Міністерство освіти і науки України

Національний університет водного господарства та природокористування

Кафедра автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій

**04-03-374М**

# МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт
з освітнього компоненту
**«Мехатроніка та роботизовані комплекси»**
для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійними програмами
«Автоматизація, комп'ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»,
«Біотехнології, біоробототехніка та біоенергетика» спеціальності 162 «Біотехнології та біоінженерія» денної та заочної форм навчання

Рекомендовано науково-методичною радою з якості ННІЕАВГ
Протокол № 5 від 25 січня 2024 р.
Рекомендовано науково-методичною радою з якості ННІБА
Протокол № 4 від 31 січня 2024 р.

Рівне – 2024

Методичні вказівки до лабораторних робіт з освітнього компонента «Мехатроніка та роботизовані комплекси» для здобувачів вищої освіти першого (бакалаврського) рівня за освітньо-професійними програмами «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка», «Біотехнології, біороботототехніка та біоенергетика» спеціальності 162 «Біотехнології та біоінженерія» денної та заочної форм навчання. [Електронне видання] / Реут Д. Т. – Рівне : НУВГП, 2024. – 75 с.

Укладач: Реут Д. Т., к.т.н., доцент кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій.

Відповідальний за випуск: Древецький В. В., д.т.н., професор, завідувач кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій.

Керівник групи забезпечення спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»: Христюк А. О., к.т.н., доцент, доцент кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій

Керівник групи забезпечення спеціальності 162 «Біотехнологія та біоінженерія»: Грицина О. О., к.т.н., доцент, доцент кафедри теплогазопостачання, вентиляції та санітарної техніки

**Зміст**

## Вступ

Лабораторні роботи з освітнього компоненту «Мехатроніка та роботизовані комплекси» дають змогу на практиці вивчити принципи функціонування мехатронних систем, навчитись використовувати датчики, приводи, засоби індикації мехатронних об’єктів, набути навичок програмування робота-маніпулятора та мобільного робота-гусеничної платформи.

У лабораторних роботах розглянуто:

– введення та виведення сигналів за допомогою анологових та цифрових входів та виходів мікроконтролерної плати Arduino Uno;

– визначення положення об'єкта в просторі цифровим акселерометром-гіроскопом;

– керування двигунами постійного струму та сервоприводами;

– керування роботизованим маніпулятором з 6 сервоприводами в режимі дистанційного й автоматичного керування в ангулярній системі координат;

– керування гусеничним роботом дистанційно через інфрачервоний канал та Bluetooth, програмування його для виконання самостійного руху.

## Лабораторна робота 1. Організація зчитування сигналів з давачів

Мета роботи: навчитися програмувати плату Arduino Uno для зчитування сигналу з давачів.

### Теоретичні відомості

Сучасні мехатронні системи неможливі без мікропроцесорної складової, в кожній наявний один або декілька мікроконтролерів або мікропроцесорів. Наймасовіше використовують саме мікроконтролери, які містять в одній мікросхемі й обчислювальне ядро, й пам'ять, і периферійні модулі вводу-виводу. Вони виконують роль центрального цифрового керуючого елемента, збираючи дані від датчиків, обмінюючись даними з іншими вузлами бортової мережі, формуючи сигнали керування приводами.

Створення компонентів техніки з мікроконтролерами вимагає як розробки апаратної частини (друкованої плати з розпаяним мікроконтролером й іншими компонентами, до якої підключаються датчики, драйвери приводів, інтерфейсні кабелі, засоби індикації тощо), так і програмного забезпечення («прошвики» мікроконтролера). Пришвидшити процес розробки прототипу дозволяє використання налагоджувальних плат, що включають мікроконтролер і потрібні для його роботи компоненти, а також роз’єми для підключення всього іншого.

Arduino – це відкрита електронна платформа для швидкого прототипування (створення прототипів) електроніки, що базується на простому у використанні апаратному та програмному забезпеченні.

Апаратна частина включає мікроконтролерну налагоджувальну плату, а програмна – середовище Arduino IDE. Платформа Arduino є відкритою, тому існує ряд аналогів Arduino-сумісних мікроконтролерних плат різного виконання.

### *Arduino Uno*

**Arduino Uno** – це плата на основі мікроконтролера ATmega328P. Мікроконтролер ATmega328P має автомобільний робочий діапазон температур -40…+125 °C і може використовуватись у рухомій техніці. На платі Arduino Uno наявні 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 внутрішньосхемного програмування (ICSP) і кнопка скидання.

Для початку роботи з платою достатньо просто подати живлення від AC/DC-адаптера або батарейки, або підключити його до комп'ютера за допомогою USB-кабелю.

### *Входи і виходи*

Кожен з 14 цифрових виводів може працювати в якості входу або виходу. Рівень напруги на виводах обмежений 5В. Максимально допустимий (критичний) струм, який може віддавати або споживати один вивід, становить 40 мА, проте рекомендується у тривалому режимі не перевищувати 20-25 мА. Усі виводи з'єднані з внутрішніми підтягуючими резисторами (за замовчуванням відключеними) опором 20-50 кОм. Крім цього, деякі виводи Arduino Uno можуть виконувати додаткові функції (рис. 1.1):

***Послідовний інтерфейс:*** виводи 0 (RX) і 1 (TX) (рис. 1.1). Використовуються для отримання (RX) і передачі (TX) даних по послідовному інтерфейсу. Ці виводи з'єднані з відповідними виводами мікроконтролера ATmega8U2 або ATmega16U2, що виконує роль перетворювача USB/UART.

У платах інших виробників можна зустріти замість нього USB/UART-перетворювачі CH340 або CP2102.

***Зовнішні переривання:*** виводи 2 і 3. Можуть служити джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих виводах. Механізм зовнішніх апаратних переривань дозволяє програмі мікроконтролера позачергово реагувати на зовнішні сигнали, що зменшує час реакції на них. Програма мікроконтролера виконує опитування входів по черзі (виконуючи інструкцію за інстукцією) в головному циклі. Якщо операцій в цьому циклі досить багато – період опитування збільшується, а значить мікроконтролер із затримкою реагуватиме на зовнішні сигнали. При надходженні сигналу переривання виконання головного циклу

зупиняється і відбувається перехід до виконання функції-обробника переривання від цього джерела. Саме в ній можна швидко відреагувати на сигнал, виконавши потрібні дії, й повернутись до виконання головного циклу програми.

***ШІМ***: виводи 3, 5, 6, 9, 10 і 11. За допомогою функції analogWrite() можуть виводити 8-бітові псевдоаналогові значення у вигляді ШІМ-сигналу.

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

***Світлодіод:*** 13. Вбудований світлодіод на платі, приєднаний до виводу 13.

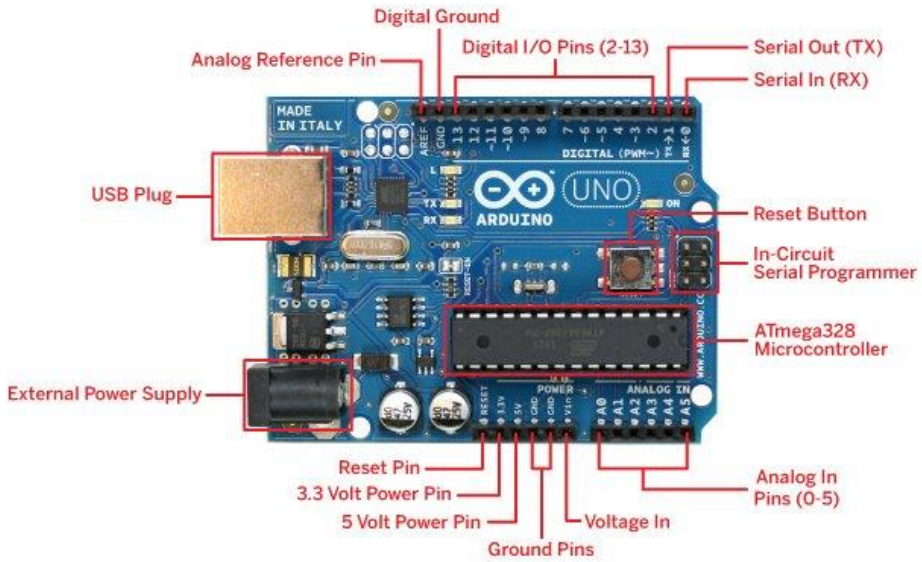



Рис. 1.1. Зовнішній вигляд та призначення виводів плати Arduino Uno

В Arduino Uno є 6 аналогових входів A0… A5 (таблиця 1.1), на кожен з яких можна подати напругу в межах 0 – 5В та отримати аналогово-цифрове перетворення вхідного сигналу у вигляді 10-бітного числа (1024 різних значення).

Верхню межу вхідної напруги можна змінити, використовуючи вивід AREF і функцію analogReference().

Таблиця 1.1

Основні характеристики Arduino Uno

| | |
|---|---|
| Мікроконтролер | ATmega328P |
| Робоча напруга | 5V |
| Вхідна напруга живлення $V_{in}$ (рекомендована) | 7-12V |
| Дискретні (цифрові) входи/виходи | 14 (6 із них можуть використовуватися як ШІМ-виходи) |
| Аналогові входи | 6 |
| Максимальний струм одного вивода | 40 mA |
| Максимальний струм вивода 3,3В | 50 mA |
| Флеш-пам'ять | 32 KB з яких 0.5 KB використовуються завантажувачем |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Тактова частота | 16 MHz |

### *Arduino Mega 2560*

Arduino Mega 2560 – це пристрій на основі мікроконтролера ATmega2560.

Основні характеристики плати Arduino Mega 2560 наведені в табл. 1.2.

Таблиця 1.2

Основні характеристики Arduino Mega 2560

| | |
|---|---|
| Мікроконтролер | ATmega2560 |
| Робоча напруга | 5V |
| Вхідна напруга (рекомендована) | 7-12V |
| Дискретні (цифрові) входи/виходи | 54 (15 із них можуть використовуватися як ШІМ-виходи) |
| Аналогові входи | 16 |
| Максимальний струм одного вивода | 40 mA |
| Максимальний струм вивода 3,3В | 50 mA |
| Флеш-пам'ять | 256 KB з яких 8 KB використовуються |

| | |
|---|---|
| | авантажувачем |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Тактова частота | 16 MHz |

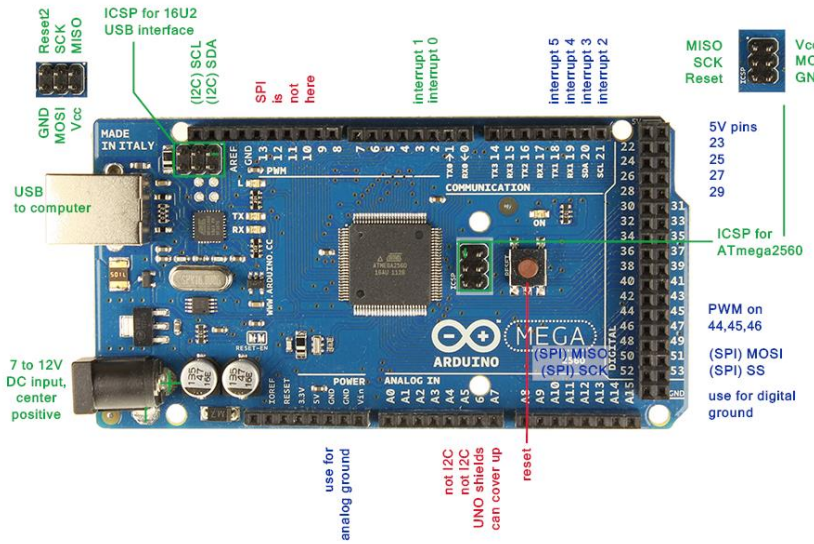



Рис. 1.2. Зовнішній вигляд та призначення виводів плати Arduino Mega 2560

### *Програмування Arduino*

Програмування мікроконтролерних плат Arduino здійснюється у вільно розповсюджуваному середовищі Arduino IDE.

