод 1 (HIGH)
delay(1000);                       // пауза 1 сек (1000 мс)
digitalWrite(LED2,LOW);    // выключаем светодиод, подавая на вывод 0 (LOW)
delay(1000);                      // пауза 1 сек (1000 мс)
digitalWrite(LED3,HIGH);   // включаем светодиод, подавая на вывод 1 (HIGH)
delay(1000);                      // пауза 1 сек (1000 мс)
digitalWrite(LED3,LOW);   // выключаем светодиод, подавая на вывод 0 (LOW)
delay(1000);                     // пауза 1 сек (1000 мс)
digitalWrite(LED4,HIGH);  // включаем светодиод, подавая на вывод 1 (HIGH)
delay(1000);                     // пауза 1 сек (1000 мс)
digitalWrite(LED4,LOW);  // выключаем светодиод, подавая на вывод 0 (LOW)
delay(1000);                    // пауза 1 сек (1000 мс)
}
```

Завдання:

1) Створити світлофор для машин так пішоходів, який працює в автоматичному режимі.
2) Створити світлофор для машин так пішоходів, який переключає режими при натисканні кнопки.

Контрольні запитання:

1) Накресліть алгоритм роботи лістінгу.
2) Розєми Arduino, та їх призначення?
3) Оператор else…if
4) Навіщо потрібен or

Лабораторна робота №4. Вивчення засобів відображення інформації.

Мета роботи: Навчитися складати схеми підключення індикатора LCD 1602 до ARDUINO та складати програму управління цими індикаторами.

ХІД РОБОТИ

Для реалізації проекту потрібні наступні компоненти:

- Плата Arduino Uno;
- USB-кабель;
- LCD монітор 1602;
- дроти.

Розглянемо на цьому занятті, як підключити LCD дисплей до Ардуіно по I2C. Розповімо, як правильно підключити LCD монітор до Arduino і розглянемо основні команди ініціалізації і управління LCD 1602. Також розглянемо різні функції в мові програмування C++, для виведення текстової інформації на дисплеї, який часто потрібно використовувати в проектах на Ардуіно.

LCD 1602 I2C підключення до Arduino

I2C - послідовна двохпровідна шина для зв'язку інтегральних схем всередині електронних приладів, відома, як I²C або IIC (англ. Inter-Integrated Circuit). I²C була розроблена фірмою Philips на початку 1980-х років, як проста 8-бітна шина для внутрішнього зв'язку між схемами в керуючої електроніці (наприклад, в комп'ютерах на материнських платах, в мобільних телефонах і т.д.).

У простій системі I²C може бути кілька ведених пристроїв і однопровідний пристрій, який ініціює передачу даних і синхронізує сигнал. До ліній SDA (лінія даних) і SCL (лінія синхронізації) можна підключити декілька ведених пристроїв. Часто провідним пристроєм є контролер Ардуіно, а відомими пристроями: годинник реального часу або LCD Display.

Як підключити LCD 1602 до Ардуіно по I2C

Рідкокристалічний дисплей 1602 з I2C модулем підключається до плати Ардуіно всього 4 проводами - 2 дроти даних і 2 дроти живлення. Підключення дисплея 1602 проводиться стандартно для шини I2C: вивід SDA підключається до порту A4, вивід SCL - до порту A5. Живлення LCD дисплея здійснюється від порту +5V на Arduino. Дивіться докладніше схему підключення ЖК монітора 1602 на рис.19.

Після підключення LCD монітора до Ардуіно через I2C вам буде потрібно встановити бібліотеку LiquidCrystal_I2C.h для роботи з LCD дисплеєм по інтерфейсу I2C і бібліотека Wire.h (мається на стандартній програмі Arduino IDE).

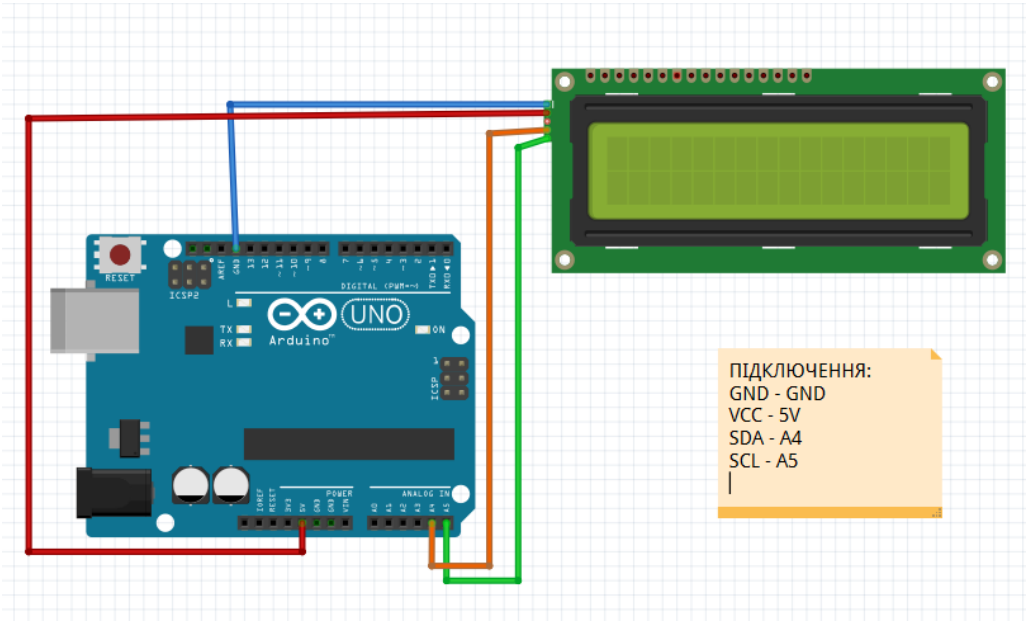


Рис. 19 - Підключення LCD 1602 к Arduino UNO через I2C.

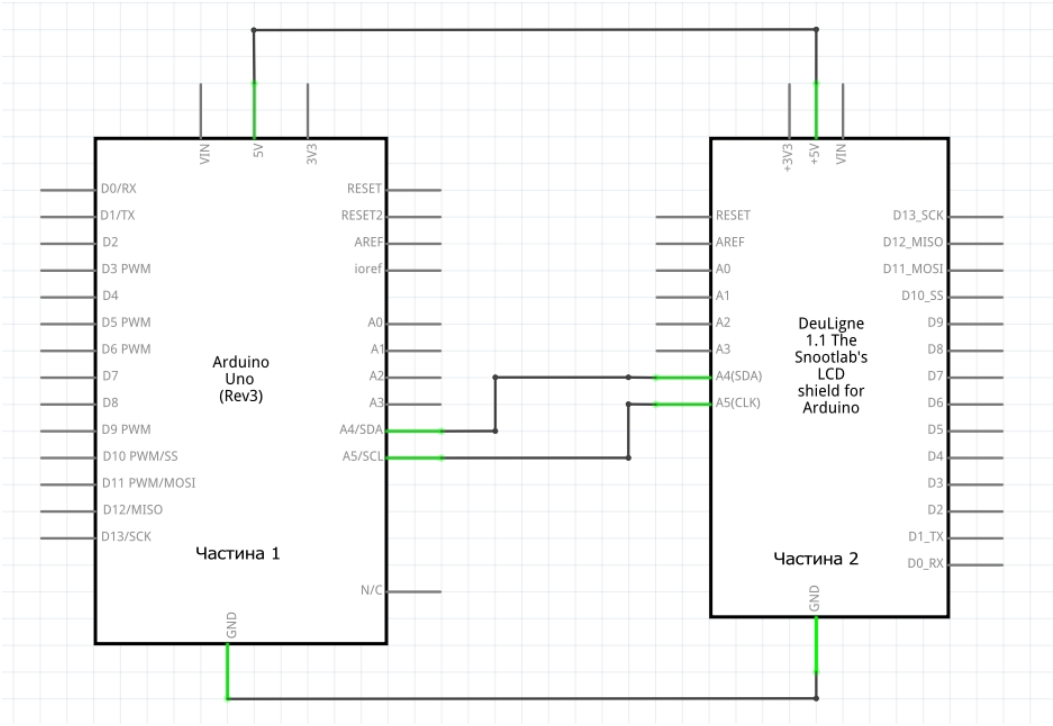


Рис. 20 – Схема електрична принципова підключення LCD 1602 к Arduino UNO через I2C.

Після установки бібліотеки напишіть наступний скетч.

ЛІСТІНГ 1

```
#include <Wire.h>                    // бібліотека для управління пристроями по I2C
#include <LiquidCrystal_I2C.h>       // підключаємо бібліотеку для LCD тисячі шістсот дві
LiquidCrystal_I2C lcd (0x27,16,2);   // присвоюємо ім'я lcd для дисплея 16x2
```

```
void setup()                    // процедура setup
{
  lcd.init();                   // ініціалізація LCD дисплея
  lcd.backlight ();             // включення підсвічування дисплея

  lcd.setCursor (0,0);          // ставимо курсор на 1 символ першого рядка
  lcd.print ( "I LOVE");        // друкуємо повідомлення на першому
рядку
  lcd.setCursor (0,1);          // ставимо курсор на 1 символ другого рядка
  lcd.print ( "ARDUINO");       // друкуємо повідомлення на другому рядку}
void loop()                     // процедура loop
{
/* це багаторядковий коментар
// спочатку процедура void loop () в скетчі не
використовується
lcd.noDisplay ();               // вимикаємо підсвічування LCD дисплея
delay (500);                    // ставимо паузу
lcd.display ();                 // включаємо підсвічування LCD дисплея
delay (500);                    // ставимо паузу*/
}
```

Пояснення до коду:

1. Бібліотека LiquidCrystal_I2C.h містить безліч команд для управління LCD дисплея по шині I²C і дозволяє значно спростити скетч;

2. Скетч містить багаторядковий коментар / * ... * /, який дозволяє закоментувати відразу кілька рядків у програмі.

На що звернути увагу:

Перед виведенням інформації на дисплей, необхідно задати положення курсора командою setCursor (0,1), де 0 - номер символу в рядку, 1 - номер рядка.

ЛІСТІНГ 2

```
#include <Wire.h>                 // бібліотека для управління пристроями по I2C
#include <LiquidCrystal_I2C.h>  // підключаємо бібліотеку для LCD тисячі
шістсот дві
LiquidCrystal_I2C lcd (0x27,20,2);      // присвоюємо ім'я lcd для дисплея 20x2
                                        // створюємо символ серця і трьох букв
на кирилиці
byte heart[8] = { 0b00000, 0b01010, 0b11111, 0b11111, 0b11111, 0b01110, 0b00100,
0b00000 };
```

```
  byte I[8] = { 0b01111, 0b10001, 0b10001, 0b01111, 0b00101, 0b01001, 0b10001,
0b00000 };
  byte D[8] = { 0b01111, 0b01001, 0b01001, 0b01001, 0b01001, 0b11111, 0b10001,
0b00000 };
  byte P[8] = { 0b11111, 0b10001, 0b10001, 0b10001, 0b10001, 0b10001, 0b10001,
0b00000 };

  void setup()                          // процедура setup
  {
  lcd.init();                           // ініціалізація LCD дисплея
  lcd.backlight ();                     // включення підсвічування дисплея
                                        // присвоюємо символам порядковий
номер  lcd.createChar(1, heart);
    lcd.createChar(2, I);
    lcd.createChar(3, D);
    lcd.createChar(4, P);
    lcd.setCursor(6,0);                 // встановлюємо курсор на 6 символ
першого рядка  lcd.print(char(2));
    lcd.print(" ");
    lcd.print(char(1));
    lcd.setCursor(6,1);                 // встановлюємо курсор на початок другого
рядка
    lcd.print("X");
    lcd.print(char(3));
    lcd.print(char(4));
    lcd.print("K");
  }
  void loop()                           // процедура loop
  {
  }
```

ЛІСТІНГ 3

```
  #include <Wire.h>                     // Підключаємо бібліотеку для роботи з
шиною I2C
  #include <LiquidCrystal_I2C.h>  // Підключаємо бібліотеку для роботи з LCD
дисплеєм
  по шині I2C
  LiquidCrystal_I2C lcd (0x27,16,2);        // Оголошуємо об'єкт бібліотеки, вказуючи
параметри
  дисплея (адреса I2C = 0x27, кількість стовпців = 16, кількість рядків = 2)
```

