Практична робота 8. Робота з сервоприводом Arduino

Теоретичні відомості

В певний момент часу, при освоєнні середовища Arduino IDE виникає необхідність розширити можливості за допомогою використання додаткових бібліотек.

Бібліотека – це набір функцій, призначених для того, щоб максимально спростити роботу з різними датчиками, РК-екранами, модулями та ін. Наприклад, вбудована бібліотека LiquidCrystal дозволяє доволі просто взаємодіяти з символьними LCD-дисплеями. Існують сотні додаткових бібліотек, які можна завантажити з мережі Інтернет. Але перед тим, як використовувати додаткові бібліотеки, необхідно спершу їх встановити.

Встановлення сторонніх бібліотек

Найчастіше бібліотеки доступні у вигляді ZIP-архіву або просто папки. Назва цієї папки є назвою бібліотеки. Всередині папки буде файл з розширенням .cpp, файл з розширенням .h, а також текстовий файл keywords.txt, папка з прикладами examples та інші файли, необхідні бібліотеці.

Починаючи з версії Arduino IDE 1.0.5, встановлювати сторонні бібліотеки можна прямо в середовищі розробки. Розпаковувати при цьому архів не потрібно.

В середовищі розробки потрібно вибрати пункт меню *Скетч – Додати бібліотеку – Додати .ZIP-бібліотеку...* (рис. 8.1).

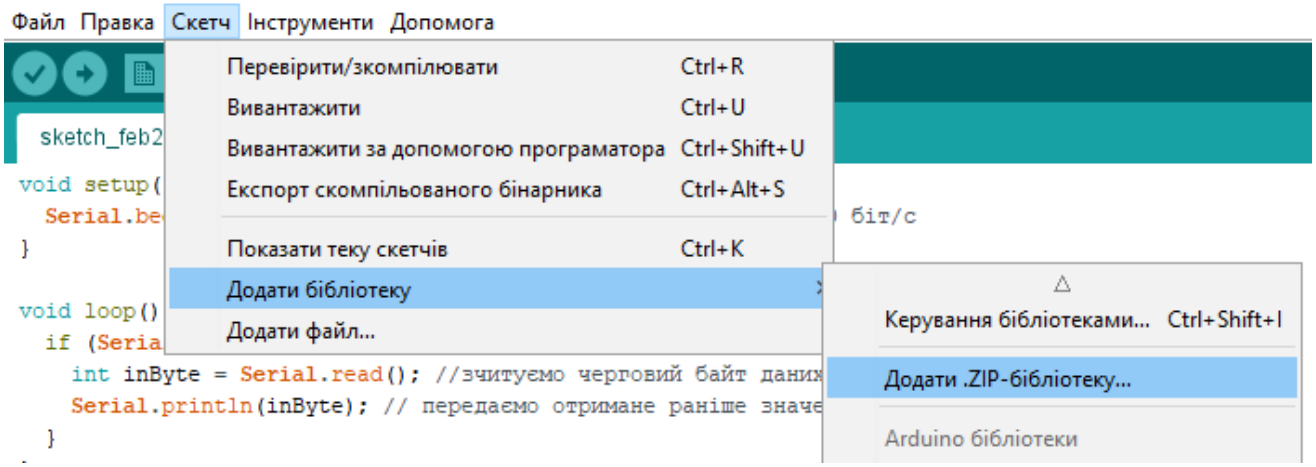


Рисунок 8.1 – Додавання сторонньої бібліотеки в середовище Arduino IDE

З'явиться діалогове вікно, яке запропонує вибрати бібліотеку, яку потрібно додати. Потрібно лише перейти до завантаженого zip-файлу та відкрити його.

Якщо тепер знову відкрити меню *Скетч – Додати бібліотеку*, додана бібліотека вже буде в самому низу списку. Тепер бібліотеку можна використовувати в програмах.