Мікроконтролери в платах Arduino випускаються з прошитим завантажувачем (bootloader), що дозволяє завантажувати в мікроконтролер нові програми без необхідності використання зовнішнього програматора.

Також мікроконтролер можна прошити і через роз'єм для внутрішньосхемного програмування ICSP (In-Circuit Serial Programming), не звертаючи уваги на завантажувач.

Мова програмування Arduino – це видозмінена мова C/C++ із розширеним набором функцій.

Кожна програма для пристроїв Arduino містить дві основні функції:

1) void setup() – функція, що виконується о дин раз, після кожної подачі живлення або скидання плати Arduino;

2) void loop() – виконується постійно в циклі після виконання функції void setup().

Перед завантаженням програми в мікроконтролер слід спочатку

вибрати тип плати Arduino та порт, до якого вона підключена.

### *Деякі функції для роботи з входам/виходами Arduino*

pinMode(pin, value) – налаштування виводу з номером pin як дискретного входу (value=INPUT) або дискретного виходу (value=OUTPUT). Наприклад:

pinMode(13, OUTPUT);

*digitalRead(pin)* – функція зчитує із заданого входу значення HIGH або LOW.

*digitalWrite(pin, value)* – подає на цифровий вхід/вихід значення HIGH або LOW;

*analogRead(pin)*– зчитує величину напруги з аналогового входу і видає результат у вигляді цілого числа від 0 до 1023.

### *Модулі мікроконтролера, задіювані для зчитування сигналів*

Мікроконтролер ATmega328P в Arduino Uno містить 3 таймери. Вони можуть використовуватись для отримання імпульсного сигналу від давачів. Мікроконтролер ATmega328P містить два 8-розрядні таймери та один 16-розрядний. Таймер 1 має регістр захоплення OCR1, куди записується вміст таймера TCNT1 по події захоплення - зміні сигналу на вході ICP1 або виході аналогового компаратора.

**Режим захоплення (capture).** У режимі захоплення 16-ти бітне значення таймера (TCNT1) захоплюється в регістр OCR1 при кожній події на вході ICP1 або виході аналогового компаратора. Подія для захоплення задається в регістрі TCCR1B (біт ICES1) та ACSR (біт ACIC).

Режим захоплення використовується для вимірювання тривалості між двома подіями, наприклад періоду, тривалості імпульсу, тривалості паузи, скважності (коефіцієнта заповнення) тощо.

*Приклад 1.* Вимірювання періоду дискретного сигналу (рис. 2.1)

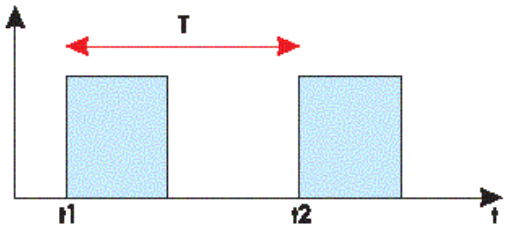



Рис. 2.1. Вимірювання періоду

*Приклад 2.* Вимірювання періоду з усередненням результату за 16 імпульсів (рис. 2.2)

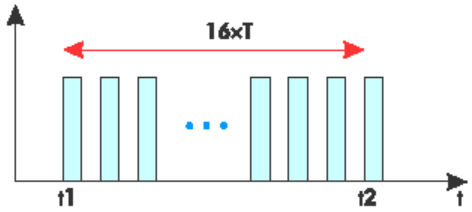


Рис. 2.2. Вимірювання періоду з усередненням

*Приклад 3.* Вимірювання тривалості імпульсу (рис. 2.3)

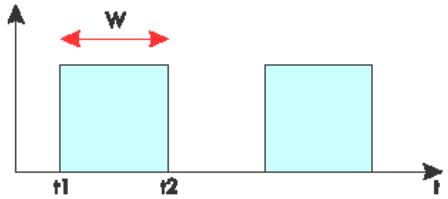


Рис. 2.3. Вимірювання тривалості імпульсу

*Приклад 4.* Вимірювання скважності імпульсів (рис. 2.4)

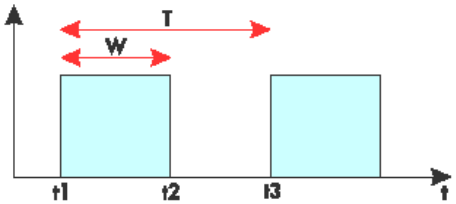


Рис. 2.4. Вимірювання скважності імпульсів

**Аналогово-цифровий перетворювач** (АЦП, англ. Analog-todigital conventer, ADC) - пристрій, що перетворює вхідний аналоговий сигнал в цифровий код (цифровий сигнал). За його допомогою можна зчитувати сигнали з датчиків із аналоговим виходом.

У мікроконтролері ATmega328P є 10-розрядний АЦП. Вхідна напруга конвертується в 10-розрядне двійкове значення. Мінімальне значення відповідає 0, а максимальне - опорній напрузі $V_{ref}$. Отриманий результат перетворення записується в 2 регістра: ADCH і ADCL та повертається функцією analogRead(pin) як число в діапазоні 0…1023. Результат перетворення можна розрахувати за формулою

$$ADC = \frac{V_{in} \cdot 1023}{V_{ref}}$$

В ATmega328P АЦП реалізовано на основі ЦАП та компаратора. На вхід ЦАП подається зростаючий цифровий код, на виході отирмується зростаюча напруга. Коли напруга з виходу ЦАП зрівняється з вимірюваною напругою з обраного каналу АЦП, процес перетворення можна вважати закінченим та код, що в цей момент був на вході ЦАП, є результатом аналогово-цифрового перетворення. Внаслідок перехідних процесів швидкість АЦП є обмеженою. Щоб

характеристики точності АЦП залишались в межах, заявлених виробником, частота опитування не повинна перевищувати 15000 вибірок за секунду.

**План роботи**

1. Навчитися підключати давачі з дискретними вихідними сигналами до Arduino та обробляти дані цих давачів.
2. Навчитися підключати давачі з аналоговими вихідними сигналами до Arduino та обробляти дані цих давачів.

**Порядок виконання роботи**

1. Підключити до плати Arduino ультразвуковий давач вимірювання відстані HC-SR04 (рис. 2.5).


Рис. 2.5. Зовнішній вигляд сенсора HC-SR04

Вивід Echo давача підключити до виводу 12 плати Arduino; вивід Trigger – до виводу 11; вивід Vcc – до виводу 5 В; вивід GND – до одного з виводів GND на платі Arduino.

Завантажити в мікроконтролер наступну програму:

```
#define echoPin 12 // Echo Pin
#define trigPin 11 // Trigger Pin
#define LEDPin 13 // LED
char rec;

int maximumRange = 350; // максимальна відствнь
int minimumRange = 0; // Minimum range needed
float duration, distance; // тривалість, яка використовується для розрахунку відстані

void setup() {
 Serial.begin (9600);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(LEDPin, OUTPUT); // світлодіод, що сигналізує помилку
}

void loop() {
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);  //період тиші
```

```
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);  //випромінюємо ультразвук

 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH); //вимірюємо час до отримання відбитої хвилі

distance = duration/58.2; //розраховуємо відстань

 if (distance >= maximumRange || distance <= minimumRange){
 //у випадку виходу за межі вимірювання сигналізуємо помилку
 Serial.println("error");
 Serial.print("\n\r");
 digitalWrite(LEDPin, HIGH);
 delay(50);
 }
 else { //у протилежному випадку надсилаємо значення відстані
 Serial.println(distance);
 Serial.print("\n\r");
 digitalWrite(LEDPin, LOW);
 delay(250);
 }
}
```

Відкрити монітор послідовного порту. Піднести руку навпроти давача на відстані 20 – 50 см. Спостерігати за виміряним значенням відстані, що виводиться на екран. Відкрити послідовний плоттер та побудувати графік зміни відстані.

2. Підключити підстроювальний або змінний резистор за схемою подільника напруги (потенціометра) згідно рис. 2.6. Вхід In підключити до контакту 5V на платі Arduino Uno, вихід Out – A0.

Відповідно до напруги на ковзному контакті (0...5 В) передавати у послідовний інтерфейс значення змінної norm, яка зберігає число від 0 до 100, яке прямопропорційно залежить від напруги.

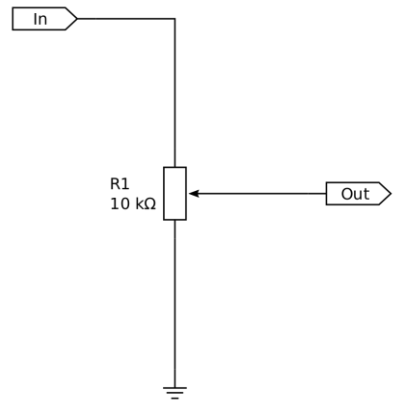



Рис. 2.6. Схема підключення змінного резистора

```
int sensorPin = A0;    // аналоговий вхід
int sensorValue = 0;  // значення сигналу від давача
int norm=0; // нормоване значення сигналу з давача
void setup() {
Serial.begin(9600);  // налаштовуємо послідовний порт для зв'язку з ПК
}

void loop() {
  sensorValue = analogRead(sensorPin);      // зчитуємо значення з давача
   norm=map(sensorValue, 0, 1023, 0, 100); // нормуємо сигнал в межах від 0 до 100
   Serial.println(norm);  // виводимо нормоване значення сигналу в порт
    delay(500);  // затримка 0,5 с
 }
```

3. Замість змінного резистора підключити до аналогового входу подільник, утворений постійним резистором й давачем освітленості (фоторезистором). Змінюючи освітленість фоторезистора, спостерігати за значенням змінної norm.

4. Оформити звіт про виконання роботи. Звіт повинен містити: назву та мету роботи; тексти програм з коментарями; висновок про виконання роботи.

**Контрольні запитання**

1. Скільки таймерів має мікроконтролер ATmega328P?
2. Яка розрядність модуля АЦП мікроконтролера ATmega328P?
3. Яка кількість каналів модуля АЦП в Arduino Uno?
4. Якою функцією можна виміряти тривалість імпульсу?
5. Якою функцією можна отримати результат АЦП?

## Лабораторна робота 2. Використання акселерометра-гіроскопа

Мета роботи: навчитися використовувати мікросхеми акселерометрів-гіроскопів для визначення положення та параметрів руху об’єктів

### Теоретичні відомості

Набільшого поширення сьогодні набули акселерометри та гіроскопи, виконані за MEMS технологією.

MEMS-датчики широко застосовуються в робототехніці, смартфонах, автомобільній промисловості для управління подушками безпеки, в охоронній сигналізації, в навігаційних системах для обчислення пройденого шляху або визначення маршруту проходження.

Найчастіше для побудови таких сенсорів використовують вимірювання зміщення інерційної маси всередині під дією зовнішніх прискорень. Вимірюваною величиною є напруга внаслідок п’єзоелектричного ефекту або зміна ємності конденсаторів.

### *Модуль акселерометра-гіроскопа GY-521*

Модуль GY-521 (рис. 2.1) на мікросхемі MPU6050 - це 3-осьовий гіроскоп і акселерометр на три координати. На платі модуля GY-521 розміщені всі необхідні для її надійного функціонування елементи, в тому числі і підтягуючі резистори. Обмін даними з контролером здійснюється по шині $I^2C$.


Рис. 2.1. Зовнішній вигляд модуля акселерометра-гіроскопа GY-521

Необхідну напругу живлення для модуля MPU6050 регулює вбудований стабілізатор напруги 3,3V.

Серцем модуля GY-521 є мікросхема MPU-6050.

Характеристики його наступні:

- Напруга живлення: 3-5 В
- Зв'язок: $I^2C$ протокол
- Діапазон вимірювань гіроскопа: ± 250, 500, 1000, 2000 °/c
- Діапазон вимірювань акселерометра: ± 2, 4, 8, 16 g
- Крок між контактами: 2.54 мм
- Вбудований 16-бітний АЦП
- Розмір плати: 20 мм x 16 мм

### План роботи

1. Ознайомитися з принципом дії MEMS-акселерометрів.