```
                                        // Якщо напис не з'явилася, замініть адресу
0x27 на
   0x3Fvoid setup(){
      lcd.init();                       // Почнемо роботу з LCD дисплеєм
      lcd.backlight ();                 // Включаємо підсвічування LCD дисплея
      lcd.setCursor (0, 0);             // Встановлюємо курсор в позицію (0
стовпець, 0
   рядок)
      lcd.print ( "HDPK");              // Виводимо текст "HDPK", починаючи з
встановленої
   позиції курсора
      lcd.setCursor (0, 1);             // Встановлюємо курсор в позицію (0
стовпець, 1
   рядок)
      lcd.print ( "www.xemttc.at.ua");  // Виводимо текст " www.xemttc.at.ua ",
починаючи з
   встановленої позиції курсора}
   void loop(){}                        // Код всередині функції loop виконується
постійно.
```

Але так як ми виводимо статичний текст, нам достатньо його вивести 1 раз при старті, без використання коду loop

Завдання для самостійного виконання:

1. Змініть скетч так, щоб написи I LOVE і ARDUINO виводилися по центру LCD дисплея і завантажте змінений скетч в мікроконтролер;
2. Зніміть багатостроковий коментар / * ... * / і завантажте скетч;
3. Додайте в скетчі функцію управління Ардуіно з комп'ютера - вивід на дисплей тексту, в залежності від команди в Serial monitor.

Контрольні запитання:

1) Накресліть алгоритм роботи 2 лістінгу.
2) Дайте визначення «Бібліотека»
3) Назвіть основні особливості використання масивів у мові програмування Arduino
4) Використання бібліотеки LiquidCrystal для роботи з LCD.

Лабораторна робота №5. Вивчення принципів роботи з таймерами/лічильниками.

Мета роботи: Вивчити принципи роботи з таймерами/лічильниками.

Навчитися складати схеми підключення семисегментних індикаторів до ARDUINO та складати програму управління цими індикаторами.

ХІД РОБОТИ

В цьому експерименті ми розглянемо роботу з семісегментним світлодіодним індикатором, яка дозволяє Arduino візуалізувати цифри.

Необхідні компоненти:

• контролер Arduino UNO R3;
• плата для прототипування;
• однорозрядний семисегментний індикатор;
• резистор 510 Ом - 7 од.;
• дроти.

Світлодіодний семисегментний індикатор являє собою групу світлодіодів, розташованих в певному порядку і об'єднаних конструктивно. Світлодіодні контакти промарковані мітками від a до g (і додатково dp - для відображення десяткового дробу), і один загальний вивід, який визначає тип підключення індикатора (схема із загальним анодом ОА, або загальним катодом ОК). Запалюючи одночасно кілька світлодіодів, можна формувати на індикаторі символи цифр. Схема однорозрядного семисегментного індикатора показана на рис. 5.

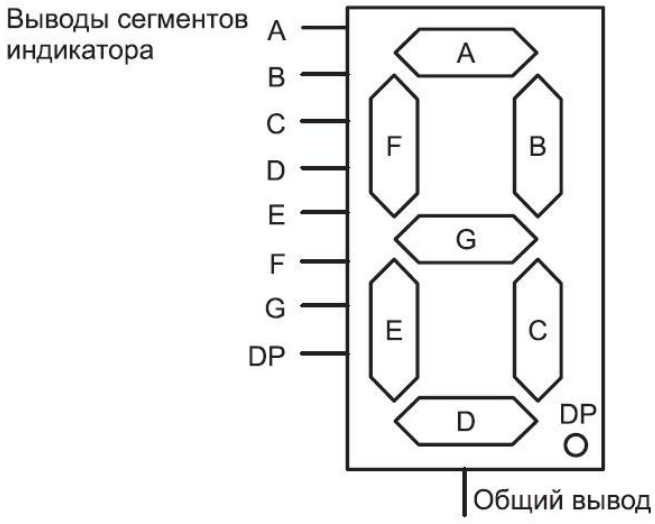


Рис. 5 - Схема однорозрядного семисегментного індикатора.

Для підключення однорозрядного світлодіодного індикатора до Arduino будемо задіювати 7 цифрових виводів, до кожного контакту a-g індикатора підключається

до виводу Arduino через обмежувальний резистор 470 Ом. У нашому експерименті ми використовуємо семисегментний індикатор з загальним катодом ОК, загальний провід приєднуємо до землі. На рис. 5 показана схема підключення однорозрядного семисегментного індикатора до плати Arduino.

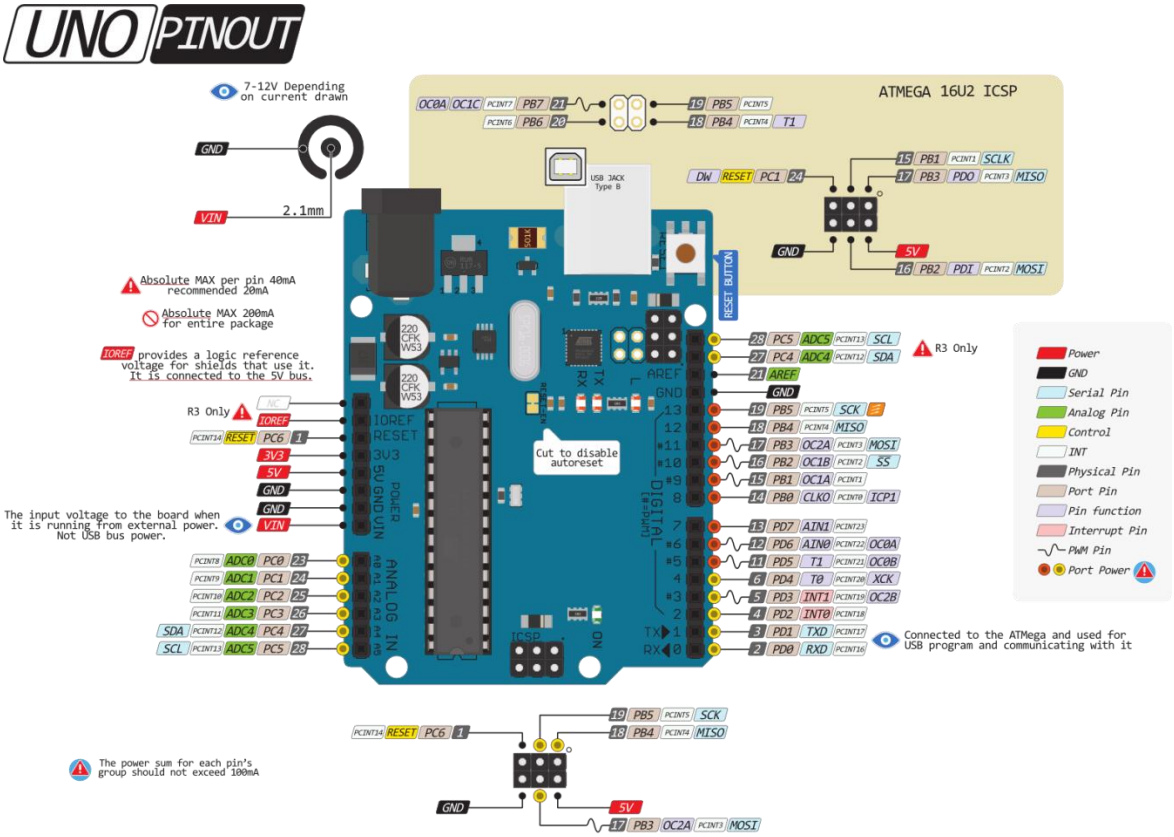


Рис.6 – Призначення контактів ARDUINO UNO.

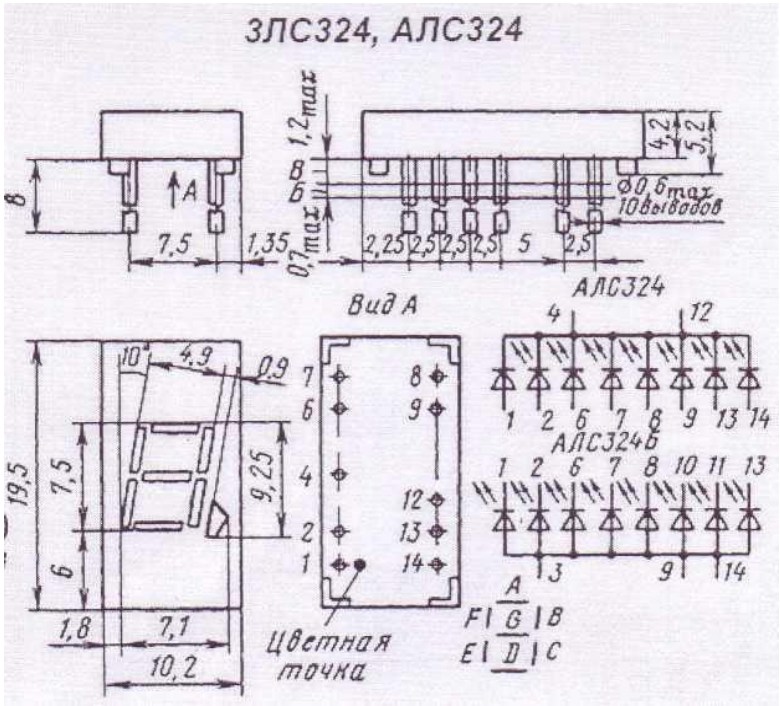


Рис.7 – Призначення контактів АЛС324А1.

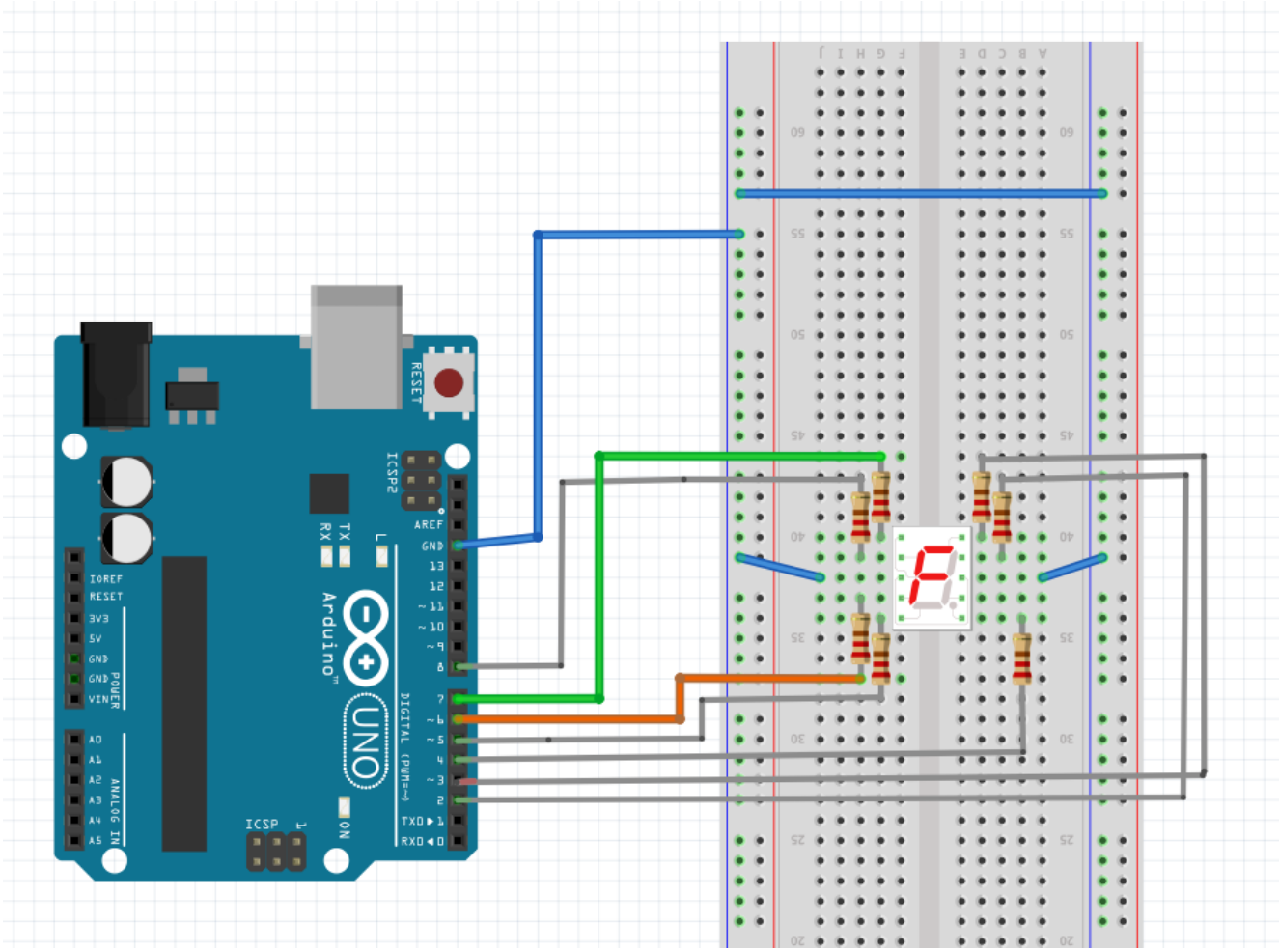


Рис.8 – Структурна схема підключення семисегментного індикатора до плати Arduino.

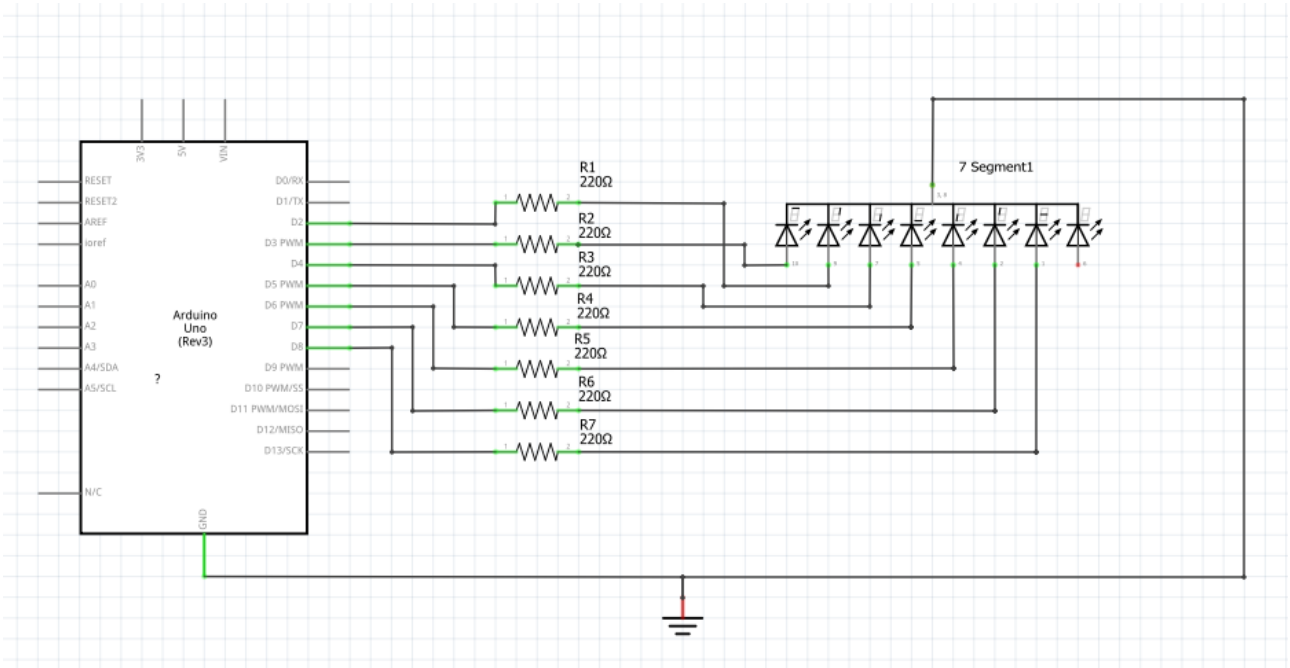


Рис.9 – Принципова електрична схема підключення семисегментного

індикатора до плати Arduino.

Приступимо до написання скетчу. Ми будемо на семисегментний індикатор в циклі виводити цифри від 0 до 9 з паузою 1 секунда. Сформуємо масив значень для цифр 0-9, де старший розряд байта відповідає мітці сегмента a індикатора, а молодший - сегменту g.

byte numbers [10] = {B11111100, B01100000, B11011010, B11110010, B01100110,
B10110110, B10111110, B11100000, B11111110, B11110110};

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

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

Скетч проекту представлений в лістингу:

ЛІСТІНГ 1