Після перезавантаження середовища приклади з встановленої бібліотеки з'являться в меню *Файл – Приклади*.

Сервоприводи

Сервопривід – це двигун, положенням валу якого можна керувати (рис. 8.2). Від звичайного двигуна він відрізняється тим, що йому можна точно в градусах задати положення, в яке стане вал. Сервоприводи використовуються для моделювання різних механічних рухів.

Конструкція сервоприводу складається з двигуна, датчика позиціонування і керуючої системи (рис. 8.3). Сервоприводи нерідко використовуються в таких сферах як обробка матеріалів, виробництво транспортного устаткування, обробка деревини, виготовлення металевих листів, виробництво будматеріалів та інші.

Рисунок 8.2 – Сервоприводи

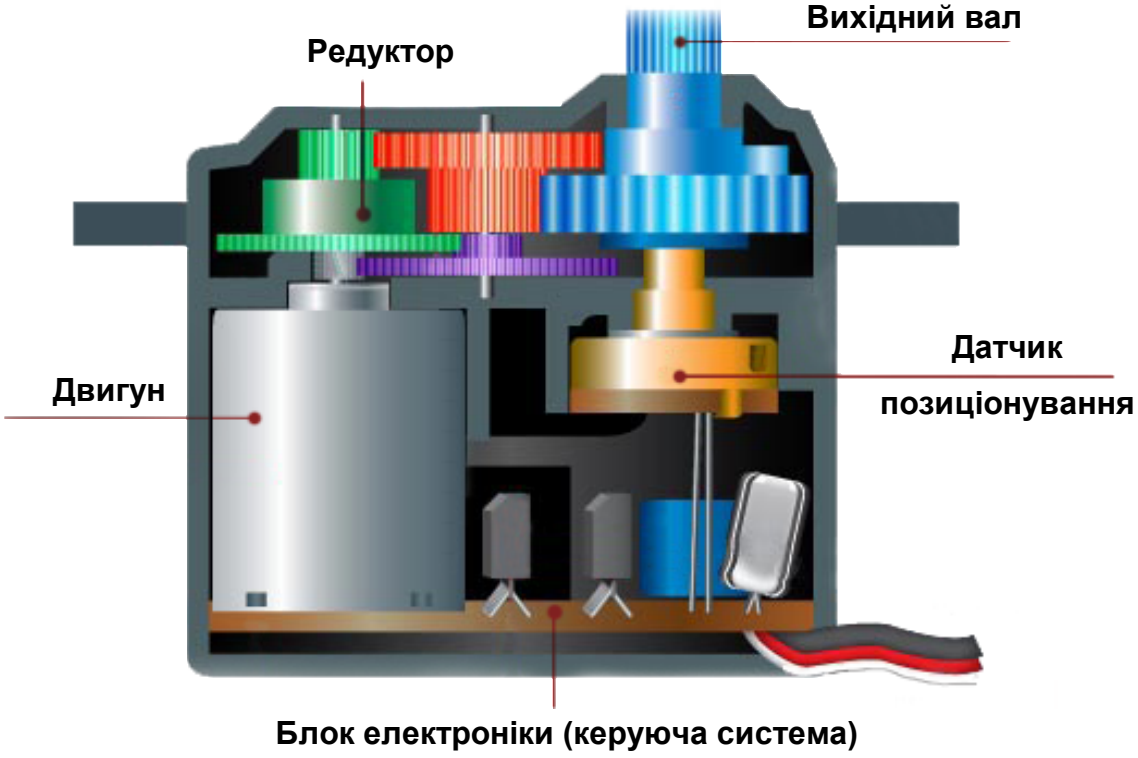


Рисунок 8.3 – Конструкція сервоприводу

Принцип роботи сервоприводу заснований на зворотному зв'язку з одним або декількома системними сигналами. Найпростішим варіантів реалізації датчика позиціонування є потенціометр, який керується вихідним валом – при зміні параметрів резистора змінюються параметри струму, що живить двигун.