2. Скласти програму для визначення прискорення об’єкта, на якому закріплений модуль з мікросхемою MPU-6050.

3. Скласти програму для визначення положення об’єкта, на якому закріплений модуль з мікросхемою MPU-6050. Перевірити дрейф нуля під час роботи.

**Порядок виконання роботи**

1. Ознайомитися з теоретичними відомостями.

2. Завантажити середовище Arduino IDE та підключити бібліотеку для роботи з MPU-6050 https://github.com/jarzebski/Arduino-MPU6050 .

3. У середовищі Adruino IDE створити нову програму.

```
#include <Wire.h>
#include <MPU6050.h>

MPU6050 mpu;

void setup()
{
  Serial.begin(115200);

  Serial.println("Initialize MPU6050");

  while(!mpu.begin(MPU6050_SCALE_2000DPS,
MPU6050_RANGE_2G))
  {
    Serial.println("Could not find a valid MPU6050 sensor, check
wiring!");
    delay(500);
  }
  checkSettings();
}

void checkSettings()
{
  Serial.println();

  Serial.print(" * Sleep Mode:            ");
  Serial.println(mpu.getSleepEnabled() ? "Enabled" : "Disabled");

  Serial.print(" * Clock Source:          ");
  switch(mpu.getClockSource())
  {
    case MPU6050_CLOCK_KEEP_RESET:     Serial.println("Stops
```

```
the clock and keeps the timing generator in reset"); break;
            case MPU6050_CLOCK_EXTERNAL_19MHZ: Serial.println("PLL
with external 19.2MHz reference"); break;
            case MPU6050_CLOCK_EXTERNAL_32KHZ: Serial.println("PLL
with external 32.768kHz reference"); break;
            case MPU6050_CLOCK_PLL_ZGYRO:      Serial.println("PLL with
Z axis gyroscope reference"); break;
            case MPU6050_CLOCK_PLL_YGYRO:      Serial.println("PLL with
Y axis gyroscope reference"); break;
            case MPU6050_CLOCK_PLL_XGYRO:      Serial.println("PLL with
X axis gyroscope reference"); break;
            case MPU6050_CLOCK_INTERNAL_8MHZ:
Serial.println("Internal 8MHz oscillator"); break;
          }

          Serial.print(" * Accelerometer:         ");
          switch(mpu.getRange())
          {
            case MPU6050_RANGE_16G:           Serial.println("+/- 16 g");
break;
            case MPU6050_RANGE_8G:            Serial.println("+/- 8 g");
break;
            case MPU6050_RANGE_4G:            Serial.println("+/- 4 g");
break;
            case MPU6050_RANGE_2G:            Serial.println("+/- 2 g");
break;
          }

          Serial.print(" * Accelerometer offsets: ");
          Serial.print(mpu.getAccelOffsetX());
          Serial.print(" / ");
          Serial.print(mpu.getAccelOffsetY());
          Serial.print(" / ");
          Serial.println(mpu.getAccelOffsetZ());

          Serial.println();
        }

        void loop()
        {
          Vector rawAccel = mpu.readRawAccel();
          Vector normAccel = mpu.readNormalizeAccel();

          Serial.print(" Xraw = ");
```

```
  Serial.print(rawAccel.XAxis);
  Serial.print(" Yraw = ");
  Serial.print(rawAccel.YAxis);
  Serial.print(" Zraw = ");

  Serial.println(rawAccel.ZAxis);
  Serial.print(" Xnorm = ");
  Serial.print(normAccel.XAxis);
  Serial.print(" Ynorm = ");
  Serial.print(normAccel.YAxis);
  Serial.print(" Znorm = ");
  Serial.println(normAccel.ZAxis);

  delay(10);
}
```

4. Скомпілювати програму та переконатися у відсутності помилок.

5. Підключити мікроконтролерну плату до USB-порта комп'ютера.

6. У середовищі Arduino IDE вибрати тип плати та порт, до якого вона підключена. Завантажити створену програму в мікроконтролер.

7. Підключити MPU-6050 до Arduino Uno згідно рис. 2.2.

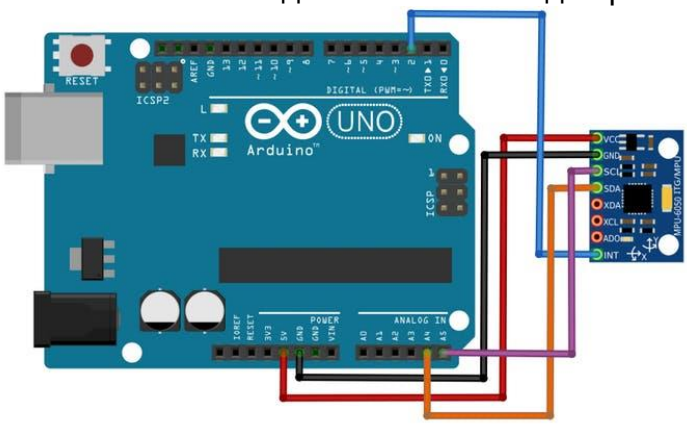



Рис. 2.2. Підключення модуля акселерометра до Arduino Uno

Підключити складену схему до USB-порту й відкрити монітор послідовного порту. Переконатися в коректності роботи програми.

8. Використовуючи приклади до бібліотеки MPU-6050, створити програму, що виводитиме в послідовний порт поточне положення модуля в просторі. Перевірити її роботу.

9. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. Який принцип дії ємнісного акселерометра-гіроскопа?

2. Який заголовний файл необхідно підключити для роботи з MPU6050?

3. Для чого потрібна нормалізація зчитаних з акселерометра значень?

4. Який інтерфейс використовується для підключення MPU6050 до Arduino Uno?

## Лабораторна робота №3. Керування колекторними двигунами постійного струму

Мета роботи: ознайомитися зі способами керування швидкістю обертів двигунів постійного струму. Навчитися створювати програми керування швидкістю обертів ДПС для мікроконтролера.

### Теоретичні відомості

У мехатронних системах одним з найпоширеніших виконавчих пристроїв є двигун постійного струму (ДПС). Конструкції ДПС відрізняються великою різноманітністю. У цій лабораторній роботі розглядається колекторний ДПС, будову якого схематично показано на рис. 3.1.

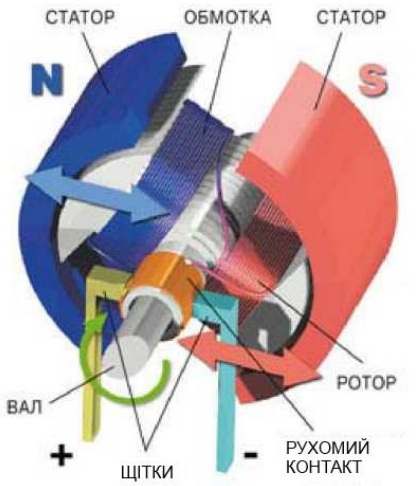



Рис. 3.1. Будова колекторного ДПС з двополюсним статором і двополюсним ротором

Найпростіший колекторний ДПС з двополюсним статором і з двополюсним ротором, складається з одного постійного магніту на статорі, з одного електромагніту з явно вираженими полюсами на роторі (якорі), щітково-колекторного вузла з двома пластинами (ламелями) і двома щітками.

#### *Широтно-імпульсна модуляція*

Регулювати швидкість обертів ДПС при зміні навантаження на валу можна зміною напруги, що подається на коло якоря. У сучасних мікропроцесорних САР напругу змінюють не безперервно, а дискретно за допомогою широтно-імпульсної модуляції (ШІМ). Принцип ШІМ

пояснюється на рис. 3.2. При ШІМ вихідний сигнал являє собою імпульсний сигнал постійної частоти і змінної скважності. Скважність – це відношення періоду проходження імпульсів T до їх тривалості $\tau$. Обернена величина – коефіцієнт заповнення:

$$s = \frac{\tau}{T}.$$

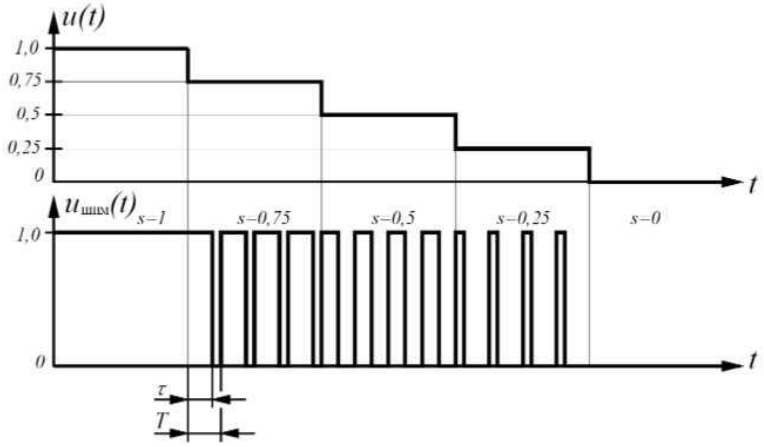



Рис. 3.2. Широтно-імпульсна модуляція

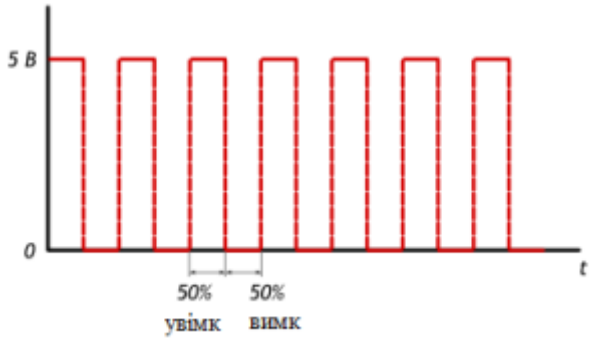
Широтно-імпульсна модуляція (ШІМ - англ. *pulse-width modulation, PWM*), або модуляція за тривалістю імпульсів (англ.*pulse-duration modulation, PDM*) – процес керування шириною (тривалістю) високочастотних імпульсів по закону, який задає низькочастотний сигнал. В електроніці це може бути керування середнім значенням вихідної напруги шляхом зміни тривалості замкнутого стану електронного (електромеханічного) ключа.

**Аналогова ШІМ.** ШІМ-сигнал генерується аналоговим компаратором, на один вхід якого подається опорний сигнал значно більшої частоти, ніж модулюючий у вигляді «трикутника» або «пилки», а на іншій – модулюючий неперервний аналоговий сигнал. Частота вихідних імпульсів ШІМ відповідає частоті «зубів» пилки. В ту частину періоду, коли напруга сигналу на позитивному вході вище сигналу на негативному вході, на виході буде одиниця, в іншу, коли сигнал на позитивному вході нижче сигналу на негативному вході - нуль.

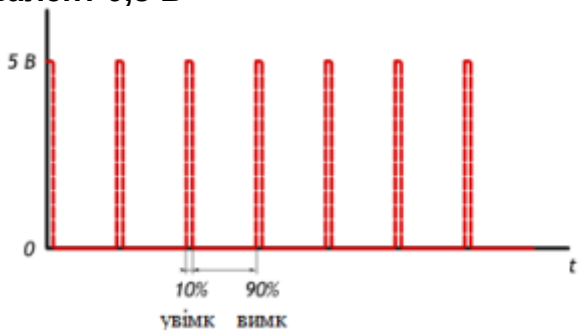
**Цифрова ШІМ.** У двійковій цифровій техніці, виходи в якій можуть мати тільки одне з двох значень, часто використовується наближення бажаного середнього рівня вихідного сигналу за допомогою ШІМ. Пилкоподібний сигнал генерується *N*-бітним лічильником.

Розглянемо кілька прикладів при напрузі живлення $V_{cc}$=5 В.