```
int pins[7]={2,3,4,5,6,7,8};    /* список виводів Arduino для підключення до розрядів
                                    a-g   семисегментного індикатора */
byte numbers[10] = {B1111110, B1010000, B1101101, B1111001,
B1010011,B0111011, B0111111, B1110000, B1111111,B1111011};
                                    // значення для виводу цифр 0-9
int number=0;                   // змінна для зберігання значення поточної цифри
void setup()
{
for(int i=0;i<7;i++)
pinMode(pins[i],OUTPUT);  // Зконфігурувати контакти як виходи
}
void loop()
{
showNumber(number);
delay(1000);
number=(number+1)%10;
}
void showNumber(int num) // функція виводу цифри на семисегментний індикатор
{
for(int i=0;i<7;i++)
{
if(bitRead(numbers[num],7-i)==HIGH) // зажечь сегмент
digitalWrite(pins[i],HIGH);
else                                // потушити сегмент
```

```
digitalWrite(pins[i],LOW);
}
}
```

При складних проектах працювати з великою кількістю станів важко. Доведеться кожного разу дивитися на схему, щоб згадати, які світлодіоди потрібно включити для відображення кожної цифри. Але можна полегшити процес теорією про одиницю інформації - байт. Байт в своєму двійковому поданні складається з 8 біт. Кожен біт може приймати значення 0 або 1. А наш світлодіодний індикатор який складається з восьми світлодіодів. Таким чином можна уявити цифру на індикаторі у вигляді набору байт, де одиниця відповідатиме за включений діод, а нуль - за вимкнений діод.

У ArduinoIDE префікс 0b індіфіцірует двійковий код.

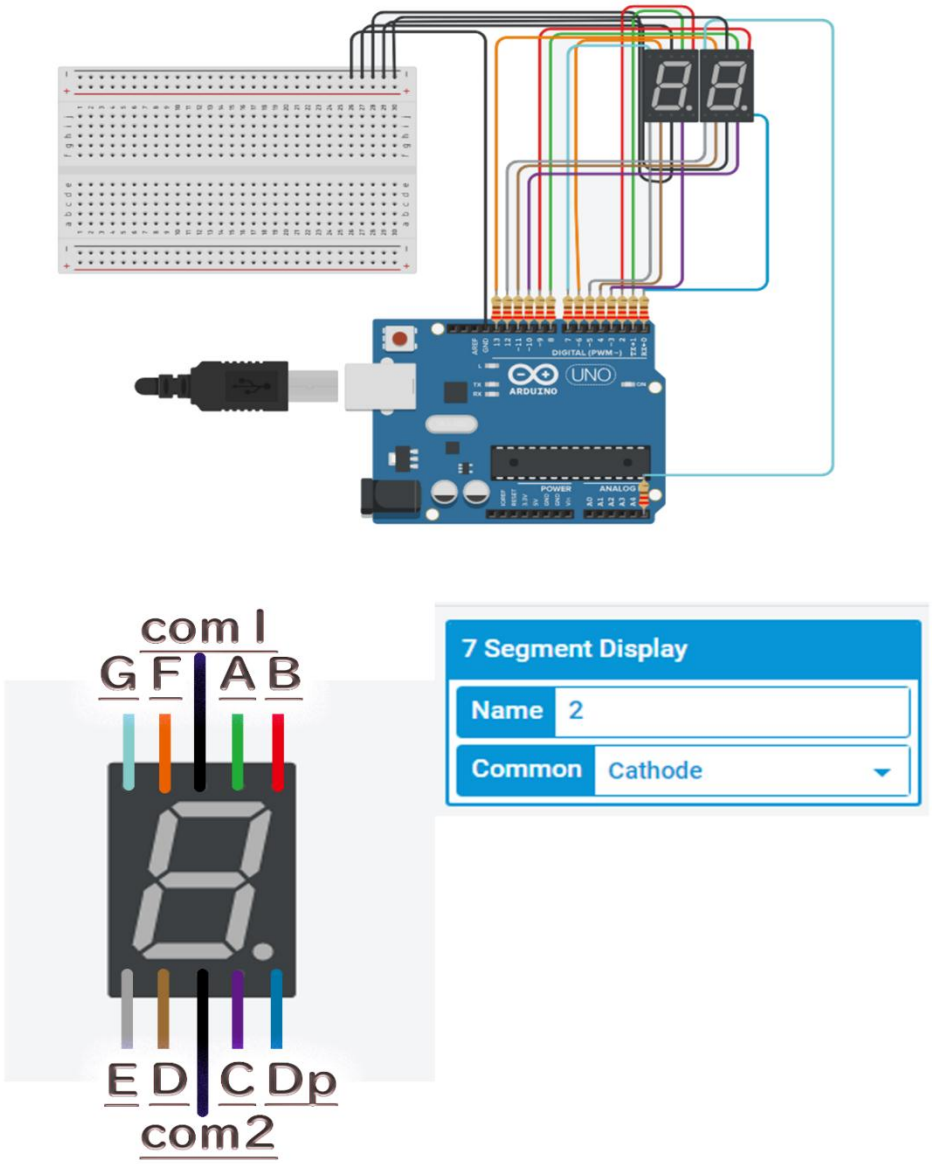


Рис.10 – Структурна схема підключення семисегментного індикатора до плати Arduino.

ЛІСТІНГ 2

```
#define led_1 1 // Определяем 1-й цифровой порт для сегментов индикаторов.
#define led_8_2 13 // Определяем последний цифровой порт сегментов.
bytenumberS[10]{ 0b0111111,0b0000110,0b1011011,0b1001111, 0b1100110,
0b1101101, 0b1111101,0b0000111,0b1111111, 0b1101111,}; // Записуем кодировку
состояний сегментов.
void setup(){
for(int i = 0; i< led_8_2; ++i){
pinMode(i + 7, OUTPUT);} // Устанавливаем режим работы цифровых портов на
выдачу сигнала.
pinMode(A5, OUTPUT); //... аналогового порта на выдачу сигнала.
}
voidloop()
{
intnumb = (millis()/10000)%10; //Вводим переменную для счёта 10-ков секунд
работы.
int numb2 = (millis()/1000)%10; //...секунд работы.
for(int i = 0; i<7;++i) // для каждого из 7 сегментов индикатора определяем:
должен ли он быть включён. Для этого считываем бит, соответствующий
текущемусегменту «i». Истина — он установлен (1), ложь — нет (0)
 {booleanenableSegment = bitRead(numberS[numb], i); //Вводим переменную для
сохранения значения елемента кодировки состояния, соответствующую номеру
кодировки. Для 10-ковсек.
boolean enableSegment2 = bitRead(numberS[numb2], i); //...длясекунд.
  if (i == 6 && enableSegment2 == 1){digitalWrite(A5, 1);}
//Инициализируемсигналнааналоговыйпорт.
else{digitalWrite(A5, 0);}
digitalWrite(i + led_1 , enableSegment);
//Передаёмсигналнасоответствующийсегментиндикатора1.