Виділяють два основних види серводвигунів – з безперервним обертанням та з фіксованим кутом (найчастіше, 180 або 270 градусів). Відмінність

сервоприводів обмеженого обертання полягає в механічних елементах конструкції, які можуть блокувати рух валу поза межами заданих параметрами кутів. Досягнувши кута 180°, вал вплине на обмежувач, а той «віддасть» команду на вимикання двигуна. У серводвигунів безперервного обертання таких обмежувачів немає.

Широке використання сервоприводів пов'язано з тим, що вони володіють стабільною роботою, високою стійкістю до перешкод, малими габаритами і широким діапазоном регулювання швидкості. Важливими особливостями сервоприводів є здатність збільшувати потужність і забезпечення зворотного інформаційного зв'язку.

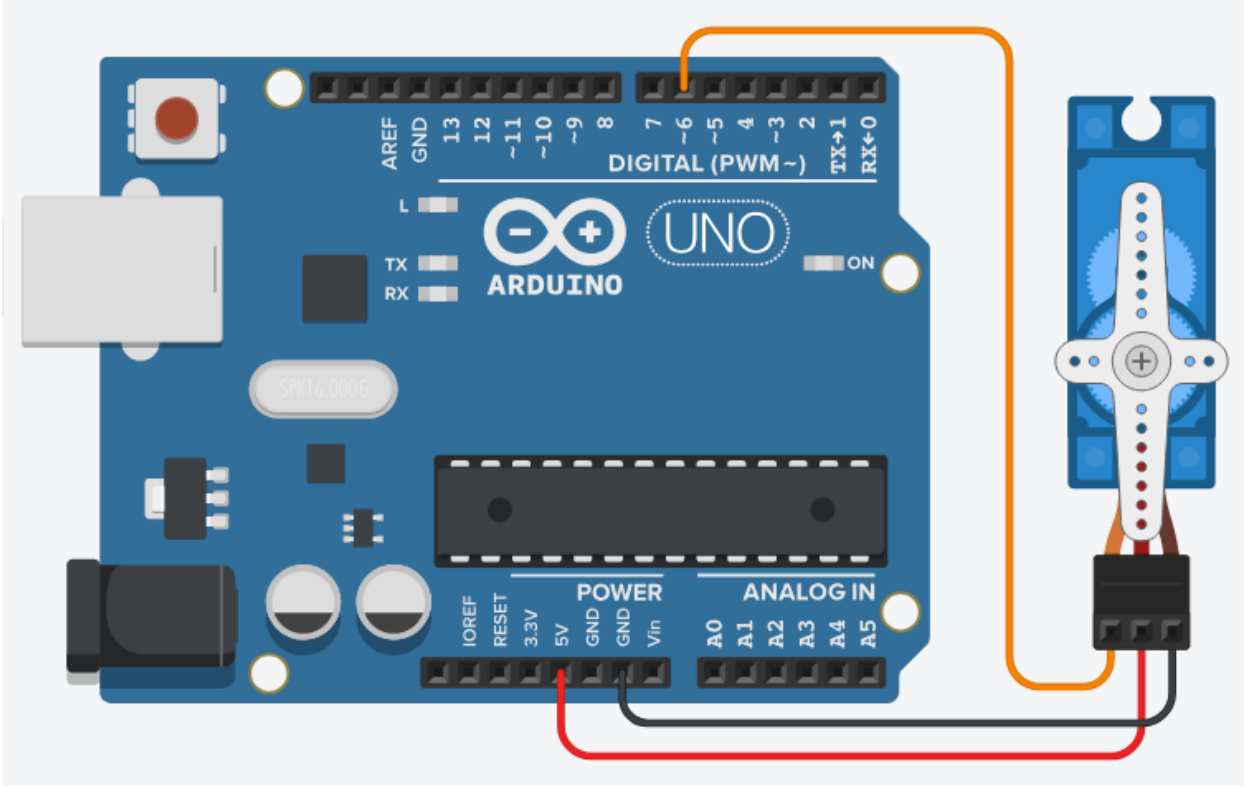


Рисунок 8.4 – Схема підключення сервопривода до Arduino

Зазвичай сервопривід має три контакти різних кольорів. Коричневий провід підключається до землі, червоний – до живлення +5В, провід оранжевого або жовтого кольору – сигнальний. До Arduino сервопривід підключається як