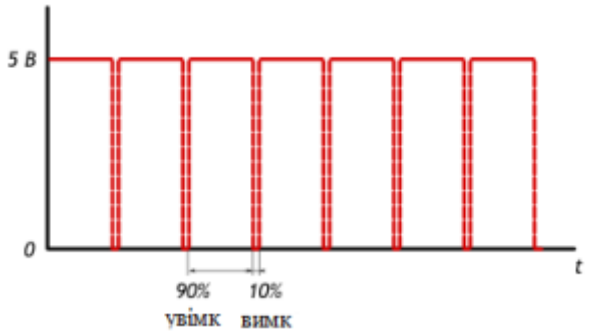
**50% — еквівалент 2,5 В**




**10% — еквівалент 0,5 В**




**90% — еквівалент 4,5 В**




Для подачі ШІМ сигналу певної скважності на дискретний вихід Arduino використовується функція analogWrite().

analogWrite() – формує задану "аналогову" напругу на виводі у вигляді ШІМ-сигналу. Після виклику analogWrite(), на виводі буде безперервно генеруватися ШІМ-сигнал із заданим коефіцієнтом заповнення до наступного виклику функції analogWrite() (або до моменту виклику digitalRead() або digitalWrite(), що взаємодіють з цим же виводом). Частота ШІМ становить приблизно **490 Гц**. На платі **Uno** та подібних виводи 5 і 6 мають частоту ШІМ приблизно **980 Гц**.

На більшості плат Arduino (на базі мікроконтролерів ATmega168 або ATmega328) функція analogWrite() працює з виводами 3, 5, 6, 9, 10 і 11. На **Arduino Mega** функція працює з виводами з 2 по 13 і 44 – 46.

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

Синтаксис:

analogWrite(pin, value)

Параметри:

pin - номер ШІМ-виходу

value - значення скважності від 0 (завжди "0") до 255 (завжди "1").

Найпростіша схема керування двигуном постійного струму складається з польового транзистора, на затвор якого подається ШІМ сигнал. Транзистор в даній схемі виконує роль електронного ключа, який комутує один з виводів двигуна на спільний провідник («мінус» живлення). Транзистор відкривається на час тривалості імпульсу.

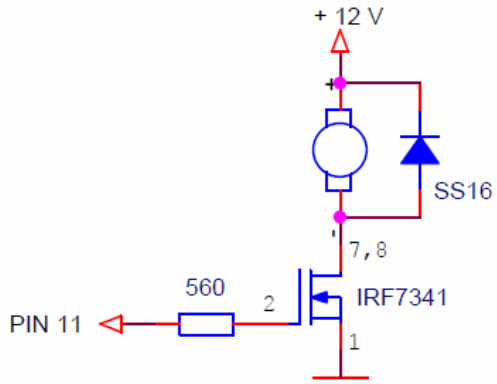



Рис. 3.3. Схема керування швидкістю обертів ДПС з використанням польового транзистора

**План роботи**

1. Ознайомитися зі способами керування швидкістю обертів ДПС.
2. Скласти програму для керування швидкістю обертів ДПС.

**Порядок виконання роботи**

1. Ознайомитися з теоретичними відомостями.
2. Завантажити середовище Arduino IDE.
3. Скласти на макетній платі наступну схему рис. 3.3.
4. В середовищі Adruino IDE створити нову програму.

```
int led = 11; // ШІМ-вивід, до якого підключений затвор транзистора
int speed = 0; // ця змінна відповідає за швидкість обертання
int step = 5; // Крок зміни швидкості
void setup ()
{  // Налаштовуємо вивід 9 як дискретний вихід
pinMode (led, OUTPUT);
}
```

```
void loop ()
{  // Встановлюємо швидкість обертання двигуна
analogWrite(led, speed); // Збільшуємо поточне значення швидкості обертання
speed = speed + step;  // Коли швидкість стає максимальною / мінімальною - починаємо її знижувати / підвищувати
if (speed == 0 || speed == 255)
{
step = -step;
}
// Пауза 30 мілісекунд
 delay (30);
}
```

5. Скомпілювати програму та переконатися у відсутності помилок.

6. Підключити мікроконтролерну плату до USB-порта комп'ютера.

7. У середовищі Arduino IDE вибрати тип плати та порт, до якого вона підключена. Завантажити створену програму в мікроконтролер. Переконатися в коректності роботи програми.

8. Самостійно створити програму для задання постійної швидкості обертання ДПС на рівні 95% від верхньої межі діапазону регулювання. Завантажте програму в МК та переконайтеся в правильності її роботи.

9. Оформити звіт про виконання лабораторної роботи. Звіт повинен містити: назву та мету лабораторної роботи; тексти програм з коментарями; висновок про виконання роботи.

**Контрольні запитання**

1. Яка будова найпростішого колекторного ДПС?
2. Які способи керування швидкістю ДПС Вам відомі?
3. Що таке широтно-імпульсна модуляція?
4. Що таке скважність імпульсів?
5. Як можна змінювати швидкість обертання ДПС за допомогою ШІМ?

## Лабораторна робота №4. Керування кроковими двигунами

Мета роботи: навчитись розробляти програми керування кроковими двигунами.

### Теоретичні відомості

Принцип дії всіх крокових двигунів ґрунтується на дискретній зміні стану магнітного поля в повітряному проміжку статора. Ротор позиціонується відповідно до конфігурації магнітного поля, створеного струмами, що протікають через обмотки полюсів статора (рис. 4.1).

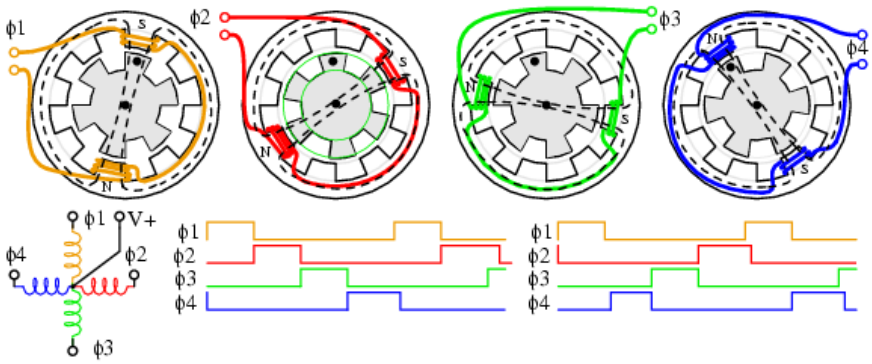



Рис. 4.1. Принцип дії крокового двигуна

Залежно від внутрішнього з'єднання обмоток у двигуні розрізняють уніполярні та біполярні крокові двигуни (рис. 4.2).

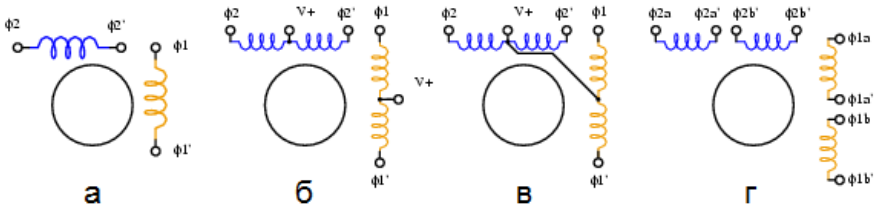



Рис. 4.2. Крокові двигуни: а – біполярний, б – уніполярний 6-провідний, в – уніполярний 5-провідний, г – 8-провідний/4-обмотковий (можна підключити і як уніполярний, і як біполярний)

Крововий двигун не обертатиметься, якщо на нього просто подати напругу живлення. Для приведення в рух потрібно перемикати струми в обмотках двигуна так, щоб двигун виконував «кроки» в потрібному напрямку (рис. 4.3). Програмно це зводиться до вибірки стану ключів (транзисторів) 1/0 з двовимірного масиву зі стовпця, який відповідає поточному номеру кроку. Якщо двигун потрібно обертати в одному напрямку – номер кроку інкрементуємо і масив проходимо зліва направо, якщо в протилежному – номер кроку декрементуємо та рухаємось справа наліво по масиву. Частота переходу між стовпцями масиву задаватиме частоту обертання ротора двигуна. Якщо не здійснювати перехід до наступного стовпця (не змінювати стан

транзисторних ключів), то двигун зупиниться.

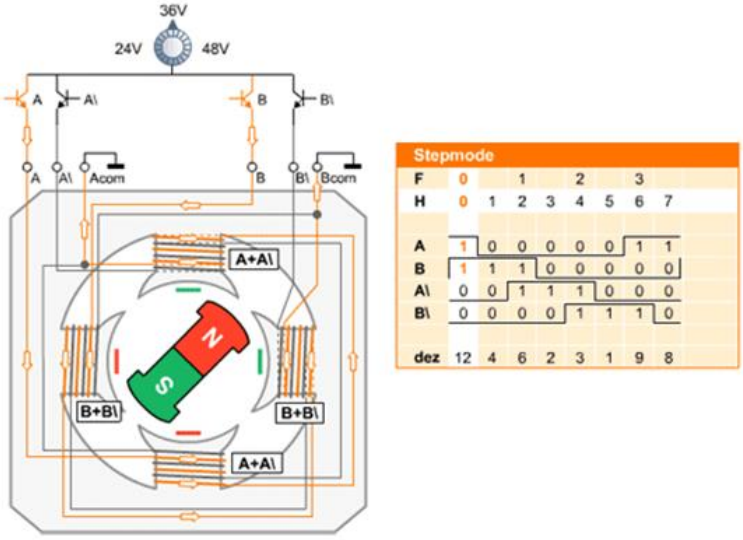



Рис. 4.3. Принцип керування уніполярним кроковим двигуном

В Arduino для керування кроковим двигуном можна використати бібліотеку Stepper. Вона надає клас Stepper. При створенні об'єкта класу Stepper у конструкторі потрібно вказати кількість кроків на оберт і номери контактів для керування двигуном. Об'єкт класу Stepper має методи *void step(int number_of_steps)* для виконання заданої кількості кроків (від'ємний аргумент відповідає за рух у протилежному напрямку) та *void setSpeed(long whatSpeed)* для задання швидкості руху.

**План роботи**

1.Скласти програму для керування положенням сервопривода.

2. Реалізувати циклограму положення вала сервопривода.

3. Програмно обмежити швидкість повороту вала сервопривода.

**Порядок виконання роботи**

1. Скласти схему для керування кроковим двигуном 28BYJ-48 з використанням збірки транзисторів Дарлінгтона ULN2003A згідно рис. 4.4.

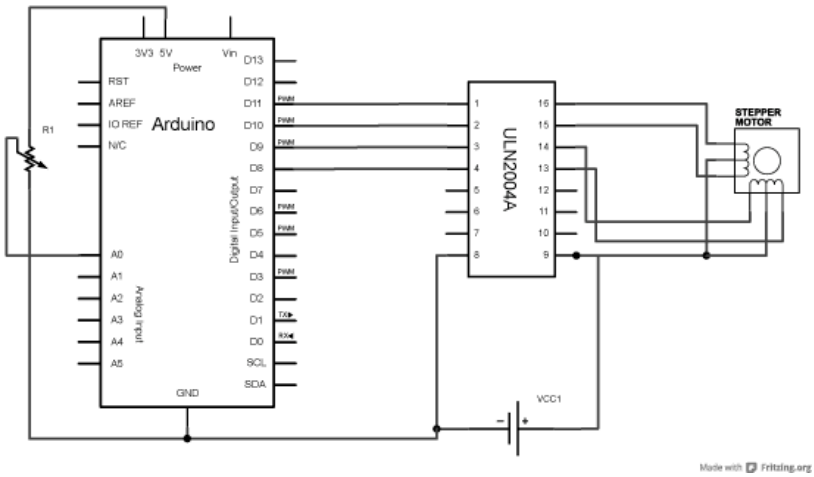



Рис. 4.4. Схема підключення уніполярного крокового двигуна до Arduino

2. Використовуючи бібліотеку Arduino Stepper і приклад програми із неї StepperOneStepAtATime, розробити програму, яка обертатиме вихідний вал двигуна зі швидкістю 1 оберт за хвилину.