digitalWrite(i + 8, enableSegment2); //...сегментиндикатора2.
}
 }
```

Контрольні запитання:

1) Дайте визначення, що таке 7-ми сегментний індикатор, які бувають?
2) Накресліть алгоритм роботи 1 та 2 лістінгу.
3) Дайте визначення, що таке масив?
4) Пояснити поняття «адреса» і «дані».

Лабораторна робота №6. Вивчення принципів роботи із двигунами.

Мета роботи: Навчитися складати схеми підключення крокового двигуна Nema 17, та двигунів постійного струму до ARDUINO та складати програму управління цим двигуном.

ХІД РОБОТИ

Завдання 1

Для реалізації проекту потрібні наступні компоненти:

- Arduino Uno R3;
- Драйвер крокового двигуна L298;
- Кроковий двигун Nema 17;
- Макетна плата на 400 точок;
- Набір макетних проводів;

Цей двигун має 200 кроків на оборот і може працювати з частотою обертання 60 об/хв. Якщо ви використовуєте інший кроковий двигун, уточніть крок його крок і максимальну частоту обертання. Ці параметри знадобляться вам при програмуванні Arduino.

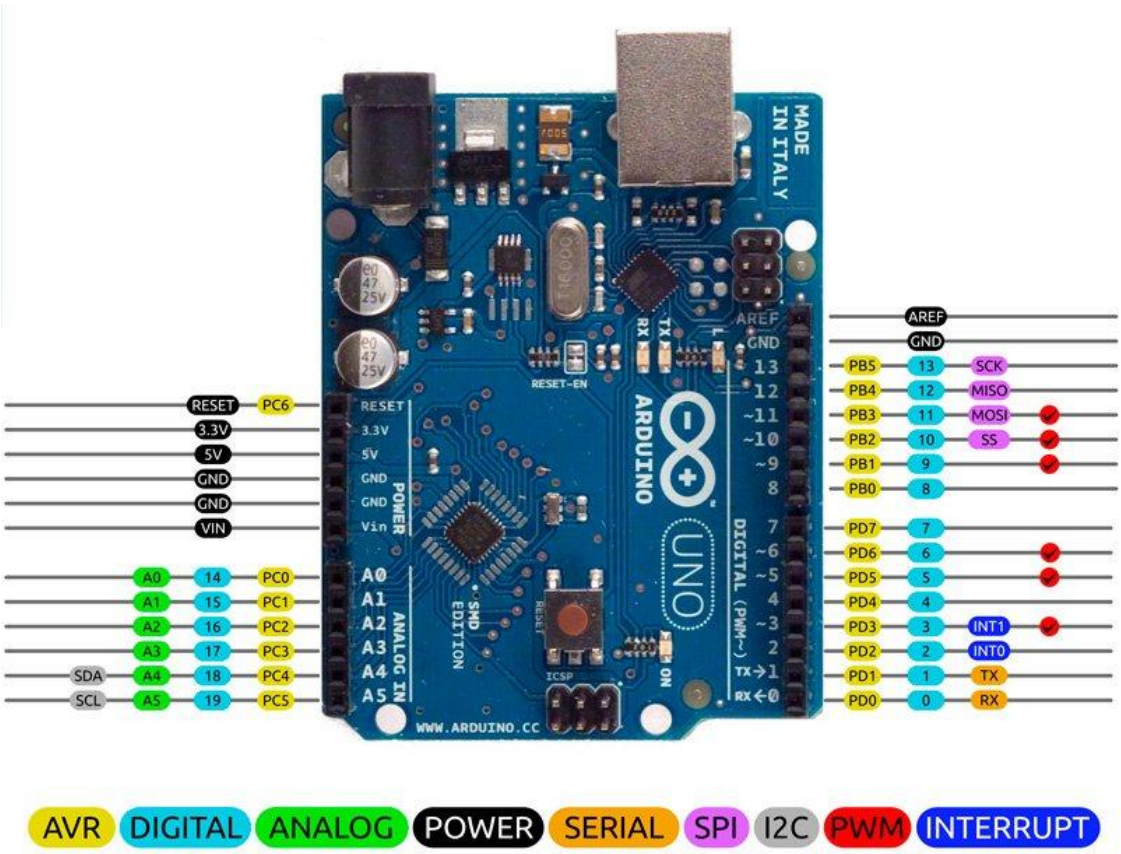


Рис.6.1 – Призначення контактів Arduino UNO.

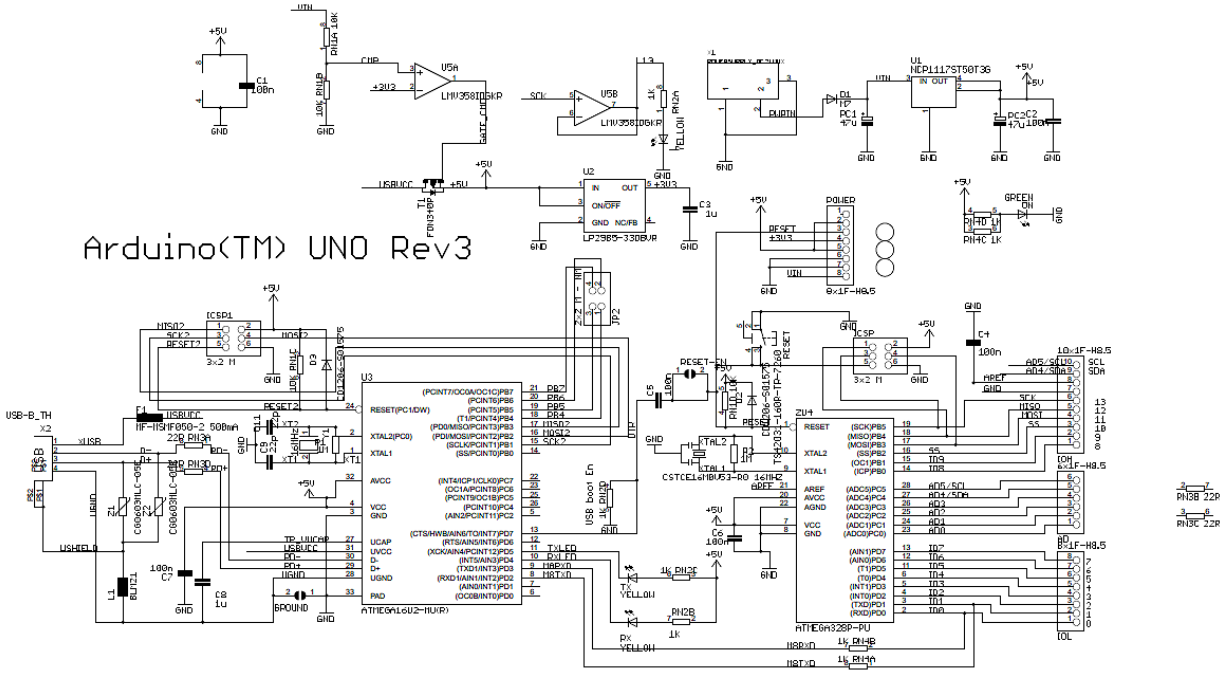


Рис.6.2 – Принципова електрична схема Arduino UNO.

Ще один важливий момент - визначити які саме кабелі відповідають A+, A-, B+ і B -. У нашому прикладі відповідні кольори кабелів: червоний, зелений, жовтий і блакитний. Переходимо до підключення.

Кабелі A +, A-, B + і B- від крокового двигуна підключаємо до пінів 1, 2, 13 і 14 відповідно. Контакти на конекторах 7 і 12 на контролері L298N залиште замкнутими. Після цього підключіть джерело живлення до піну 4 (плюс) і 5 (мінус) на контролері.

Знову таки, якщо джерело живлення менше 12 вольт, контакт, зазначений 3 на малюнку модуля, можна залишити замкнутим. Після цього, підключіть Піни модуля L298N IN1, IN2, IN3 і IN4 до відповідних цифрових пінів D8, D9, D10 і D11 на Arduino.

Тепер підключаємо GND пін з Arduino до піну 5 на контролері, а 5V до 6 піну на модулі. З управлінням крокового двигуна проблем бути не повинно завдяки вбудованій в Arduino IDE бібліотеці Stepper Library.

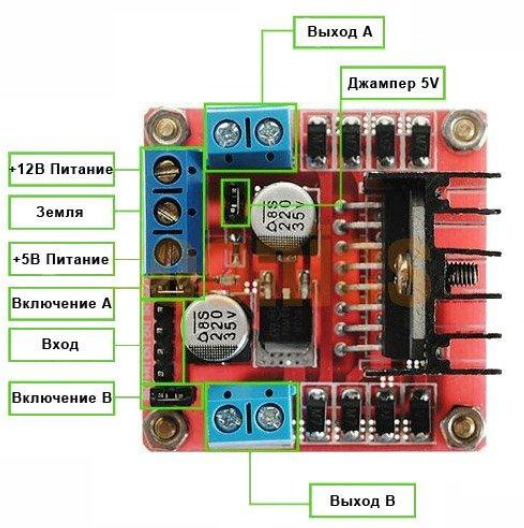


Рис.6.3 – Драйвер крокового двигуна L298.

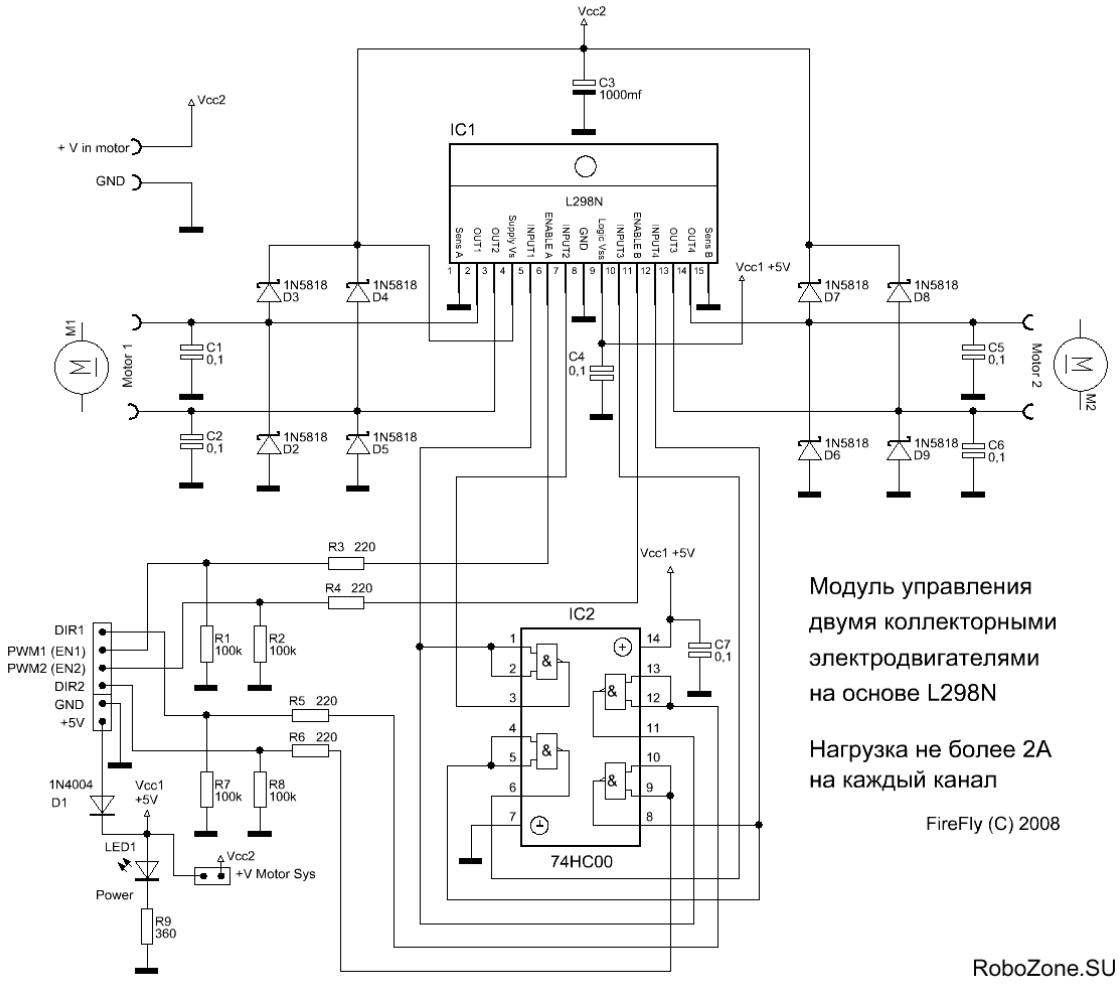


Рис.6.4 – Принципова електрична схема драйверу крокового двигуна L298.

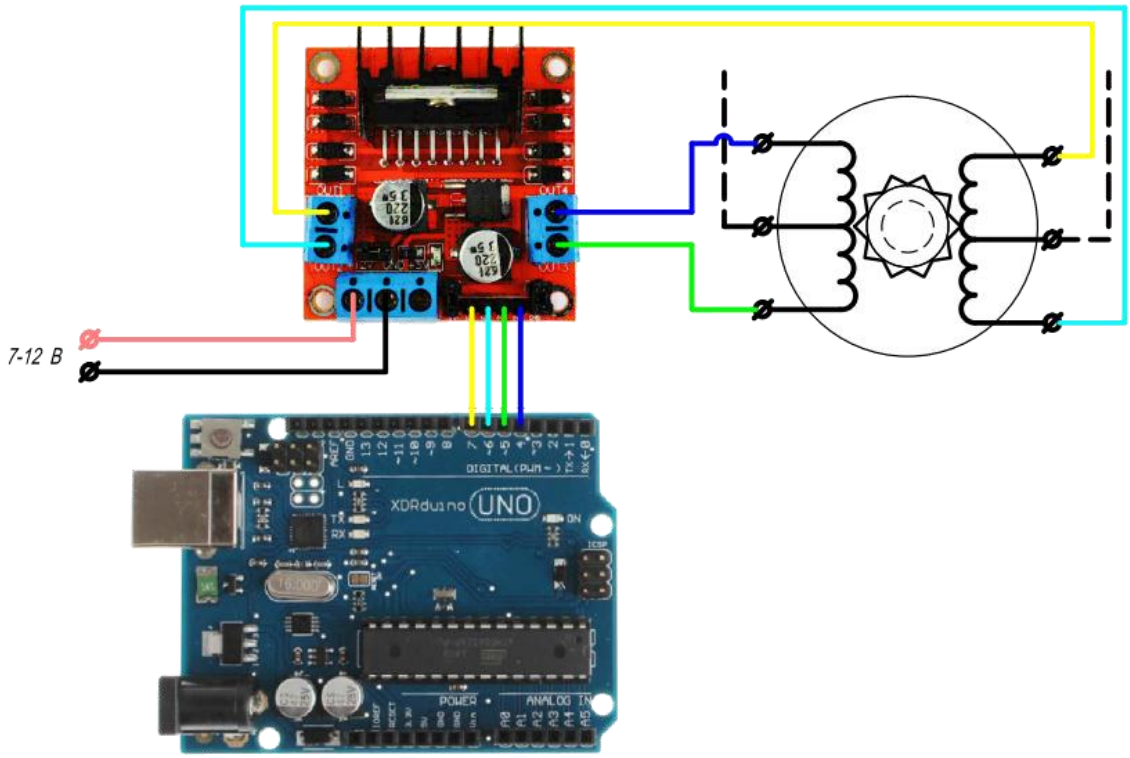


Рис.6.5 – Структурна схема підключення крокового двигуна Nema 17 до ARDUINO.

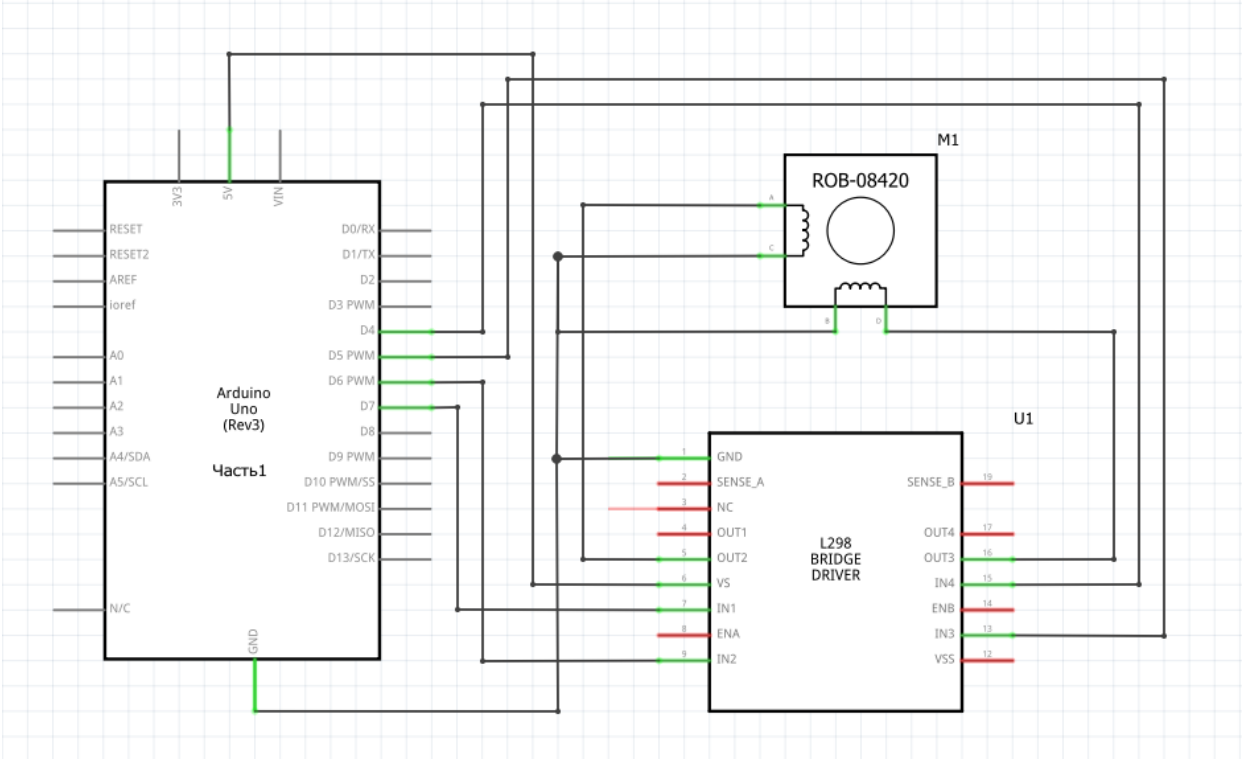


Рис.6.7 – Принципова електрична схема підключення крокового двигуна Nema 17 до ARDUINO.

Скетч проекту представлений в лістингу:

ЛІСТІНГ 1

```
#include <Stepper.h>
const int stepsPerRevolution = 200;           // кількість обертів двигуна
    // ініціалізувати шаблонну бібліотеку на виводах від 8 до 11:
Stepper myStepper(stepsPerRevolution, 4,5,6,7);
void setup() {
   // встановити швидкість 30 об/хв:
  myStepper.setSpeed(30);
   // ініціалізувати послідовний порт:
  Serial.begin(9600);
}
void loop() {
  // перший етап обертання в одну сторону:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(5000);
  // другий етап обертання в іншу сторону:
  Serial.println("counterclockwise");
  myStepper.step(-stepsPerRevolution);
  delay(5000);
}
```

ЛІСТІНГ 2

```
/* Управління кроковим двигуном, підключеного до контролера на базі
мікросхеми
 L298*/
#include <Stepper.h>     // Підключаємо бібліотеку управлення кроковим
двигуном
const int StepsForRotation=5;     // 5 шагів на оберт -  1.8 градуса на один крок
Stepper stepmotor (StepsForRotation, 4,5,6,7); /* Ініціалізуємо  кроковий двигун
200 кроків на оберт, управління обмотками через 4 та 7 цифрові виходи */
void setup()
{
  stepmotor.setSpeed(60);   /* Встанавлюємо швидкість обертання 60 обертів у
                                    мінуту (1 оборот в секунду)*/
}
void loop()
{
  stepmotor.step(100);      /* Посилаємо двигуну команду зробити 100 кроків
```