показано на рисунку 8.4. Помаранчевий провід (сигнальний) підключається до цифрового ШІМ-піну.

Для того, щоб спростити керування сервоприводом за допомогою Arduino, використовується вбудована бібліотека Servo. Розглянемо простий приклад використання Servo:

Лістинг 8.1 – Керування сервоприводом з використанням бібліотеки Servo

```
#include <Servo.h>          // підключення бібліотеки Servo

Servo servo;                // створення об'єкту класу Servo

void setup() {
  servo.attach(6);          // вибір піна підключення сервопривода
  servo.write(0);           // встановлення початкового положення 0°
  }

void loop() {
  servo.write(90);          // поворот валу на 90°
  delay(1000);              // затримка 1 секунду
  servo.write(135);         // поворот валу на 135°
  delay(100);
  servo.write(180);         // поворот валу на 180°
  delay(1000);
  servo.write(0);
  delay(1000);
}
```

кінець лістингу 8.1

Директива #include

#include використовується для підключення сторонніх бібліотек в код програми. Це дає доступ до великої кількості стандартних бібліотек C (бібліотекою називають групи попередньо написаних функцій), і бібліотек написаних спеціально для Arduino.

Варто звернути увагу на те, що #include, так само як і #define, не вимагає крапки з комою в кінці рядка, якщо ж її додати компілятор видасть критичну помилку.

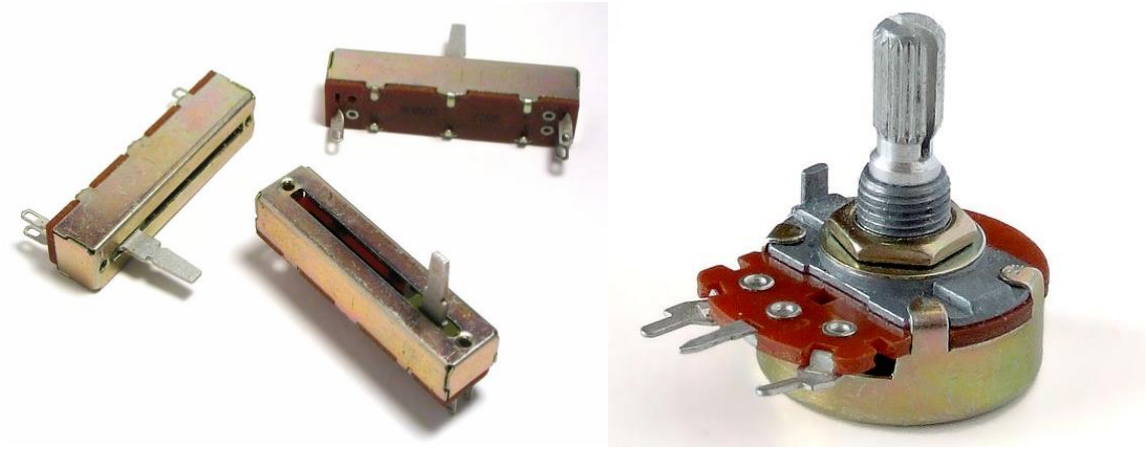
Бібліотека Servo

Ця бібліотека надає набір функцій для керування сервоприводами. Стандартні сервоприводи дозволяють повертати привід на визначений кут від 0 до 180 градусів, зазвичай. Деякі сервоприводи дозволяють здійснювати повні оберти на заданій швидкості.