3. Записати в Arduino Uno програму, що змінює положення валу крокового двигуна відповідно до положення потенціометра, підключеного до входу A0. Перевірити її роботу.

```
#include <Stepper.h>
#define STEPS 64
Stepper stepper(STEPS, 8, 9, 10, 11);

void setup() {
  stepper.setSpeed(30);
}
void loop() {
  int val = analogRead(0);
  stepper.step(val - previous);
  previous = val;
}
```

4. Модифікувати попередню програму таким чином, щоб середнє положення ковзного контакту потенціометра відповідало зупиненому двигуну, відхилення ліворуч/праворуч від середнього положення спричиняло обертання двигуна у відповідному напрямку, причому чим більше відхилення від середнього положення (*abs(512-analogRead(A0))*), тим більшу швидкість повинен розвивати кроковий двигун. Прошити плату й перевірити правильність роботи програми.

5. Оформити звіт про виконання роботи. Звіт повинен містити: назву та мету роботи; тексти програм з коментарями; висновок про виконання роботи.

### Контрольні запитання

1. Що таке кроковий двигун?
2. Які розрізняють крокові двигуни за схемою підключення обмоток?
3. Що буде, якщо кроковий двигун підключити напряму до джерела живлення?
4. Яка бібліотека використовується у роботі для керування кроковим двигуном?
5. Який метод з класу Stepper дозволяє змінити швидкість обертання?
6. Який метод з класу Stepper дозволяє виконати переміщення на задану кількість кроків?

## Лабораторна робота №5. Дослідження роботи сервоприводів та реалізація циклограми

Мета роботи: ознайомитися зі способами керування положенням сервоприводів. Навчитися створювати програми керування швидкістю повороту і положенням вихідного вала сервопривода.

### Теоретичні відомості

Під сервоприводом (рис. 5.1) найчастіше розуміють механізм з електродвигуном, якому можна скомандувати повернутися в заданий кут і утримувати це положення. Однак, якщо сказати повніше, сервопривод - це привод з управлінням через негативний зворотний зв'язок, що дозволяє точно керувати параметрами руху. Сервоприводом є будь-який тип механічного приводу, що має в складі давач (положення, швидкості, зусилля тощо) і блок керування приводом, який автоматично підтримує необхідні параметри згідно заданого завдання. Тобто сервопривод отримує на вхід значення керуючого параметра, наприклад, заданий кут повороту. Блок управління порівнює це значення зі значенням зі свого датчика. На основі результату порівняння привод виконує певнку дію, наприклад: поворот, прискорення або сповільнення так, щоб значення з внутрішнього давача стало якомога ближче до значення зовнішнього керуючого параметра.

Рис. 5.1. Сервоприводи

Найбільш поширені сервоприводи, які утримують заданий кут, і сервоприводи, що підтримують задану швидкість обертання.

Сервопривод – це регульований редукторний електродвигун. Він зазвичай складається з приводного механізму з двигуном постійного струму, плати управління і потенціометра, котрий забезпечує зворотний зв'язок (рис. 5.2).

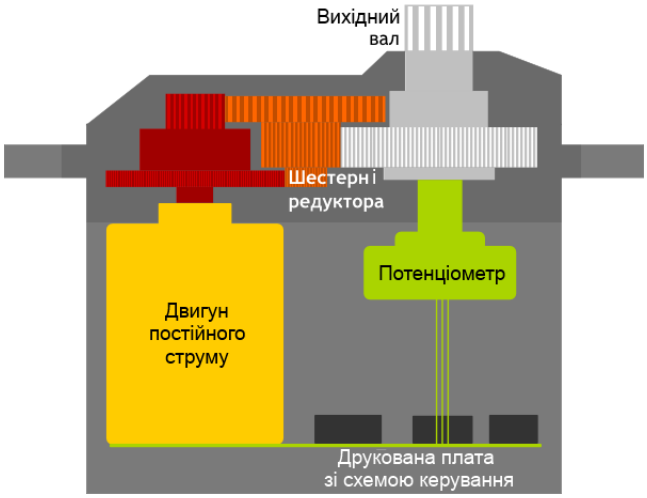



Рис. 5.2. Будова сервопривода

***Управління сервоприводом. Інтерфейс керуючих сигналів***

Керуючий сигнал для сервопривода – імпульси постійної частоти і змінної ширини. Те, яке положення повинен зайняти сервопривод, залежить від довжини імпульсів. В малих сервоприводах, які часто використовуються в RC-моделях, імпульси повинні надходити з частотою 50 Гц. Це означає, що імпульс випускається і приймається раз в 20 мс. Зазвичай при цьому тривалість імпульсу 1520 мкс означає, що сервопривод повинен зайняти середнє положення. Збільшення або зменшення довжини імпульсу змусить сервопривод повернутися за годинниковою або проти годинникової стрілки відповідно. При цьому існують верхня і нижня межі тривалості імпульсу (рис. 5.3).

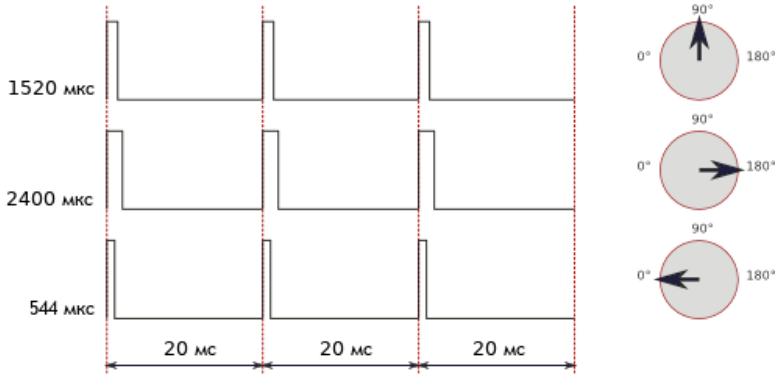


Рис. 5.3. Залежність кута повороту сервопривода від трвалості керуючих імпульсів

У бібліотеці Servo для Arduino за замовчуванням виставлені наступні значення довжин імпульсу: 544 мкс – для 0° і 2400 мкс – для 180°. На конкретному пристрої заводські налаштування можуть відрізнятися від вказаних. Навіть у рамках однієї і тієї ж моделі сервоприводу може існувати похибка, що допускається при виробництві, яка призводить до того, що робочий діапазон довжин імпульсів трохи відрізняється. Для точної роботи кожен конкретний сервопривод повинен бути відкалібрований: шляхом експериментів необхідно підібрати коректний діапазон, характерний саме для нього. При формуванні сигналу керування для сервопривода важливою є довжина імпульсів, а не частота їх появи. 50 Гц – це норма, але сервопривод буде працювати коректно і при 40, і при 60 Гц. При цьому слід мати на увазі, що при сильному зменшенні частоти він може працювати ривками, а при сильному завищенні частоти може перегрітися і вийти з ладу.

Сервоприводи малої потужності (наприклад, SG90) можна безпосередньо підключати до плати Arduino (рис. 5.4), для підключення більш потужних – слід використовувати зовнішнє джерело живлення.

Бібліотека Servo дозволяє здійснювати програмне керування сервоприводами. Управління здійснюється наступними методами:

attach() – приєднує об’єкт Servo до конкретного виводу плати. Можливі два варіанти синтаксису для цього методу: servo.attach(pin) і servo.attach (pin, min, max). При цьому pin - номер виводу, до якого приєднують сервопривод, min і max - довжини імпульсів в мікросекундах, що відповідають за кути повороту 0° і 180°. За замовчуванням виставляються рівними 544 мкс і 2400 мкс відповідно.

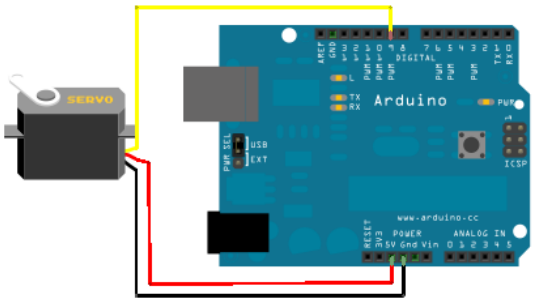



Рис. 5.4. Підключення сервопривода малої потужності до мікроконтролерної плати Arduino Uno

write() - віддає команду сервоприводу прийняти деяке значення параметра. Синтаксис наступний:

servo.write (angle), де angle - кут, на який повинен повернутися сервопривод.

writeMicroseconds() - віддає команду надіслати на сервопріводімульс певної довжини, є низькорівневим аналогом попередньої команди. Синтаксис наступний:

servo.writeMicroseconds(uS), де uS – довжина імпульсу в мікросекундах.

read() - читає поточне значення кута, в якому знаходиться сервопривод. Синтаксис наступний: servo.read(), повертається ціле значення від 0 до 180.

attached() - перевірка, чи було приєднано об’єкт Servo до конкретного виводу. Синтаксис наступний: servo.attached(), повертається true, якщо приєднано до якого-небудь виводу, або false в зворотному випадку.

detach() - виконує дію, зворотну дії attach(), тобто від'єднує об’єкт Servo від виводу, до якого він була прив’язаний. Синтаксис наступний: servo.detach().

***Приклад програми з використанням бібліотеки Servo***

```
#include <Servo.h>
Servo armservo;
void setup()
{
  armservo.attach(9);
}
void loop()
{
  armservo.write(90);// встановлюємо сервопривод в середнє положення
  delay(500);
```

```
  armservo.write(0); // встановлюємо сервопривод в крайнє лвве положення
   delay(500);
   armservo.write(180);// встановлюємо сервопривод в крайнє праве положення
   delay(500);
}
```

**План роботи**

1.Скласти програму для керування положенням сервопривода.

2. Реалізувати циклограму положення вала сервопривода.

3. Програмно обмежити швидкість повороту вала сервопривода.

**Порядок виконання роботи**

1. Складіть схему за рис. 5.4, використовуючи сервопривод SG90.

2. Завантажте програму з прикладу бібліотеки Servo.

Напишіть програму для реалізації наступної циклограми (рис. 5.5).

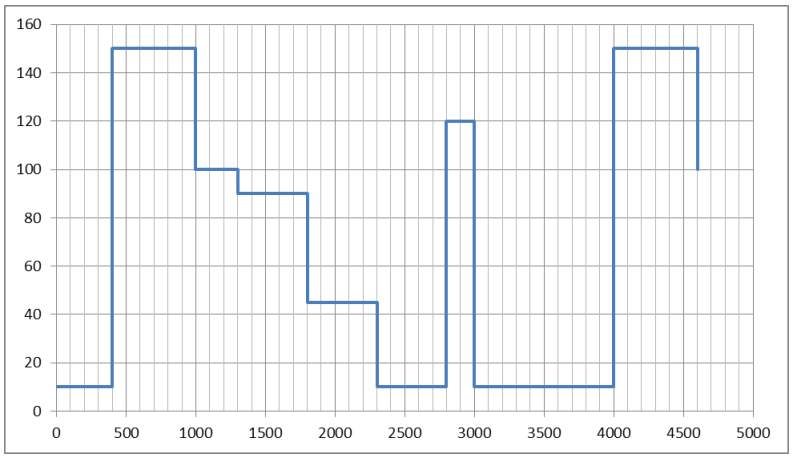



Рис. 5.5. Задана циклограма роботи сервопривода

Вісь абсцис – час в мс, вісь ординат – положеня вала сервопривода в градусах.

3. Для того, щоб мати змогу змінювати швидкість обертання використайте бібліотеку VarSpeedServo (встановіть через менеджер бібліотек). Завантажити в мікроконтролер приклад Sweep з VarSpeedServo, змінюючи значення швидкості. Переконатися в правильності роботи програми та точності позиціювання сервопривода.

4. Оформити звіт про виконання роботи. Звіт повинен містити: назву та мету роботи; тексти програм з коментарями; висновок про виконання роботи.

**Контрольні запитання**

1. Що таке сервопривод?