```
                                    (половину оборота)*/
  delay(1000);                  //Очикуємо одну секунду
  stepmotor.step(-100);    /* Посилаємо двигуну команду зробити 100 кроків
                                    (половину оборота) в зворотньому напрямі */
  delay(-1000);
}
```

Завдання № 2

Для реалізації проекту потрібні наступні компоненти:

- Arduino Uno R3;
- Драйвер двигуна L298;
- Двигун постійного струму 3В – 2 од;
- Набір макетних проводів.

Драйвер двигуна L298. Даний модуль дає можливість управляти одним або двома двигунами постійного струму. Для початку, підключіть двигуни до пінів A і B на контролері L298N.

Якщо ви використовуєте в проекті кілька двигунів, переконайтеся, що у них витримана однакова полярність при підключенні. Інакше, при завданні руху, наприклад, за годинниковою стрілкою, один з них буде обертатися в протилежному напрямку. Перевірте, з точки зору програмування Arduino це незручно.

Після цього підключіть джерело живлення. Плюс - до четвертого піну на L298N, мінус (GND) - до 5 піну. Якщо ваше джерело живлення до 12 вольт, конектор, зазначений 3 на рис. 17, можна залишити. При цьому буде можливість використовувати 5 вольт з піну 6 з модуля.

Даний пін можна використовувати для живлення Arduino. При цьому не забудьте підключити пін GND з мікроконтролера до 5 піну на L298N для замикання ланцюга. Тепер вам знадобиться 6 цифрових пінив на Arduino. Причому деякі піни повинні підтримувати ШІМ-модуляцію.

ШІМ-піни позначені знаком "~" поруч з порядковим номером. На рис.16 наведені ШІМ-піни на платі Arduino Uno.

Рис.6.8 – ШІМ-піни на платі Arduino Uno.

Тепер підключіть цифрові піни Arduino до драйверу. У нашому прикладі два двигуна постійного струму, так що цифрові піни D9, D8, D7 і D6 будуть підключені до пінів IN1, IN2, IN3 і IN4 відповідно. Після цього підключіть пін D10 до піну 7 на L298N (попередньо забравши коннектор) і D5 до піну 12 (знову таки, прибравши коннектор).

Напрямок обертання ротора двигуна управляється сигналами HIGH або LOW на кожен привід (або канал). Наприклад, для першого мотора, HIGH на IN1 і LOW на IN2 забезпечить обертання в одному напрямку, а LOW і HIGH змусить обертатися в протилежну сторону. При цьому двигуни не будуть обертатися, поки не буде сигналу HIGH на піні 7 для першого двигуна або на 12 піні для другого. Зупинити їх обертання можна подачею сигналу LOW на ті ж зазначені вище піни. Для керування швидкістю обертання використовується ШІМ-сигнал.

Скетч наведений нижче, спрацьовує у відповідності зі схемою підключення, яку ми розглядали вище. Двигуни постійного струму і Arduino живляться від зовнішнього джерела живлення.

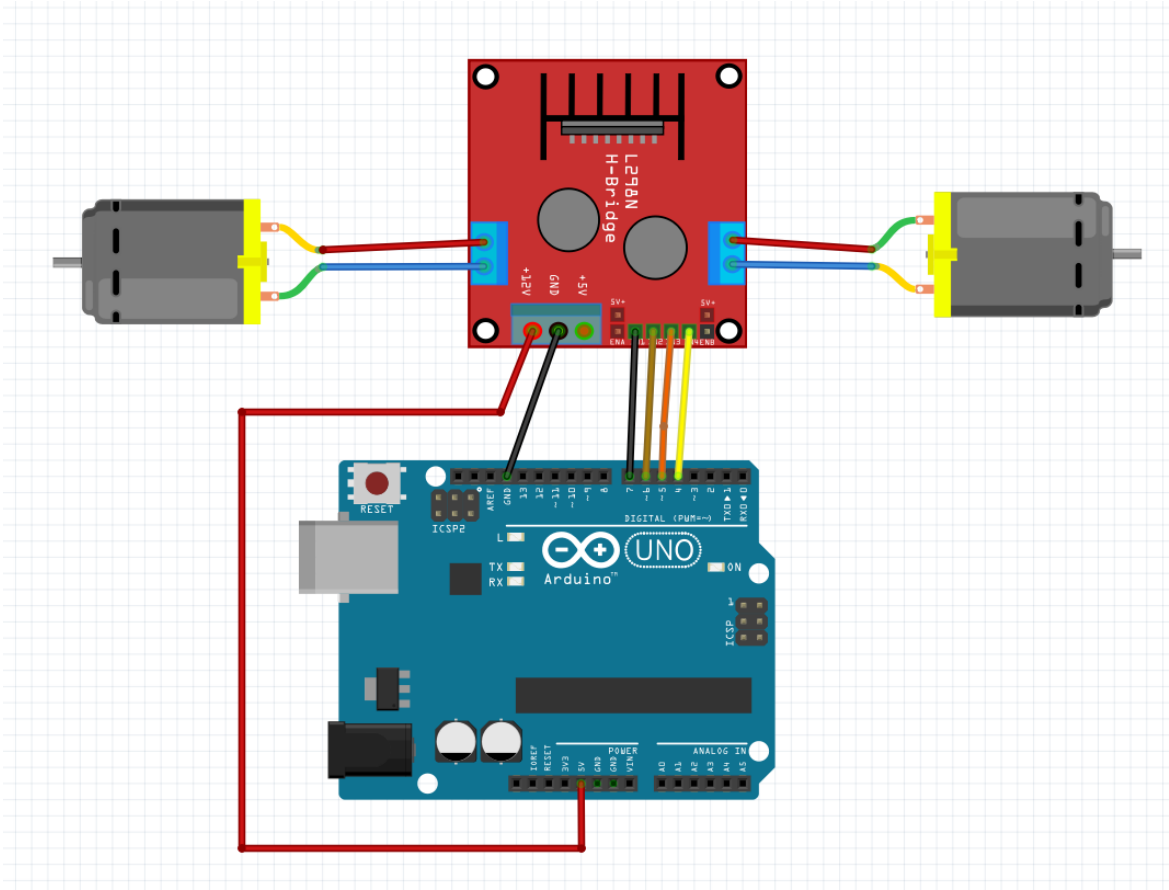


Рис.6.9 – Структурна схема підключення ДПС.

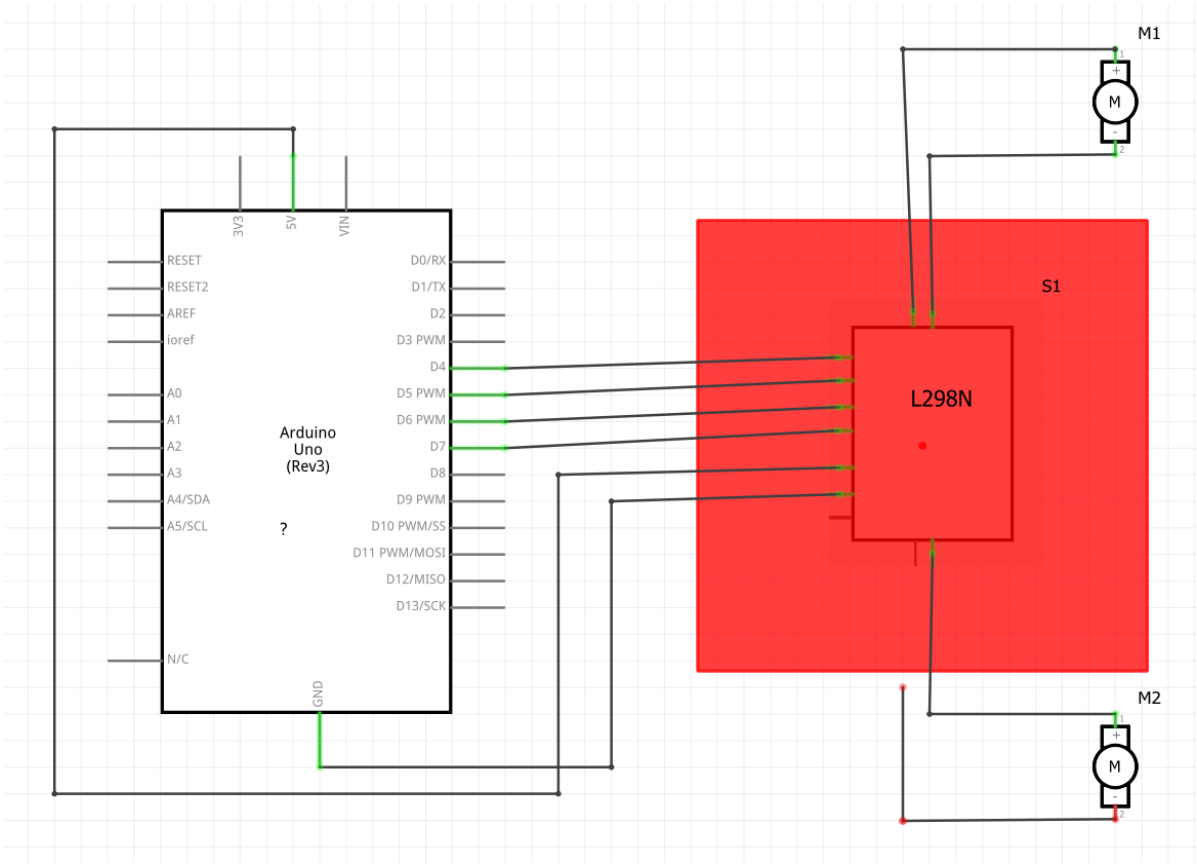


Рис.6.10 – Принципова електрична схема підключення ДПС до ARDUINO.

Пояснення до скетчу для управління двигунами постійного струму

У тілі функції demoOne () ми включаємо двигуни і починаємо з ними працювати при ШІМ- значенні 200.

Через деякий час двигуни починають обертатися в протилежному напрямку (завдяки зміні HIGH і LOW в тілі функції digitalWrite ()). Для демонстрації можливостей зміни швидкості обертання, використовуємо доступний ШІМ-діапазон в тілі функції demoTwo (). Сигнал на піні змінюється від нуля до 255 і знову до нуля.

ЛІСТІНГ 3

```
                                    // підключить піни контролера до цифрових
пінів Arduino
                                    // перший двигун
  int enA = 10;
  int in1 = 9;
  int in2 = 8;
                                    // другий двигун
  int enB = 5;
```

```
  int in3 = 7;
  int in4 = 6;
  void setup()
  {
                                    // ініціалізуємо всі піни для управління
двигунами як outputs
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  }
  void demoOne()
  {
                                // ця функція забезпечить обертання двигунів в двох
                                напрямках на встановленої швидкості
                                // запуск двигуна A
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
                                // встановлюємо швидкість 200 з доступного
діапазону
                                 0 ~ 255
  analogWrite(enA, 200);
                                // запуск двигуна B
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
                                // встановлюємо швидкість 200 з доступного
діапазону
                                 0 ~ 255
  analogWrite(enB, 200);
  delay(2000);
                                // міняємо напрям обертання двигунів
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  delay(2000);
                                // вимикаємо двигуни
```

```
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  }
  void demoTwo()
  {
        // ця функція забезпечує роботу двигунів у всьому
        діапазоні можливих швидкостей
                // зверніть увагу, що максимальна швидкість
визначається
                самим двигуном і напругою живлення
                // ШІМ-значення генеруються функцією analogWrite
()
                // і залежать від вашої плати управління
                // запускають двигуни digitalWrite (in1, LOW);
  digitalWrite(in2, HIGH);
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
                // прискорення від нуля до максимального значення
  for (int i = 0; i < 256; i++)
  {
  analogWrite(enA, i);
  analogWrite(enB, i);
  delay(20);
  }
                // гальмування від максимального значення до
                мінімального
  for (int i = 255; i >= 0; --i)
  {
  analogWrite(enA, i);
  analogWrite(enB, i);
  delay(20);
  }
                // тепер відключаємо мотори
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
```

```
}
void loop()
{
demoOne();
delay(1000);
demoTwo();
delay(1000);
}
```

Контрольні запитання:

1) Накресліть алгоритм роботи лістінгу № 1,2,3
2) Дайте визначення, що таке Кроковий двигун?
3) Які бувають крокові двигуни?
4) Дайте визначення, що таке двигун постійного струму?
5) Охарактеризуйте драйвер двигуна L293D
6) Порівняйте драйвера двигунів L298N та L293D

Лабораторна робота №7. Підключення сервоприводу до arduino

Мета роботи: Навчитися складати схеми підключення сервоприводу до ARDUINO та складати програму управління.

ХІД РОБОТИ

Для реалізації проекту нам потрібні наступні компоненти:

- контролер Arduino UNO R3;
- плата для прототипування;
- сервопривід;
- потенціометр 1 кОм;
- дроти;
- зовнішній блок живлення +5 В.

Сервопривід (див. рис. 17.1) - пристрій, що забезпечує перетворення сигналу в строго відповідне цим сигналом переміщення (як правило, поворот) виконавчого пристрою. Являє собою прямокутну коробку з мотором, схемою і редуктором всередині і вихідним валом, який може повертатися на строго фіксований кут, який визначається вхідним сигналом.

Як правило, цей кут має межу в 60 в 180. Крім цього, ще бувають сервоприводи і постійного обертання.

Рис. 7.1 – Сервопривід.

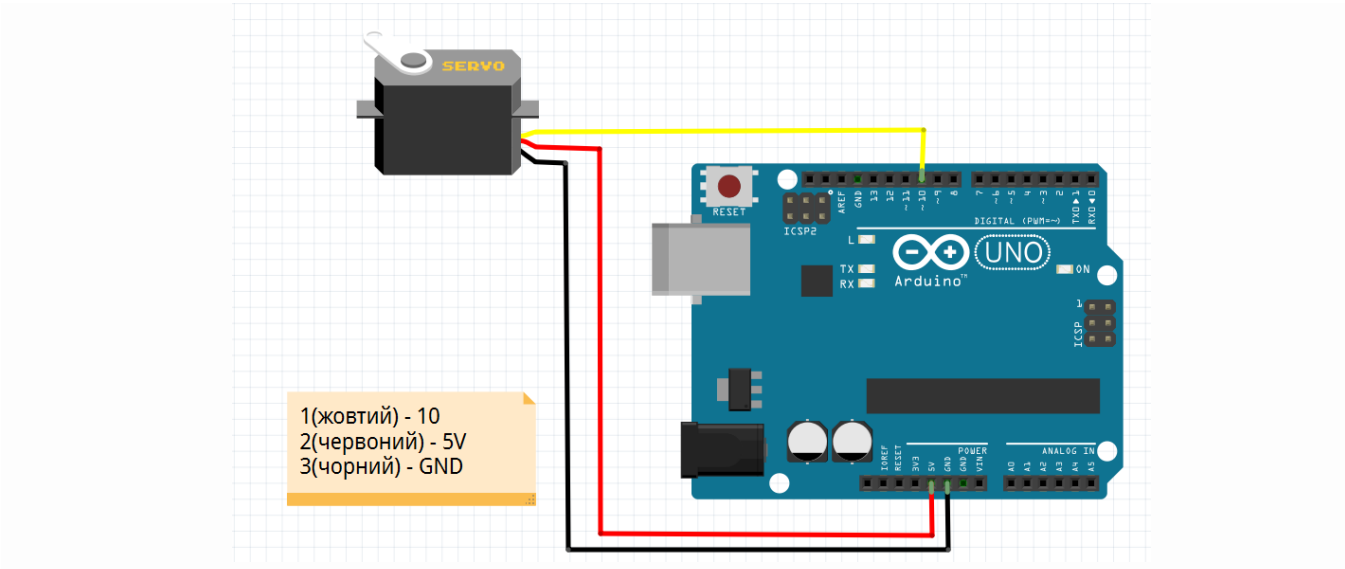


Рис.7.2 – Структурна схема лабораторної моделі.

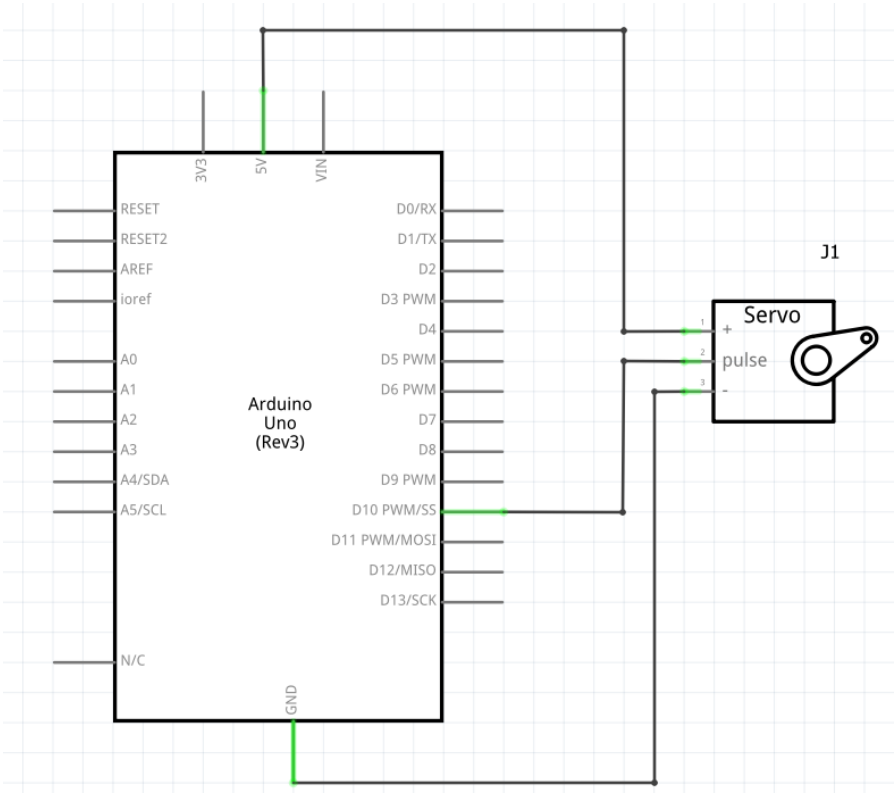


Рис.7.3 – Принципова електрична схема лабораторної моделі.

СКЕТЧ 1