Функції:

- *attach(pin)* – підключає Servo до зазначеного виходу, з якого здійснюється керування приводом;
- *write(angle)* – передає значення для керування приводом. Для стандартного сервоприводу це кут повороту. Для приводу постійного обертання, функція задає швидкість обертання (0 – для максимальної швидкості обертання в одну сторону, 180 – для максимальної швидкості в іншу сторону і близько 90 для нерухомого стану);
- *read()* – зчитує значення поточного положення сервоприводу.

Потенціометр – це резистор з трьома виходами, один із яких рухомий, що використовується, як дільник напруги (рис. 8.5).



a) b)

Рисунок 8.5 – Лінійний (a) та типовий (b) потенціометр

Електричний струм проходить між кінцевими контактами, а потрібна споживачеві напруга знімається з повзунка, положення якого можна механічно змінювати (рис. 8.6).

Потенціометри використовуються в якості регуляторів параметрів (гучності звуку, потужності, вихідної напруги і т.д.). Також потенціометри можна використовувати як змінний резистор, задіюючи лише два виходи – один із кінцевих і повзунок.

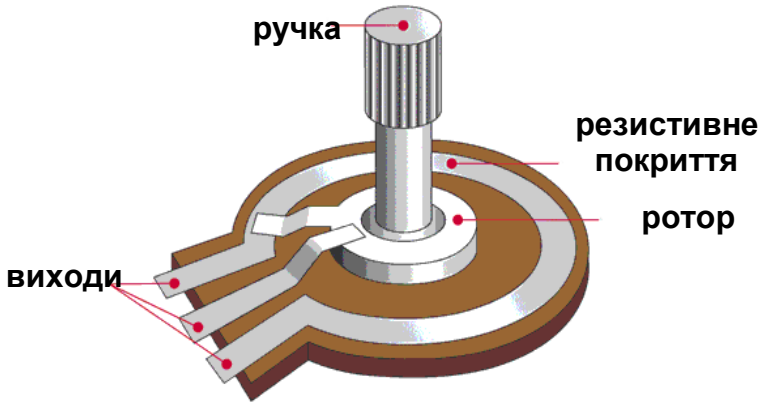


Рисунок 8.6 – Будова типового потенціометра

Підключення потенціометра до Arduino виконується відповідно до схеми, представленої на рисунку 8.7:

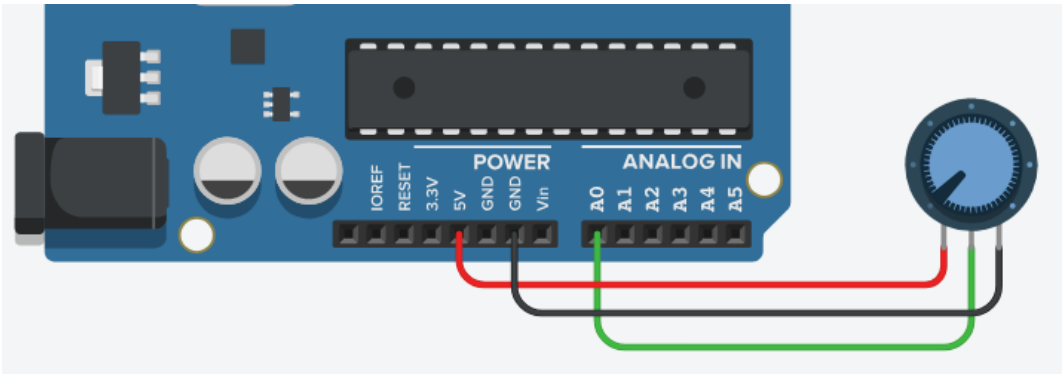


Рисунок 8.7 – Підключення потенціометра до Arduino

Для цього три виходи потенціометра необхідно з'єднати з зазначеними виходами плати:

- Чорний – GND;
- Червоний – живлення 5В;
- Зелений – від центрального виходу до аналогового входу (наприклад, A0).