2. Назвіть основні елементи сервопривода.

3. Яка роль датчиків у конструкції сервопривода?

4. Який вигляд має задаючий сигнал для сервоприводів, використаних у практичній роботі?

5. Чим відрізняється сервопривод від крокового двигуна?

6. Як реалізоване керування положенням вала сервопривода, що використано в цій роботі?

7. Яка бібліотека дає змогу обмежувати швидкість повороту сервопривода?

## Лабораторна робота №6. Дослідження роботи маніпулятора з дистанційним управлінням

Мета роботи: Ознайомитися з принципом управління роботом-маніпулятором. Навчитися створювати програми керування для дистанційного управління маніпулятором.

### Теоретичні відомості

У процесі автоматизації виробництв, зокрема біотехнологічних, все більшого поширення набуває використання роботизованих комплексів, що складаються з механічних маніпуляторів та систем управління ними. Застосування промислових роботів-маніпуляторів дозволяє виключити вплив людського фактора на виробництві, підвищити точність виконання операцій, певною мірою зменшити вплив шкідливих факторів на персонал, забезпечити безперебійну роботу виробництва.

Промисловий робот – автоматична машина, що складається з маніпулятора і пристрою програмного керування його рухом, призначений для заміни людини при виконанні основних і допоміжних операцій у виробничих процесах.

Маніпулятор – сукупність просторового важільного механізму і системи приводів, що здійснює під керуванням програмованого автоматичного пристрою чи людини-оператора дії (маніпуляції), аналогічні діям руки людини.

Промислові роботи призначені для заміни людини при виконанні основних і допоміжних технологічних операцій у процесі виробництва. Гнучкі автоматизовані виробництва, які створюються на базі промислових роботів, дозволяють вирішувати задачі автоматизації на підприємствах.

Маніпулятор промислового робота повинен забезпечувати рух вихідної ланки і, закріпленого в ній, об'єкта маніпулювання в просторі за заданою траєкторією і з заданою орієнтацією. Промисловий робот із шістьма ступенями свободи є складною автоматичною системою. У реальних конструкціях промислових роботів часто використовуються також механізми зі ступенями свободи менше шести. Найбільш прості маніпулятори мають три, рідше два, рухи. Такі маніпулятори значно дешевші у виготовленні та експлуатації, але вимагають специфічних

вимог до організації робочого простору.

Розглянемо для прикладу структурну і функціональну схеми промислового робота з трьома рухомими ланками. Основний механізм руки маніпулятора складається з нерухомої ланки *0* і трьох рухомих ланок *1, 2* і *3* (рис. 6.1).

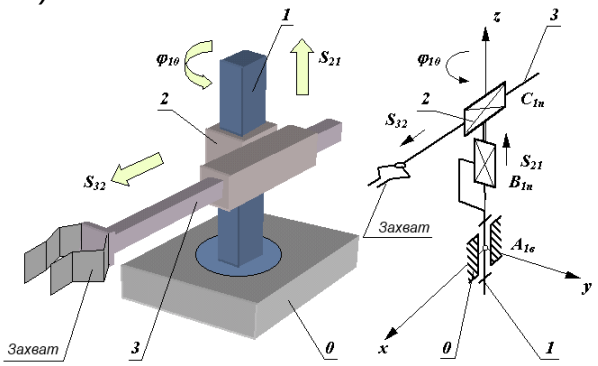



Рис. 6.1. Структурна і функціональна схеми промислового маніпулятора

Механізм цього маніпулятора відповідає циліндричній системі координат. У цій системі ланка *1* може обертатися відносно ланки *0* (відносне кутове переміщення $\varphi_{10}$), ланка *2* переміщається по вертикалі відносно ланки *1* (відносне лінійне переміщення $S_{21}$) і ланка *3* переміщається в горизонтальній площині відносно ланки *2* (відносне лінійне переміщення $S_{32}$). На кінці ланки *3* закріплений захоплюючий пристрій (захват), призначений для захоплення й утримання об'єкта маніпулювання. Ланки основного важільного механізму маніпулятора утворюють між собою три однорухливі кінематичні пари (одну обертальну *A* і дві поступальні *B* та *C*) і можуть забезпечити переміщення об'єкта в просторі без керування його орієнтацією.

Для виконання кожного з трьох відносних рухів маніпулятор повинен бути оснащений приводами, що складаються з двигунів, редуктора і системи давачів зворотного зв’язку. Оскільки рух об'єкта здійснюється за заданим законом руху, то в системі повинні бути пристрої, що зберігають і задають програму руху. Перетворення заданої програми руху в сигнали керування приводами $u_i$ здійснюється системою керування. При необхідності вона коректує ці впливи за сигналами $\Delta x_i$, що надходять до неї з давачів зворотного зв'язку (рис. 6.2).

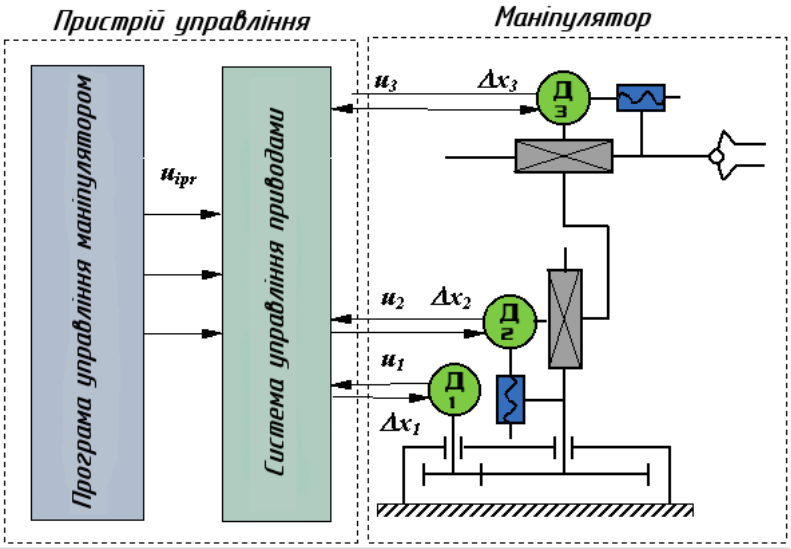



Рис. 6.2. Функціональна схема промислового робота

Отже, маніпуляційний робот складається з декількох ступенів рухливості (ланок) і приводів, що призводять ланки в рух. У якості приводів робота найчастіше використовуються крокові двигуни або сервоприводи різної потужності. Крововим двигунам надається перевага, якщо швидкість переміщення ланок робота не є критичним параметром. Наприклад, такий тип приводів може використовуватися при побудові вантажних маніпуляторів. Якщо ж потрібно забезпечити високу швидкість руху робота, то найбільш доцільно використовувати сервоприводи.

Процес розробки робота-маніпулятора включає два етапи:

- розробка механічної частини робота (вибір матеріалу для виготовлення складових частин, а також типу виконавчих механізмів);

- розробка системи управління маніпулятором (вибір контролера, вибір засобів програмування, розробка алгоритмів керування).

Швидкодію промислових роботів визначають за максимальною швидкістю лінійних переміщень центра захвату маніпулятора. Розрізняють промислові роботи з малою ($V_M < 0,5$ м/с), середньою ($0,5 < V_M < 1,0$ м/с) і високою ($V_M > 1,0$ м/с) швидкодією. Сучасні промислові роботи мають в основному середню і високу швидкодію.

Точність маніпулятора промислового робота характеризується абсолютною лінійною похибкою позиціювання центра захвату. Промислові роботи поділяються на групи з малою ($\Delta r_M < 1$ мм), середньою (0,1 мм $< \Delta r_M <$ 1 мм) і високою ($\Delta r_M < 0,1$ мм) точністю позиціювання.

**Опис дослідної установки**

Досліджуваний маніпулятор, зображений на рис. 6.3, має 5 ступенів свободи. Для приведення в рух ланок маніпулятора в кожному

із суглобів встановлено сервопривод MG996R. Перший ступінь рухомості забезпечує основа маніпулятора, другий – плече, третій – лікоть, четвертий – обертання кисті, п'ятий – поворот кисті, шостий – захват.

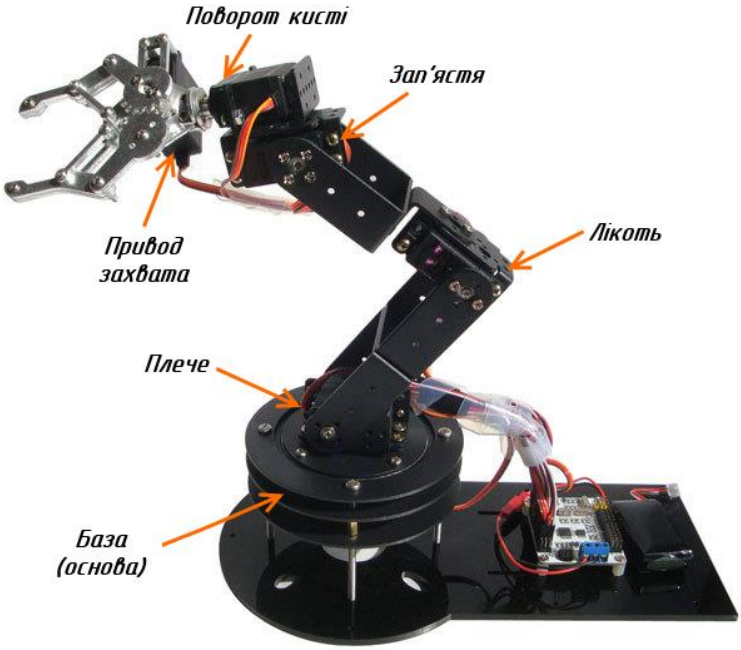



Рис. 6.3. Зовнішній вигляд досліджуваного маніпулятора

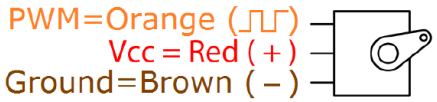
Параметри сервоприводів MG996R (рис. 6.4) подані нижче.


Рис. 6.4. Зовнішній вигляд та розміри сервопривода MG996R

Основні характеристики:

З'єднувальний провід: довжина 300мм.

Розміри: 40.7 мм x 19,7 мм x 42.9 мм.
Маса: 55 г
Робоча швидкість: 0.17с / 60 градусів (4.8В без навант.)
Робоча швидкість: 0.14с/ 60 градусів (6.0В без навант.)
Обертальний момент: 9.4 кгс·см (4.8В), 11 кгс·см (6 В)
Джерело живлення: через зовнішній адаптер.
Робоча напруга: 4.8 – 7.2V
Редуктор : металеві шестерні.
Робочий струм при русі: 500 – 900 мА (6 В).
Струм в загальмованому стані: 2.5 А (6 В).
Підключення:




Для живлення сервоприводів у практичній роботі використовується блок живлення 5В, 10А.

Система управління роботом-маніпулятором реалізована на основі мікроконтролерної плати Arduino Mega 2560.

**План роботи**

1. Ознайомитися з будовою та принципами управління роботами-маніпуляторами.
2. Ознайомитися з будовою дослідної установки.
3. Визначити межі безпечних кутів повороту (робочу зону) маніпулятора.