```
#include <Servo.h> //используем библиотеку для работы с сервоприводом
Servo servo; //объявляем переменную servo типа Servo
void setup() //процедура setup
{
servo.attach(10); //привязываем привод к порту 10
}
void loop() //процедура loop
{
servo.write(0); //ставим вал под 0
delay(2000); //ждем 2 секунды
servo.write(180); //ставим вал под 180
delay(2000); //ждем 2 секунды
}
```

Сервопривод підключається за допомогою трьох проводів до пристрою (драйверу або контролера) і джерела живлення. Сервопривод управляється за допомогою імпульсів змінної тривалості. Кут повороту визначається тривалістю імпульсу, який подається по сигнальному проводу. Це називається широтно-імпульсною модуляцією. Сервопривод очікує імпульсу кожні 20 мс. Тривалість імпульсу визначає, наскільки далеко має повертатися мотор. Наприклад, імпульс в 1,5 мс диктує мотору поворот в положення 90 ° (нейтральне положення). Коли сервопривід отримує команду на переміщення, його керуючий орган переміщується в це положення і утримує його. Якщо зовнішня сила діє на сервопривід, коли він утримує задане положення, сервопривід буде чинити опір переміщенню з цього положення. Максимальна величина сили, яку може витримувати сервопривід, характеризує обертовий момент сервоприводу. Однак сервопривід НЕ назавжди утримує своє становище, імпульси позиціонування повинні повторюватися, інформуючи сервопривід про збереження положення.

У нашому експерименті ми будемо управляти положенням сервоприводу за допомогою потенціометра. Схема підключення сервоприводу і потенціометра до плати Arduino показана на рис. 17.2.

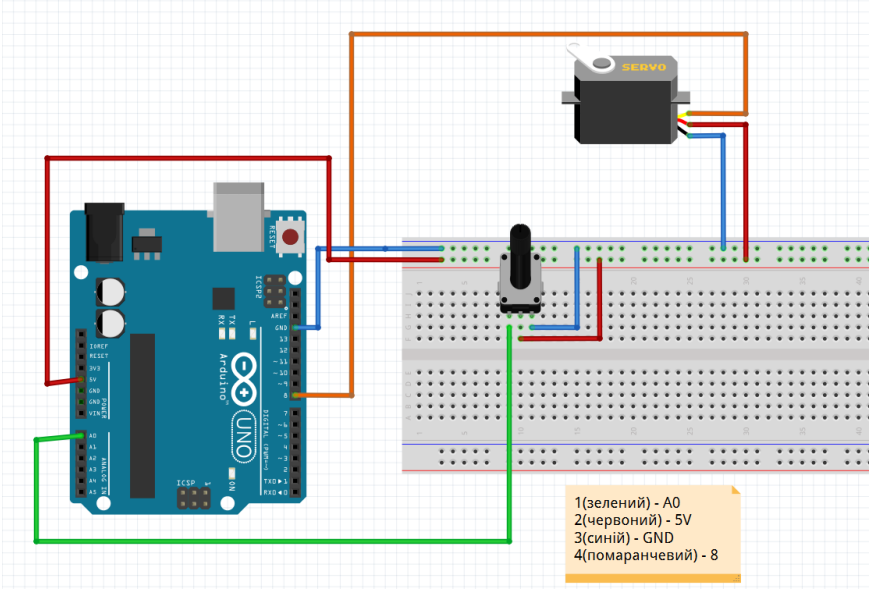


Рис. 7.4 - Схема підключення сервоприводу та потенціометра до Arduino.

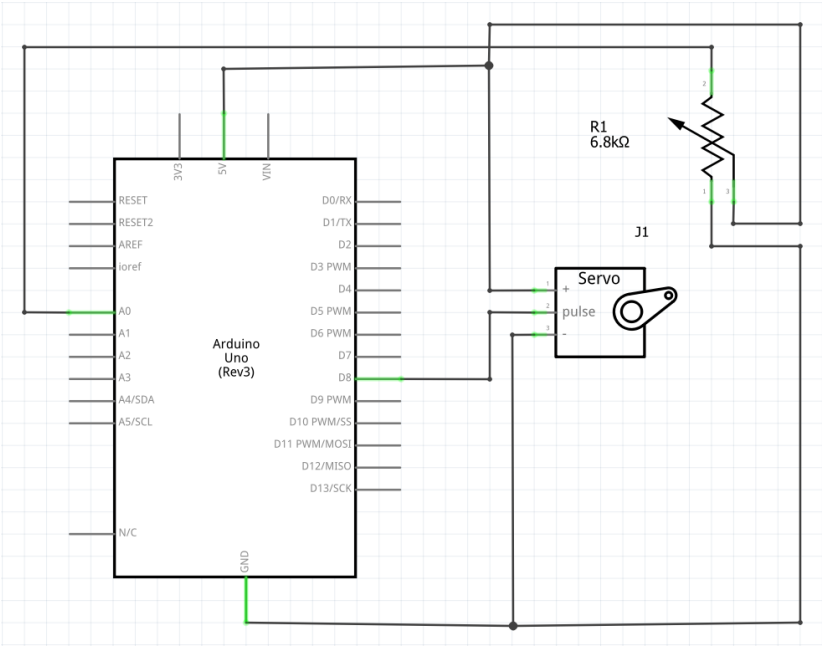


Рис.7.5 – Принципова електрична схема підключення сервоприводу та потенціометра до Arduino.

Сервопрівод підключається трьома проводами: харчування (Vcc), «земля» (Gnd) і сигнальний (С). Харчування червоний провід, він може бути підключений до +5 В зовнішнього джерела живлення, чорні (або коричневий) провід - «земля» - підключається до GND-висновку Arduino GND, сигнальний (помаранчевий / жовтий / білий) провід підключається до цифрового висновку контролера Arduino. Для харчування сервоприводу використовуємо окремий блок живлення +5 В. Для управління сервоприводом в Arduino є стандартна бібліотека Servo. На платах, відмінних від Mega, використання бібліотеки відключає можливість застосування analogWrite () (ШІМ) на пінах 9 і 10 (незалежно, підключені до цих пінам серви чи

ні). На платах Mega до 12 сервоприводів можуть використовуватися без впливу на функціональність ШІМ, але використання від 12 до 23 сервомашинок відключить PWM ШІМ на пінах 11 і 12.

Аналогові дані потенціометра (0-1023) масштабується функцією map () в значення кута повороту сервоприводу (0-180) і за допомогою бібліотечної функції servo.write (angle) даємо сервоприводу команду для повороту. Скетч приведений в лістингу.

СКЕТЧ 2