Змінюючи положення ротора підключеного потенціометра, відбувається зміна параметру опору, яка викликає зміна показника на нульовому піні плати Arduino.

В плату Arduino вбудований аналого-цифровий перетворювач, здатний зчитувати напругу і переводити його в цифрові показники зі значенням від нуля до 1023. При повороті ручки до кінцевого положення в одному з двох можливих напрямків, напруга на виході дорівнює нулю, а, отже, напруга, яке буде генеруватися становить 0 В. При повороті ротора до кінця в протилежному напрямку на пін надходить напруга величиною 5В, а значить числове значення становитиме 1023.

Розглянемо приклад скетчу, за допомогою якого здійснюється регулювання яскравості підключеного до Arduino світлодіоду за допомогою потенціометра:

Лістинг 8.2 – Регулювання яскравості світлодіоду потенціометром

```
#define led 11                  // пін підключення світлодіода
#define pot A0                  // пін підключення потенціометра

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

void loop(){
  int rotate, bright;           // визначаємо змінні типу int
  rotate = analogRead(pot);     // отримуємо значення з потенціометра
  bright = rotat / 4;           // перетворюємо 10-бітне значення у 8-бітне
  analogWrite(led, bright);     // задаємо яскравість світлодіода
```

```
}
```

кінець лістингу 8.2

Директива #define

#define це зручна директива, який дозволяє задати ім'я константі перед тим як програма буде скомпільована. Визначені цією директивою константи не займають програмної пам'яті (на відміну від використання змінних), оскільки компілятор замінює всі звернення до них їх значеннями на етапі компіляції, відповідно вони служать виключно для зручності програміста.

Синтаксис:

#define constantName value

Компілятор замінить будь-яке згадування *constantName* в тексті програми на значення *value*.

Функція analogRead()

Функція зчитує величину напруги з зазначеного аналогового входу. У складі Arduino є 6-канальний (8-канальний – в Mini та Nano, 16 – в Mega) 10-бітний аналогово-цифровий перетворювач, який перетворює вхідну напругу з діапазону від 0 до 5В в цілочисельні значення в межах від 0 до 1023 відповідно.

Синтаксис функції:

analogRead(pin)

де *pin* – номер аналогового входу, з якого буде зчитуватися напруга.

Функція analogWrite()

Функція формує задану аналогову напругу на виході у вигляді ШІМ-сигналу. Може використовуватися для зміни яскравості світлодіода або керування швидкістю обертання двигуна. Після виклику analogWrite(), на виході буде безперервно генеруватися ШІМ-сигнал з заданим коефіцієнтом заповнення до наступного виклику функції analogWrite() (або до моменту виклику digitalRead() або digitalWrite()) на цьому ж виході.

На більшості плат Arduino (на базі мікроконтролерів ATmega168 або ATmega328) функція analogWrite() працює з виходами 3, 5, 6, 9, 10 і 11. На Arduino Mega функція працює з виходами з 2 по 13.

При роботі з analogWrite() попередній виклик функції pinMode() для перемикання пінів в режим «вихід» не потрібний.

Функція analogWrite() не має нічого спільного з аналоговими входами і функцією analogRead().

Синтаксис функції:

analogWrite(pin, value)

pin – вихід, на якому буде формуватися напругу.

value – коефіцієнт заповнення (лежить в межах від 0 до 255).

Хід роботи

1. Побудуйте схему підключення до Arduino сервопривода та потенціометра.

2. Запрограмуйте Arduino таким чином, щоб при повороті ручки потенціометра змінювався кут повороту валу сервопривода (від 0 до 180°), а значення цього кута повороту виводилось на комп’ютер.

3. Оформити звіт по роботі. Звіт повинен містити тему та мету роботи, короткі відомості про функції, що розглядаються в роботі, рисунок побудованої схеми підключення та код програми Arduino.