**Порядок виконання роботи**

1. Ознайомитися з теоретичними відомостями.
2. Завантажити середовище Arduino IDE.
3. Створити нову програму.

```
#include <VarSpeedServo.h>
VarSpeedServo armservo1, armservo2, armservo3, armservo4, armservo5, armservo6;  // create servo object to control a servo
byte a1,a2,a3,a4,a5,a6; //angle of Servo
byte oa1,oa2,oa3,oa4,oa5,oa6; //Old angle of Servo
boolean serialflag = 0; //

void setup()
{
  // initialize serial:
  Serial.begin(9600);

  armservo1.attach(2,740,2365); // attaches the servo on pin 2 to the servo object
  armservo2.attach(3,760,2365);  // attaches the servo on pin 3 to the
```

```
servo object
          armservo3.attach(4,720,2365);  // attaches the servo on pin 4 to the
servo object
          armservo4.attach(5,720,2300);  // attaches the servo on pin 5 to the
servo object
          armservo5.attach(10,670,2300);  // attaches the servo on pin 10 to
the servo object
          armservo6.attach(11,720,2300);  // attaches the servo on pin 11 to
the servo object
        oa1 = armservo1.read();
        oa2 = armservo2.read();
        oa3 = armservo3.read();
        oa4 = armservo4.read();
        oa5 = armservo5.read();
        oa6 = armservo6.read();
        }

        void loop()
        {
          if (a1 != oa1){setPosServo(armservo1,a1);oa1 = a1;printserial(); }
          if (a2 != oa2){setPosServo(armservo2,a2);oa2 = a2;printserial(); }
          if (a3 != oa3){setPosServo(armservo3,a3);oa3 = a3;printserial(); }
          if (a4 != oa4){setPosServo(armservo4,a4);oa4 = a4;printserial(); }
          if (a5 != oa5){setPosServo(armservo5,a5);oa5 = a5;printserial(); }
          if (a6 != oa6){setPosServo(armservo6,a6);oa6 = a6;printserial(); }
          if (serialflag == 1) {printserial(); serialflag = 0;}
          //printserial(); delay(200);
        }

        void setPosServo(VarSpeedServo armservo,int angle){
          armservo.write(angle,15,true);
        }

        void serialEvent() {
          byte armservo;
          while (Serial.available()) {
            // get the new byte:
            byte inByte = (byte)Serial.read();
            if ((inByte >= 201) && (inByte<=206)  ) {
            armservo = inByte - 200;
            }
            if (inByte <= 180) {
              if (armservo == 1) {a1 = inByte;}
              if (armservo == 2) {a2 = inByte;}
```

```
            if (armservo == 3) {a3 = inByte;}
            if (armservo == 4) {a4 = inByte;}
            if (armservo == 5) {a5 = inByte;}
            if (armservo == 6) {a6 = inByte;}
            serialflag = 1;
            armservo = 0;
          }
        }
      }

      void printserial() {
        delay(20);
        Serial.print(armservo1.read());//  Надіслати поточне положення привода 1
        Serial.print(",");
        Serial.print(armservo2.read());//  Надіслати поточне положення привода 2
        Serial.print(",");
        Serial.print(armservo3.read());//  Надіслати поточне положення привода 3
        Serial.print(",");
        Serial.print(armservo4.read());//  Надіслати поточне положення привода 4
        Serial.print(",");
        Serial.print(armservo5.read());//  Надіслати поточне положення привода 5
        Serial.print(",");
        Serial.print(armservo6.read());//  Надіслати поточне положення привода 6
        Serial.println();
        delay(20);
      }
```

4. Скомпілювати програму та переконатися у відсутності помилок.

5. Підключити мікроконтролерну плату до USB-порту комп'ютера.

6. У середовищі Arduino IDE вибрати тип плати та порт, до якого вона підключена. Завантажити створену програму в мікроконтролер.

7. Запустити програму RoboArm Control.

8. З дозволу викладача ввімкнути живлення стенда.

9. Плавно переміщуючи повзунки смуг прокрутки, спостерігати за переміщенням маніпулятора.

10. Визначити межі кута повороту кожного сервопривода. Результати записати в табл. 6.1.

Таблиця 6.1

| № з/п | Сервопривод | Нижня межа кута повороту | Верхня межа кута повороту |
|---|---|---|---|
| 1 | База | | |
| 2 | Плече | | |
| 3 | Лікоть | | |
| 4 | Зап'ястя | | |
| 5 | Поворот кисті | | |
| 6 | Захват | | |

11. Вимкнути живлення стенда.

12. Від'єднати плату Arduino від комп'ютера.

13. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; заповнену таблицю 5.1; висновки.

**Контрольні запитання**

1. Що таке промисловий робот?
2. Що таке маніпулятор?
3. Яка сфера застосування промислових роботів і маніпуляторів?
4. Наведіть приклад функціонально схеми промислового робота?
5. Які типи приводів використовують у промислових маніпуляторах?
6. Що таке ступені свободи маніпулятора?
7. Які приводи маніпулятора застосовують в лабораторному стенді? Який принцип управління цими приводами?

## Лабораторна робота №4 (БІО). Реалізація системи автоматичного керування роботом-маніпулятором

Мета роботи: навчитися програмувати роботу робота-маніпулятора за заданою програмою руху в автоматичному режимі.

### Теоретичні відомості

В більшості застосувань промислові роботи працюють в автоматичному, а не дистанційному режимі керування. Залежно від ступеня врахування й реагування на зовнішні впливи системи автоматичного керування роботами класифікують від найпростіших програмних (які виконують жорстко запрограмовану послідовність керуючих впливів заданої величини без врахування впливу збурень) до інтелектуальних самоорганізованих. Перші характеризуються меншою складністю програмування й моделювання роботи виробничої системи при її проектуванні.

У таких системах програма часто являє собою однотипну послідовність переміщень із заданою швидкістю в задане положення в задані моменти часу, тому може бути організована у вигляді циклу по масиву зі заданими переміщеннями, швидкостями, затримками. Оскільки кожна позиція маніпулятора вимагає подачі окремого сигналу керування на кожен сервопривод, то масив буде двохвимірний: один вимір визначатиметься кількістю керованих сервоприводів, а інший – кількістю виконуваних переміщень у нове положення за один робочий цикл.

```
#include <VarSpeedServo.h>
VarSpeedServo myservo[6];
int minImp[6]={900,720,720,720,670,720};
int maxImp[6]={2365,1000,2365,2300,2300,2300};
uint8_t curServoN;
const uint8_t positionsTotal=3;
uint8_t pos[6][positionsTotal]={\
  0,0,0,0,0,0,\
  120,20,30,25,0,30,\
  60,0,20,0,90,0};
uint8_t speed[6][positionsTotal]={\
  15,20,20,20,30,20,\
  15,20,20,20,30,20,\
  15,20,20,20,30,30};
bool wait[6][positionsTotal]={\
  false,false,false,false,false,true,\
  false,false,true,false,true,true,\
  true,false,true,false,true,true,};
int delays[positionsTotal]={4000,5000,3000};
```

```
void setup() {
  myservo[0].attach(2, minImp[0], maxImp[0]);
  myservo[1].attach(3, minImp[1], maxImp[1]);
  myservo[2].attach(4, minImp[2], maxImp[2]);
  myservo[3].attach(5, minImp[3], maxImp[3]);
  myservo[4].attach(10, minImp[4], maxImp[4]);
  myservo[5].attach(11, minImp[5], maxImp[5]);
  for(curServoN=0;curServoN<6;curServoN++) {
    myservo[curServoN].write(0, 25, false);
  }
  delay(1000);
}

void loop() {
  for(uint8_t i=0;i<positionsTotal;i++) {
    for(curServoN=0;curServoN<6;curServoN++) {
      myservo[curServoN].write(pos[curServoN][i], speed[curServoN][i],
wait[curServoN][i]);
    }
    delay(delays[i]);
  }
}
```

**План роботи**

1. Ознайомитися з прикладом програми для автоматичного програмного керування роботом-маніпулятором.

2. Написати програму, що реалізує переміщення маніпулятора в задані положення в автоматичному режимі.

**Порядок виконання роботи**

1. Ознайомитися з прикладом програми для автоматичного програмного керування роботом-маніпулятором у теоретичних відомостях. За потреби змінити мінімальну і максимальну тривалість імпульсів керування відповідно до використаних у роботі сервоприводів. Завантажити програму в контролер робота. З дозволу викладача подати живлення на сервоприводи.

2. Додати ще одну точку в траекторії руху захвату робота.

3. Змінити кути повороту сервопривода у програмі так, щоб маніпулятор захоплював об’єкт із однієї точки й переносив його в іншу точку лабораторного стенда.

4. Зменшити затримки утримання робота в заданій точці та швидкості повороту сервоприводів (другий аргумент методу write). Порівняти тривалість робочого циклу робота до і після зміни.

5. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. Яку величину задає перший аргумент методу VarSpeedServo.write()?
2. Яку величину задає другий аргумент методу VarSpeedServo.write()?
3. Що задає третій аргумент методу VarSpeedServo.write()?
4. Як задати тривалості імпульсів керування сервоприводом за допомогою бібліотеки VarSpeedServo?
5. Як у програмі вказати номер контакту, на який потрібно генерувати сигнал керування сервоприводом?

## Лабораторна робота 7. Реалізація захисту і блокування роботи маніпулятора при виявленні перешкод у автоматичному режимі

Мета роботи: навчитися програмувати роботу мехатронної системи в автоматичному режимі та зупинку роботи при виявленні в робочій зоні тіла людини або іншої перешкоди за допомогою датчиків руху.

**Теоретичні відомості**

Для виявлення в робочій зоні маніпулятора сторонніх об’єктів широко застосовують датчики руху, датчики відстані, датчики присутності об’єкта, відеокамери в поєднанні з засобами машинного зору.

Датчик руху фіксує переміщення об'єктів і використовується для контролю або автоматичного запуску необхідних дій у відповідь на переміщення об'єкта. На нерухомі об’єкти датчики руху, як правило, не реагують. Найчастіше використовуються інфрачервоні, ультразвукові, радіохвильові датчики руху.

Для виявлення людського тіла здебільшого використовують пасивні інфрачервоні давачі руху. Вони фіксують зміну інфрачервоного випромінювання, що потрапляє в датчик з різних точок простору (зміну теплового фону), й здатні виявити рух теплокровних тварин або інших об’єктів, які досить інтенсивно випромінюють у діапазоні хвиль, відмінному від оточення.

***Давач руху HC-SR501***

Для виконання лабораторної роботи використовується пасивний інфрачервоний давач руху HC-SR501 (рис. 7.1).

Рис. 7.1. Зовнішній вигляд давача руху HC-SR501

Він складається з піроелектричного датчика (містить у собі 2 піроелектричні елементи), лінзи Френеля та схеми вторинного перетворення сигналу. Лінза Френеля забезпечує *значну* зміну освітленості різних частин піроелектричного елемента при *незначному* русі теплого об'єкта внаслідок *дифракції* й формування максимумів та мінімумів освітленості на поверхні чутливого елемента. Різниця напруг з двох елементів опрацьовується мікросхемою BISS0001, в результаті на виході датчика при виявленні руху зміниться логічний рівень сигналу.


Рис. 7.2. Призначення контактів датчика руху HC-SR501

Активний рівень сигналу на виході утримується впродовж часу, що задається підстроювальним резистором (рис. 7.2.). Інший підстроювальний резистор задає чутливість давача.

**План роботи**

1. Ознайомитися з документацією на давач руху HC-SR501.
2. Написати програму, що реалізує зупинку маніпулятора при виявленні руху руки людини в зоні дії маніпулятора.

## Порядок виконання роботи

1. Відкрити документацію на датчик руху HC-SR501 та знайти межі тривалості імпульсу, що генерується при виявленні руху. За допомогою викрутки задати мінімальну тривалість.

2. Скласти схему та написати програму для Arduino, яка запалює світлодіод при виявленні датчиком руху та відлагодити її роботу. За основу можна взяти програму Button з прикладів середовища Arduino IDE.

3. Аналогічну функціональність реалізувати з використанням механізму зовнішніх апаратних переривань.