```
#include <Servo.h> // подключение библиотеки Servo
Servo servo1;
const int pinServo=8; // Пин для подключения сервопривода
const int POT=0; // Аналоговый вход A0 для подключения потенциометра
int valpot = 0; // переменная для хранения значения потенциометра
int angleServo = 0; // переменная для хранения угла поворота сервы
void setup()
{
// подключить переменную servo к выводу pinServo
servo1.attach(pinServo);
}
void loop()
{
valpot = analogRead(POT); // чтение данных потенциометра
// масштабируем значение к интервалу 0-180
angleServo=map(valpot,0,1023,0,180);
// поворот сервопривода на полученный угол
servo1.write(angleServo);
delay(15); // пауза для ожидания поворота сервопривода
}
```

Контрольні запитання:

1) Накресліть алгоритм роботи лістінгу № 1,2
2) Що таке потенціометр?
3) З чого складається сервопривід?

Лабораторна робота №8. Контроль оточуючого середовища за допомогою аналогових і дискретних датчиків.

Мета роботи: освоїти навички роботи з аналоговими та дискретними датчиками фізичних величин та управління силовими пристроями.

Елементи теорії.

Мікроклімат приміщення – це сукупність фізичних чинників та умов навколишнього середовища, які зумовлюють його тепловий стан і впливають на теплообмін людини. Основними і найбільш важливими чинниками, які формують мікроклімат приміщень, є температура та вологість повітря. Для забезпечення нормальної життєдіяльності людини необхідно створити комфортні умови всередині приміщення. Створювані умови залежать від багатьох чинників, таких, як: пора року, час доби, погодні умови поза приміщенням та ін. Корегування мікроклімату приміщень здійснюється за допомогою комплексних та спеціалізованих систем клімат-контролю.

Найпростіша система контролю мікроклімату включає у себе модулі вимірювання температури та вологості і в залежності від їх показань вмикаються системи кондиціонування, зволоження або осушення повітря.

Для контролю параметрів навколишнього середовища, таких, як температура та вологість, за допомогою Arduino можна використати багатофункціональний модуль DHT-11, зовнішній вигляд якого показаний на рис.8.1.

Рисунок 8.1 – Зовнішній вигляд датчика контролю температури і вологості DHT-11

Датчик має наступні робочі характеристики:

Відносна вологість	Температура
Роздільна здатність: 16 біт Діапазон вимірювання: 20 – 90% Точність: ±5% при 25 °C Гістерезис: < ±3% RH	Роздільна здатність: 16 біт Діапазон вимірювання: 0 – 50°C Точність: ±1°C Гістерезис: < ±0,2°C

Для контролю освітленості можна використати дискретний датчик на фоторезисторі (рис. 8.2), котрий реагує на величину світлового потоку, що діє на чутливий елемент.

Рисунок 8.2. – Зовнішній вигляд дискретного датчика освітленості на фоторезисторі

Підключення силових навантажень до Arduino моде здійснюватись багатьма свособами, наприклад через відкривання транзистора, тиристора, семістора або оптопати. Одним з найбільш простих і надійних рішень буде використання для даної цілі механічних реле, а між ними і схемою керування для захисту від можливого електричного пробою доцільно буде розмістити оптичну розв'язку.

Порядок виконання роботи.

1. Підключити до одного з цифрових входів (необхідно використовувати саме цифровий, а не аналоговий вхід, оскільки датчик має вбудований аналого-цифровий перетворювач) Arduino багатофункціональний датчик DHT-11 та датчик освітленості на фоторезисторі. Для програмної обробки даних з датчика температури і вологості необхідно підключити бібліотеку DHT та ініціалізувати датчик командою DHT dht(DHTPIN, DHTTYPE);, де DHTPIN – цифровий вхід до якого підключений сигнальний вихід датчика, DHTTYPE – тип датчика (у нашому випадку – DHT11). Програмний запуск датчика здійснюється командою dht.begin(); у тілі функції Setup. Для отримання значень температури та вологості в бібліотеці DHT описані функції dht.readHumidity(); та dht.readTemperature(); відповідно, які повертають значення типу float. Для обробки даних з датчика освітленості підключення додаткових бібліотек не потрібне, оскільки він на виході формує дискретний сигнал, а регулювання чутливості здійснюється розміщеним на платі змінним резистором.
2. Реалізувати виведення на символьний дисплей поточних значень температури та вологості, а також часу доби (день або ніч).
3. Здійснити увімкнення вентилятора при досягненні порогового значення температури денної – 20°C, нічної – 24°C. Для запобігання частого спрацювання системи кондиціонування необхідно передбачини гістерезис температури 2°C. Систему увімкнення вентилятора необхідно здійснити через реле, підключене до цифрового виходу Arduino.

Контрольні запитання:

1. Які датчики контролю фізичних параметрів навколишнього середовища можна підключити до Arduino?
2. Особливості обробки даних з аналогових та дискретних датчиків фізичних величин.
3. Технічні характеристики датчиків температури і вологості сімейства DHT (DHT-11, DHT-21, DHT-22).
4. Накресліть алгоритм роботи вашої програми.

Лабораторна робота №9. Підключення світлодіодного індикатора до arduino.

Мета роботи: Навчитися підключати та програмувати світлодіодний індикатор на arduino.

ХІД РОБОТИ

Мікросхема MAX7219 (MAX7221) призначена для управління семисегментными світлодіодними індикаторами.

Використання даного драйвера в електронних пристроях на мікроконтролері значно спрощує виведення інформації на індикатори. У динамічній Відпадає необхідність реалізації індикації, як наслідок економія процесорного часу, спрощення коду програми. Управління драйвером здійснюється по інтерфейсу SPI і для реалізації якого буде потрібно виділити всього 3 лінії вводу/виводу мікроконтролера, максимальна частота тактирования дорівнює 10 Мгц. Крім семисегментних індикаторів, з допомогою драйвера можна керувати світлодіодним матрицьою.

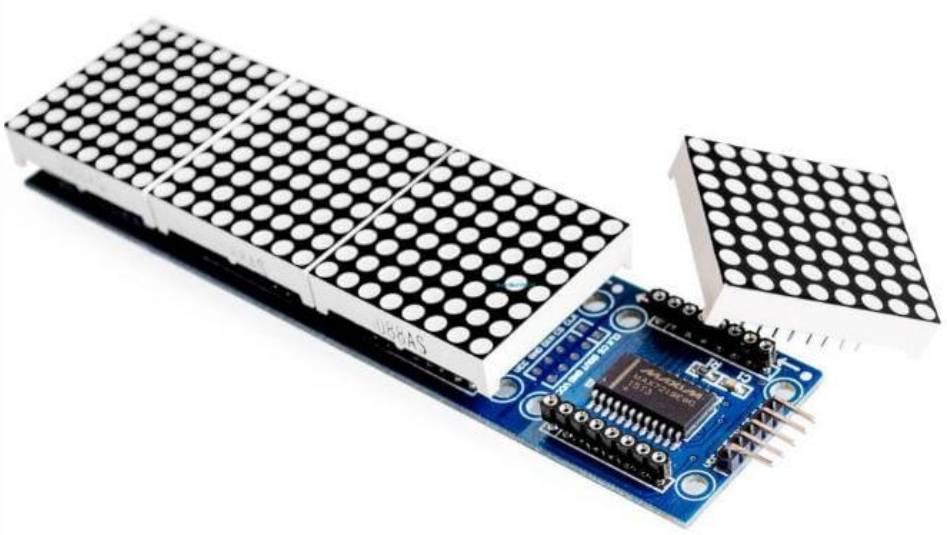


Рис.9.1 – Мікросхема MAX7219 (MAX7221) та семисегментні світлодіодні індикатори.

Arduino Uno - це пристрій на основі мікроконтролера ATmega328 (datasheet). У його склад входить все необхідне для зручної роботи з мікро контролером: 14 цифрових входів / виходів (з них 6 можуть використовуватися в якості ШІМ-виходів), 6 аналогових входів, кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрисхемного програмування (ICSP) і кнопка скидання. Для початку роботи з устройством досить просто подати живлення від AC / DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою USB-кабелю.

Arduino Uno програмується за допомогою програмного забезпечення Ардуіно .

Для цього з меню Tools> Board необхідно вибрати "Arduino Uno" з мікроконтролером, відповідним вашій платі.

В Arduino Uno є відновлювані запобіжники, що захищають USB-порт комп'ютера від коротких замикань і перевантажень. Незважаючи на те, що більшість комп'ютерів мають власний захист, такі запобіжники забезпечують додатковий рівень захисту.

Якщо від USB-порту споживається струм більше 500 мА, запобіжник автоматично розірве з'єднання до усунення причин короткого замикання або перевантаження.

На комп'ютері має бути встановлено програмне забезпечення для використовуваного Arduino і драйвер до нього.

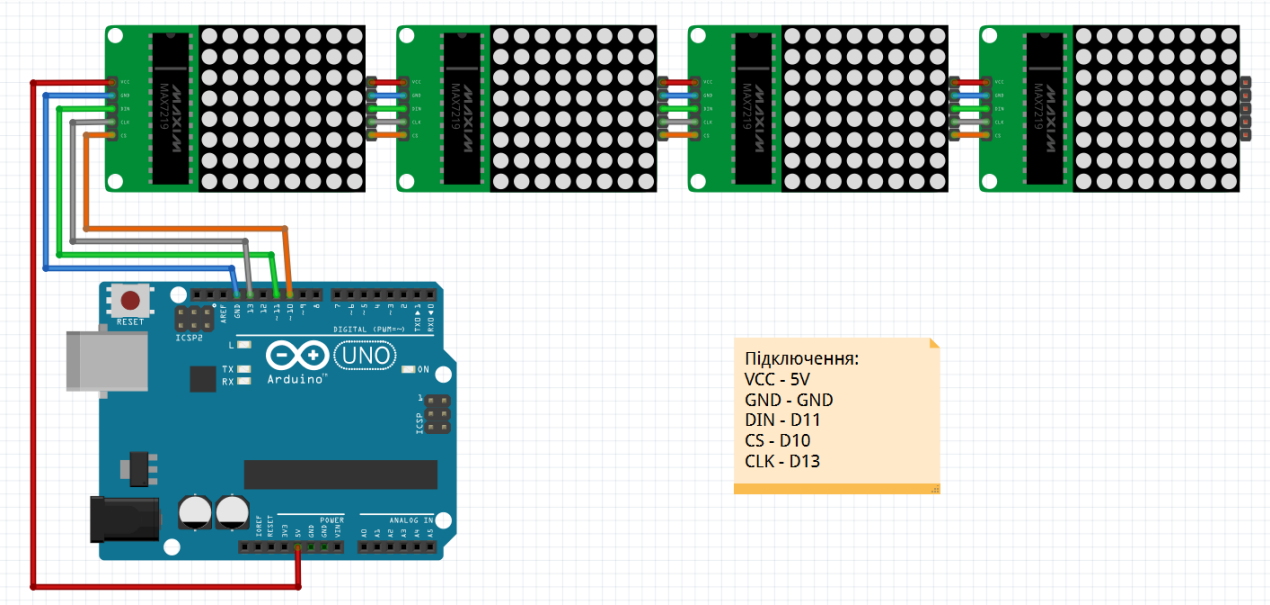


Рис.9.2 – Структурна схема лабораторної моделі.

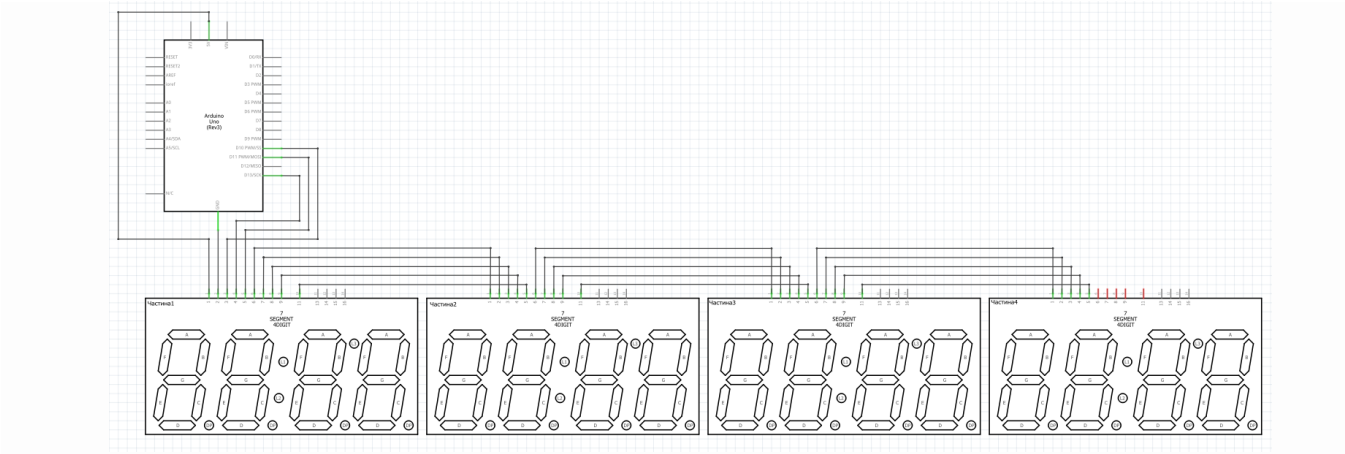


Рис.9.3 – Принципова електрична схема підключення матричних блоків один до одного.

Матричний модуль може мати штиркові з'єднання або контакти на платі у вигляді друкованих провідників. Від цього залежить спосіб їх з'єднання. У першому випадку для отримання надійного електричного контакту задіють джгут з дротів з коннекторами, а в другому доведеться встановити і запаяти перемички.

СКЕТЧ

```
#include <SPI.h>
#include <Adafruit_GFX.h>
#include <Max72xxPanel.h>

int pinCS = 9;
int numberOfHorizontalDisplays = 4; // теперь у нас по-горизонтали 4 матриц
int numberOfVerticalDisplays = 1; // а по-вертикали, по-прежнему, одна

Max72xxPanel matrix = Max72xxPanel(pinCS, numberOfHorizontalDisplays, numberOfVerticalDisplays);

String tape = "Kharkiv State Politechnical Colledge"; // текст, который будет плыть
int wait = 40; // время между крайними перемещениями букв

int spacer = 1; // расстояние между буквами
int width = 5 + spacer; // размер шрифта

void setup() {
   matrix.setIntensity(7); // яркость
matrix.setRotation( 0, 1 );
matrix.setRotation( 1, 1 );
```

```
matrix.setRotation( 2, 1 );
matrix.setRotation( 3, 1 );
}

void loop() {
   for ( int i = 0 ; i < width * tape.length() + matrix.width() - 1 - spacer; i++ ) {
      matrix.fillScreen(LOW);

      int letter = i / width;
      int x = (matrix.width() - 1) - i % width;
      int y = (matrix.height() - 8) / 2; // center the text vertically

      while ( x + width - spacer >= 0 && letter >= 0 ) {
         if ( letter < tape.length() ) {
            matrix.drawChar(x, y, tape[letter], HIGH, LOW, 1);
         }
         letter--;
         x -= width;
      }

      matrix.write();
      delay(wait);
   }
}
```

Контрольні запитання:

1) Накресліть алгоритм роботи лістінгу № 1,2
2) Що таке мікросхема MAX7219?
3) Якою командою ми можемо змінити напрям тексту ?

Лабораторна робота №10. Мережевий обмін за допомогою Arduino.

Мета роботи: Дослідити роботу Arduino Mega 2560 з використанням Ethernet shield

Теоретичні відомості

Arduino Ethernet shield (рис. 8.1) дозволяє підключити плату Arduino до ме- режі. Вона заснована на ethernet-мікросхемі Wiznet W5100. Wiznet W5100 підтри- мує стеки TCP і UDP в Ip-Мережі, а також до чотирьох одночасних підключень до сокетів. Для створення скетчів, які підключаються до мережі за допомогою даної плати, використовується бібліотеку Ethernet Library . Плата Ethernet shield з'єдну- ється із платою Arduino за допомогою довгих штирьків, що проходять через неї. Це дозволяє не змінювати розташування виводів і встановлювати інші плати по- верх неї (рис. 8.1).

Рисунок 10.1 – Arduino Ethernet shield

Плата Ethernet shield має стандартне з'єднувач RJ-45 із вбудованим лінійним трансформатором і опцією POE (Power Over Ethernet) для одержання живлення від звичайної витої пари 5 категорії. Останні версії плати мають також з'єднувач для карт типу microSD, які можуть використовуватися для зберігання файлів і ро- боти з ними по мережі. Плата сумісна з Arduino Uno і Mega (при використанні бі-

бліотеки Ethernet Library). Картрідер microSD доступний за допомогою бібліоте- ки SD Library . При застосуванні цієї бібліотеки вивід 4 використовується для си- гналу SS (Slave Select).

Останні версії плати також мають контролер скидання, який дозволяє бути упевненим у правильному перезапуску W5100 при запуску. Попередні версії плати були не сумісні з Arduino Mega і вимагали ручного скидання після вмикання. Попередня версія плати мала з'єднувач для повнорозмірної карти SD, що зараз не підтримується.

Шестиконтактний з'єднувач для послідовного програмування сполучимо з кабелями й платами-перехідниками FTDI-USB. Він підтримує автоматичне скидання, що дозволяє завантажувати скетчі без натискання кнопки скидання на платі. При підключенні через адаптер FTDI-USB, Arduino і Ethernet shield одержують живлення від адаптера.

Версія плати підтримує підключення адаптера Power Over Ethernet (POE) для одержання живлення від звичайної витої пари 5 категорії, а також забезпечує:

- сумісність із EEE802.3af;
- низькі пульсації й шум на виході (100m Vpp);
- діапазон вхідної напруги від 36 до 57 В;
- захист від перевантаження й короткого замикання;
- вихідна напруга 9 В;
- високоефективний DC/ DC-Перетворювач: 75 % при навантаженні в 50%

rduino здійснює зв'язок з W5100 і картою SD за допомогою шини SPI (че-

A

рез з'єднувач ICSP header). Він розташований на виводах 11, 12 і 13 плати Duemilanove і виводах 50, 51 і 52 плати Mega. На обох платах вивід 10 використовується для вибору W5100 і ввод 4 – для карти SD. Ці контакти не можуть бути використані для іншого вводу/виводу. На платі Mega апаратний вивод SS 53 не використовується для вибору ні W5100, ні карти SD, але він повинен бути сконфігурований на вивід, інакше інтерфейс SPI не буде працювати.

Оскільки W5100 і карта SD розділяють шину SPI, одночасно працювати во- ни не можуть. Якщо ви використовуєте обидва цих периферійних пристрої у своїй програмі, вам слід подбати про відповідні бібліотеки. Якщо ви не використовуєте одне із цих периферійних пристроїв, вам слід відключити його. Щоб зробити це, сконфігуруйте вивід плати 4 як вихід і запишіть у нього "1". Для W5100 встановіть на виводі 10 високий рівень.

Кнопка скидання плати перезапускає й дочірню плату, і плату Arduino. Плата має декілька індикаторних світлодіодів:

- PWR – індикація наявності живлення плати;
- LINK – індикація наявності мережного лінка, блимання при відправленні або одержанні даних;
- FULLD – індикація повнодуплексного з'єднання;
- 100M – індикація з'єднання на швидкості 100 Мбіт/с (на відміну від з'єднання на 10 Мбіт/с);
- RX – мигає при одержанні платою даних;
- TX – мигає при відправленні платою даних;
- COLL – мигає при мережній колізії.

Перемичка, що запаюється, "INT" може бути замкнена, що дозволить платі Arduino одержувати повідомлення (через переривання) про події від W5100, але зараз це не підтримується бібліотекою Ethernet Library. Перемичка з'єднує вивід INT мікросхеми W5100 і цифровий вивід 2 плати Arduino.

Бібліотека Ethernet Library

Для роботи з Ethernet shield використовується стандартна Arduino- Бібліотека Ethernet Library . Вона поставляється в складі дистрибутива Arduino. Розглянемо її докладніше.

Клас Ethernet (Ethernet class)

Клас Ethernet ініціює бібліотеку й мережні налаштування. Починаючи з ве- рсії 1.0, бібліотека підтримує DHCP. Використання Ethernet.begin (mac) дозволяє автоматично одержувати IP-Адресу.

Функції класу Ethernet:

- begin() ;

- localip() .

Функція Ethernet.begin()

Функція Ethernet.begin() ініціює бібліотеку й мережні налаштування. Синтаксис функції:

Ethernet.begin(mac) Ethernet.begin(mac,ip) Ethernet.begin(mac,ip,gateway) Ethernet.begin(mac, ip, gateway, subnet) Параметри:

- mac – mac-адреса (Media Access Control) пристрою (масив з 6 байтів). Це апаратна адреса адаптера вашого пристрою. Нові Arduino Ethernet shield включають наклейку з Mac-Адресою пристрою. Для більш старих його потрібно знайти самостійно;
- ip– IP-Адреса пристрою (масив з 4 байтів);
- gateway – IP-Адреса мережного шлюзу (масив з 4 байтів). Опція – за замовчуванням IP-Адреса пристрою з останнього октету встановлений в 1;
- subnet – маска підмережі (масив з 4 байтів). Опція – за замовчуванням 255.255.255.0.

Значення, що повертається: немає.

Функція Ethernet.localIP()

Функція Ethernet.localIP()повертає IP-Адресу Ethernet shield. Синтаксис функції:

Ethernet.localIP()

Значення, що повертається: Ip-Адреса пристрою.

Клас IPAddress (IPAddress class)

Клас IPAddress працює з локальними й віддаленими IP-Адресами.

Функція IPAddress()

Функція IPAddress() визначає IP-Адресу. Вона може бути використана для оголошення локальних і віддалених адрес.

Синтаксис:

IPAddress(address)

Параметр: address– призначена IP-Адреса. Значення, що повертається: немає.

Клас Server (Server class)

Клас Server створює сервер, який може передавати дані й одержувати дані від підключених клієнтів.

Функції класу Server:

- ethernetServer() ;
- begin() ;
- available() ;
- write() ;
- print() ;
- println() .

Функція ethernetServer()

Функція ethernetServer() створює сервер, який прослуховує вхідні з'єднання на зазначений порт.

Синтаксис:

ethernetServer (port)

Параметр: port– призначуваний для прослуховування порт.

Функція begin()

Функція begin()запускає прослуховування вхідних з'єднань. Синтаксис:

server.begin() Параметрів немає. **Функція available()**

Функція available() одержує посилання від підключеного до сервера клієнта для обміну даними.

Синтаксис: server. available()

Параметрів немає.

Значення, що повертається: посилання на підключений до сервера клієнт (EthernetClient client = server.available();).

Приклад використання функції available()представлено в лістингу 8.1.

Лістинг 1

```
#include <Ethernet.h> #include <SPI.h>
// mac-адреса Ethernet shield:

byte mac[] = { 0xde, 0xad, 0xbe, 0xef, 0xfe, 0xed };

// Ip-Адреса, призначена Ethernet shield: byte ip[] = { 192, 168, 0, 177 };
// адреса шлюзу:

byte gateway[] = { 192, 168, 0, 1 };

// маска:

byte subnet[] = { 255, 255, 0, 0 };

// порт для сервера

EthernetServer server = EthernetServer(23); void setup()
{

// ініціалізація Ethernet shield Ethernet.begin(mac, ip, gateway, subnet);
// запуск сервера server.begin();
}

void loop()

{

// одержання посилання від підключеного клієнта EthernetClient client = server.available();
// поки підключення активне if (client == true) {
// читаємо дані, що відправляються клієнтом,

// і відправляємо їх назад клієнту: server.write(client.read());
}

}
```

Функція write()

Функція write() відправляє дані всім клієнтам, підключеним до сервера. Синтаксис:

server.write(data)

Параметр: data– дані, що відправляються (байт або символ).

Функція print()

Функція print() відправляє дані всім клієнтам, підключеним до сервера.

При цьому число відправляється як послідовність цифр даного числа.

Синтаксис:

server.print(data) server.print(data,BASE) Параметри:

- data– дані, що відправляються (байт, символ, число, рядок).
- BASE– вид переданих символів:
  - BIN– двійковий;
  - DEC– десятковий;
  - OCT– восьмеричний;
  - HEX– шістнадцятеричний.

Значення, що повертається: кількість переданих байтів. Функція println() Функція println() відправляє дані всім клієнтам, підключеним до серве-

ра.

Аналогічна функції print() з додаванням у відправлення символу пере-

ходу на новий рядок.

Синтаксис:

server.println() server.println(data) server.println(data,BASE) Параметри:

- data– дані, що відправляються (байт, символ, число, рядок).
- BASE– вид переданих символів:
  - BIN– двійковий;
  - DEC– десятковий;
  - OCT– восьмеричний;
  - HEX– шістнадцятеричний.

Значення, що повертається: кількість переданих байтів.

Клас Client (Client class)

Клас Client створює клієнтів, які можуть підключатися до серверів і обмінюватися даними.

Функції класу Client:

- client ;
- EthernetClient() ;

- connected() ;

-connect() ;

- write() ;
- print() ;
- println() ;
- available() ;
- read() ;
- flush() ;
- stop() .

Функція client()

Функція client() створює клієнта, який підключається до сервера по за- значеним в параметрах IP-Адресі й порту.

Синтаксис: client(ip, port)

Параметри:1) ip– Ip-Адреса підключення клієнта (масив з 4 байтів);

2) port– порт підключення клієнта.

Функція EthernetClient()

Функція EthernetClient() створює клієнта, який підключається до сервера. Параметри підключення (Ip-Адреса й порт) визначаються у функції client.connect()) .

Синтаксис: EthernetClient()

Параметрів немає. Значення, що повертається: немає.

Функція connected()

Функція connected() визначає, підключений клієнт чи ні. Клієнт вважа- ється підключеним, якщо з'єднання закрите, але є непрочитані дані.

Синтаксис: client.connected()

Параметрів немає. значення, що повертаються: true – якщо клієнт підклю- чений, і false– якщо немає.

Функція connect()

Функція connect() підключається до сервера по зазначеним IP-Адресі й порту.

Синтаксис: client.connect()

Параметри: 1) ip– IP-Адреса підключення клієнта (масив з 4 байтів);

2) port– порт підключення клієнта.

Значення, що повертаються: true – якщо клієнт підключений, і false – якщо ні.

Функція write()

Функція write()відправляє на сервер дані з підключеного клієнта. Синтаксис:
client.write(data)

Параметр: data – дані, що відправляються (байт або символ). Значення, що повертається: немає.

Функція print()

Функція print() відправляє дані на сервер з підключеного клієнта. При

цьому число відправляється як послідовність цифр даного числа.

Синтаксис:

client.print(data) client.print(data,BASE) Параметри:

- data– дані, що відправляються (байт, символ, число, рядок).
- BASE– вид переданих символів:
  - BIN– двійковий;
  - DEC– десятковий;
  - OCT– восьмеричний;
  - HEX– шістнадцятеричний.

Значення, що повертається: кількість переданих байтів.

Функція println()

Функція println() відправляє дані на сервер з підключеного клієнта. Аналогічна функції print() з додаванням у відправлення символу переходу на новий рядок.

Синтаксис:

client.println() client.println(data) client.println(data,BASE) Параметри:

- data– дані, що відправляються (байт, символ, число, рядок).
- BASE– вид переданих символів:
  - BIN– двійковий;

- DEC– десятковий;
- OCT– восьмеричний;
- HEX– шістнадцятеричний.

Значення, що повертається: кількість переданих байтів.

Функція available()

Функція available() повертає число байтів, доступних для читання (тобто обсяг даних, який був відправлений сервером для клієнта).

Синтаксис: client.available()

Параметрів немає. Значення, що повертається: кількість доступних для читання байтів.

Функція read()

Функція read() одержує наступний (після останнього, отриманого коман- дою read()) байт від сервера або –1, якщо даних більше немає.

Синтаксис: client.read()

Параметрів немає. Значення, що повертається: байт або –1.

Функція flush()

Функція flush() скидає отримані від сервера, але ще не прочитані дані. Синтаксис: client.flush()

Параметрів немає. Значення, що повертається: немає.

Функція stop()

Функція stop()відключає клієнта від сервера. Синтаксис: client.stop()

Параметрів немає. Значення, що повертається: немає.

Клас EthernetUDP (EthernetUDP class)

Клас EthernetUDP дозволяє відправляти й одержувати UDP-Повідомлення. Функції класу EthernetUDP:

- begin() ;
- read() ;
- write() ;
- beginPacket() ;
- endPacket() ;

- parsePacket() ;
- available() ;
- remoteIP() ;
- remotePort() .

Функція begin()

Функція begin() ініціює бібліотеки Ethernet UDP і мережні параметри. Синтаксис: client.begin(port)

Параметр: port – локальний порт для прослуховування. Значення, що повертається: немає.

Функція read()

Функція read() читає UDP-Дані із зазначеного буфера. Якщо аргументи не задані, то вона поверне наступний символ у буфері.

Синтаксис: 1) UDP.read()

2) UDP.read(packetbuffer,maxsize)

Параметри:

1) packetbuffer– буфер для зберігання вхідних пакетів;

2) maxsize – максимальний розмір буфера. Значення, що повертається: символи з буфера. **Функція write()**

Функція write() записує UDP-Дані у віддалене з'єднання. Дані повинні бути укладені між beginPacket() і endPacket(). beginPacket()ініціює пакет даних, дані не відправляються до виклику endPacket().

Синтаксис: UDP.write(message)

Параметр: message – повідомлення, що відправляється. Значення, що повертається: кількість символів, що вертаються.

Приклад використання функції write()представлено в лістингу 8.2.

Лістинг 2

```
// MAC-адреса Ethernet shield:
byte mac[] = { 0xde, 0xad, 0xbe, 0xef, 0xfe, 0xed };
// IP-Адреса в локальній мережі:
IPAddress ip(192, 168, 1, 177);
// порт
```

```
unsigned int localport = 8888; EthernetUDP Udp;
void setup() {

// запустити Udp-З'єднання Ethernet.begin(mac,ip); Udp.begin(localport);
}

void loop() {

Udp.beginPacket(Udp.remoteIP(), Udp.remotePort()); Udp.write("hello");
Udp.endPacket();

}
```

Функція beginPacket()

Функція beginPacket() ініціює Udp-Підключення на відправлення да- них у віддалене з'єднання.

Синтаксис: UDP. beginPacket (remoteip, port)

Параметри:1) remoteip – Ip-Адреса віддаленого з'єднання;

2) port – порт віддаленого з'єднання.

Функція endPacket()

Функція endPacket() викликається після запису Udp-Даних вилученого з'єднання.

Синтаксис: UDP. endPacket()

Параметрів немає. Значення, що повертається: немає.

Функція parsePacket()

Функція parsePacket() перевіряє наявність пакета UDP і його розмір. Функція parsePacket() повинна викликатися перед читанням буфера з UDP.read() .

Синтаксис: UDP. parsePacket()

Параметрів немає. Значення, що повертається: розмір отриманих даних. Приклад використання функції parsePacket() представлено в лістингу
8.3.

Лістинг 3

```
// MAC-адреса Ethernet shield

byte mac[] = { 0xde, 0xad, 0xbe, 0xef, 0xfe, 0xed };

// ip-адреса

IPAddress ip(192, 168, 1, 177);
```

```
// порт
unsigned int localport = 8888; EthernetUDP Udp;
void setup() {
// запустити Udp-З'єднання Ethernet.begin(mac,ip); Udp.begin(localport);
Serial.begin(9600);
}
void loop() {
// одержати розмір даних
int packetsize = Udp.parsePacket(); if(packetsize)
{
Serial.print("size="); Serial.println(packetsize);
}
delay(10);
}
```

Функція available()

Функція available() одержує кількість даних, доступних для читання з буфера.

Синтаксис: UDP. available()

Параметрів немає. Значення, що повертається: кількість байтів, доступних для читання.

Функція remoteIP()

Функція remoteIP() одержує IP-Адресу вилученого з'єднання. Ця функ- ція повинна бути викликана після UDP.parsePacket().

Синтаксис: UDP. remoteIP()

Параметрів немає. Значення, що повертається: IP-Адреса вилученого з'єд- нання (4 байта).

Функція remotePort()

Функція remotePort() одержує IP-Адресу вилученого з'єднання UDP. Синтаксис: UDP. remotePort()

Параметрів немає. Значення, що повертається: порт з'єднання з вилученим хостом.

Опис лабораторної установки

До складу лабораторної установки входить:

- Персональний комп’ютер;
- Arduino Mega 2560;
- адаптер на базі контролера W5100;
- блок живлення;
- макетна плата;
- набір резисторів;
- набір провідників.

Порядок виконання роботи

1. Під’єднати Ethernet shield до плати Arduino.
2. Визначити MAC адрес плати визначити IP адресу та встановити з’єднання між сервером та клієнтом.
3. Отримати у викладача завдання щодо під’єднання датчиків та передачі отриманих даних через створену мережу.

Контрольні запитання

1. Використання Ethernet shield для передачі даних та створення мережі..
2. Що таке «клієнт» і «сервер».
3. Як передати дані від клієнта до сервера і навпаки.
4. Використання бібліотек Arduino для створення мережі передачі даних.

Зміст