4. На основі програми з попереднього пункту й програми з попередньої роботи скласти програму, яка в автоматичному режимі (не чекаючи даних від ПК) здійснюватиме переміщення маніпулятора між як мінімум троьма положеннями й блокуватиме переміщення при виявленні руху датчиком руху. Після перевірки програми та правильності підключення датчика викладачем завантажити програму в плату керування маніпулятором та перевірити правильність роботи.

```
#include <VarSpeedServo.h> //підключаємо бібліотеку для роботи з сервоприводами з регульованою швидкістю
VarSpeedServo armservo1, armservo2, armservo3, armservo4, armservo5, armservo6; //створюємо відповідні програмні об’єкти
const byte interruptPin = 21; //передбачаємо контакт для сигналу зовнішнього переривання
void setup() {
  armservo1.attach(2, 900, 2365); //задаємо тривалості імпульсів керування кожним сервоприводом
  armservo2.attach(3, 720, 1000);
  armservo3.attach(4, 720, 2365);
  armservo4.attach(5, 720, 2300);
  armservo5.attach(10, 670, 2300);
  armservo6.attach(11, 720, 2300);
attachInterrupt(digitalPinToInterrupt(interruptPin), stopMove, CHANGE); //налаштовуємо переривання по будь-якій зміні сигналу, обробник переривання – функція stopMove()
  pinMode(interruptPin, INPUT); //нала-штовуємо відповідний контакт на вхід
//задаємо початкове положення сервоприводів: аргумент false дозволяє виконати одночасне переміщення всіх ланок; при значенні true сервоприводи вмикалися б по черзі
  armservo1.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не очікувати закінчення руху, а одразу переходити до виконання наступного рядка
```

```
        armservo2.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не очікувати закінчення руху, а одразу переходити до виконання наступного рядка
        armservo3.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не очікувати закінчення руху, а одразу переходити до виконання наступного рядка
        armservo4.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не очікувати закінчення руху, а одразу переходити до виконання наступного рядка
        armservo5.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не очікувати закінчення руху, а одразу переходити до виконання наступного рядка
        armservo6.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не очікувати закінчення руху, а одразу переходити до виконання наступного рядка
        delay(4000); //пауза перед переходом до робочого цилку
      }
      //функція обробки переривання викликається при зміні логічного рівня на вході 21 плати Arduino Mega, спричиненого датчиком руху
      void stopMove() {
        armservo1.stop();
        armservo2.stop();
        armservo3.stop();
        armservo4.stop();
        armservo5.stop();
        armservo6.stop();
      }
      void loop() {
      //перша позиція в робочому циклі
        armservo1.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не очікувати закінчення руху, а одразу переходити до виконання наступного рядка
        armservo2.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не очікувати закінчення руху, а одразу переходити до виконання наступного рядка
        armservo3.write(0, 20, false); // 0 – задане положення/кут повороту, 20 – швидкість руху до заданого положення, false – не
```

```
очікувати закінчення руху, а одразу переходити до виконання
наступного рядка
        armservo4.write(0, 20, false); // 0 – задане положення/кут
повороту, 20 – швидкість руху до заданого положення, false – не
очікувати закінчення руху, а одразу переходити до виконання
наступного рядка
        armservo5.write(0, 20, false); // 0 – задане положення/кут
повороту, 20 – швидкість руху до заданого положення, false – не
очікувати закінчення руху, а одразу переходити до виконання
наступного рядка
        armservo6.write(0, 20, false); // 0 – задане положення/кут
повороту, 20 – швидкість руху до заданого положення, false – не
очікувати закінчення руху, а одразу переходити до виконання
наступного рядка
        delay(3000); //затримка для закінчення повороту сервоприводів
(якщо з попередньої ітерації циклу вони були не в початковому
нульовому положенні)
        armservo2.write(68, 20, false); //нахил плеча вперед
        delay(3000);
        armservo6.write(51, 20, false); //захоплення предмета (armservo6
– сервопривод захвату)
        delay(3000);
        armservo2.write(0, 20, false); //повернення плеча в попереднє
положення
        delay(3000);
      //переміщення в нову точку
        armservo3.write(39, 15, true); // 39 – задане положення/кут
повороту, 15 – швидкість руху до заданого положення, true – очікувати
закінчення руху, лише потім переходити до виконання наступного рядка
        armservo1.write(46, 15, true); // 46 – задане положення/кут
повороту, 15 – швидкість руху до заданого положення, true – очікувати
закінчення руху, лише потім переходити до виконання наступного рядка
        armservo2.write(40, 15, false); // 40 – задане положення/кут
повороту, 15 – швидкість руху до заданого положення, false – не
очікувати закінчення руху, а одразу переходити до виконання
наступного рядка
        delay(3000);
        armservo2.write(78, 15, false); //нахил плеча
        delay(3000);

        armservo6.write(0, 20, false); //відпускання предмета
        delay(3000);

      //піднімання плеча й поворот платформи в початкову позицію
```

armservo2.write(0, 15, true);  // 0 – задане положення/кут повороту, 15 – швидкість руху до заданого положення, true –очікувати закінчення руху, лише потім переходити до виконання наступного рядка

//Якщо одразу перейти до повороту платформи без очікування підйому плеча, при виконанні повороту можна зачепити низькорозташовані об’єкти

armservo1.write(0, 20, true); //поворот платформи в початкове положення

delay(3000);

//переведення решти сервоприводів маніпулятора в початкове положення

armservo2.write(0, 20, false);
armservo3.write(0, 20, false);
armservo4.write(0, 20, false);
armservo5.write(0, 20, false);
armservo6.write(0, 20, false);

delay(10000); //пауза перед повторенням операцій у наступній ітерації головного циклу

}

5. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. Які види давачів руху ви знаєте?
2. Який принцип дії пасивного інфрачервоного датчика руху?
3. Яке фізичне явище використовується в датчику HC-SR501 для виявлення руху?
4. Яке призначення підстроювальних резисторів в HC-SR501?
5. Для чого використовується лінза Френеля в PIR датчиках руху?
6. В якому випадку пасивний інфрачервоний датчик може не виявити частину тіла людини, що рухається в полі зору датчика?

## Лабораторна робота №8. Виведення даних на дисплей рухомої платформи

Мета роботи: навчитись використовувати засоби індикації в мехатронних об’єктах.

### Теоретичні відомості

Для взаємодії з людиною в мехатронних системах використовуються різноманітні засоби індикації: дискретні (одиночні) світлодіоди для сигналізації стану/режиму (горить/погашений/блимає), сегментні світлодіодні індикатори для виведення чисел, світлодіодні матриці для виведення графічних об’єктів, рідкокристалічні та OLED-дисплеї тощо.

У даній роботі використовується робот Keyestudio KS0428 Mini Tank Robot (рис. 8.1) – це мініатюрний та доступний багатоцільовий робот, розроблений для освітніх цілей та творчого експериментування.

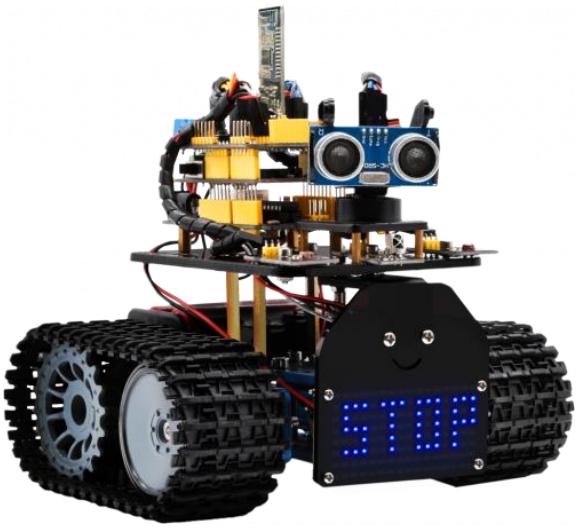



Рис. 8.1. Keyestudio KS0428 Mini Tank Robot

Складові Keyestudio KS0428 Mini Tank Robot V2:

1) Arduino Uno-сумісна мікроконтролерна плата (рис. 8.2), що базується на мікроконтролері ATmega328P та обладнана конвертером UART-USB CP2102.

2) Два двигуни, кожен з яких керується окремо, що дозволяє легко керувати рухом робота, здійснювати повороти та маневри.

3) L298P Shield – плата з драйвером двигунів для Arduino, спроектована для управління двома двигунами постійного струму.

4) Sensor Shield V5 – це плата для Arduino, яка дозволяє під'єднувати до Arduino різноманітні датчики. Плата має 16 контактів для підключення датчиків, а також ряд інших функцій, які полегшують використання датчиків.

5) Серводвигун — це поворотний привод для керування положенням платформи з ультразвуковим датчиком.

6) Ультразвуковий датчик HC-SR04 – датчик, що виявляє перешкоди та об'єкти на шляху, що дозволяє створювати автономних роботів з функцією обходу перешкод.

7) Bluetooth модуль HM-10 – дозволяє керувати роботом бездротово з смартфона або планшета.

8) 8x16 LED-матриця – матричний світлодіодний дисплей 8x16 з контролером/драйвером AIP1640, підключений через інтерфейс I2C, дозволяє виводити різноманітну інформацію, зображення, виконувати анімацію.

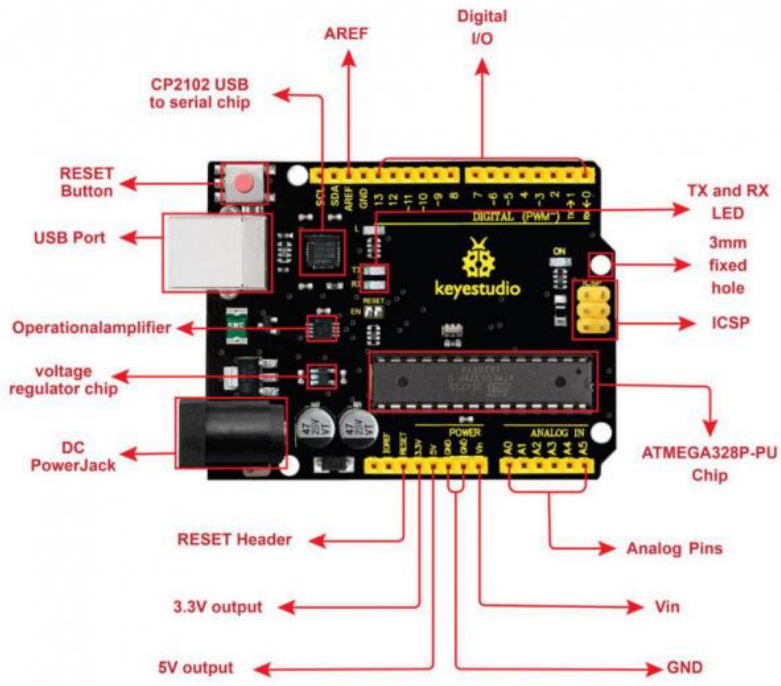



Рис. 8.2. Компоненти плати Keyestudio V4.0

8x16 світлодіодна матриця використовується для відображення символьної інформації або зображень. Вона складається з 8 рядків і 16 стовпців світлодіодів, що в сумі дає 128 світлодіодів (рис. 8.3). Для керування 8x16 LED панеллю використовується контролер AIP1640. Він отримує дані від комп'ютера або іншого пристрою і використовує ці дані для управління світлодіодами.

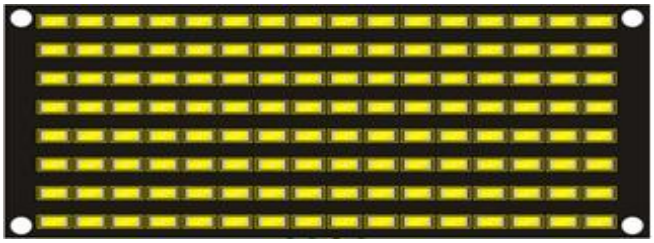


Рис. 8.3. 8x16 світлодіодна матриця

Матриця світлодіодів може використовуватись відображення різноманітного тексту, такого як час, дата, температура чи інша інформація, а також відображення простих зображень, таких як логотипи чи значки.

Оскільки включенням-виключенням світлодіодів займається контролер, розробнику потрібно запрограмувати завантаження стану кожного пікселя (світлодіода) в контролер. Це зводиться до завантаження масиву 8x16=128 бітів, об'єднаних у байти. Щоб не визначати значення кожного байта вручну, можна скористатись сервісами на зразок http://dotmatrixtool.com/. На рис. 8.4. показано процес створення зображення 8x16 та генерування відповідного масиву типу uint_8.

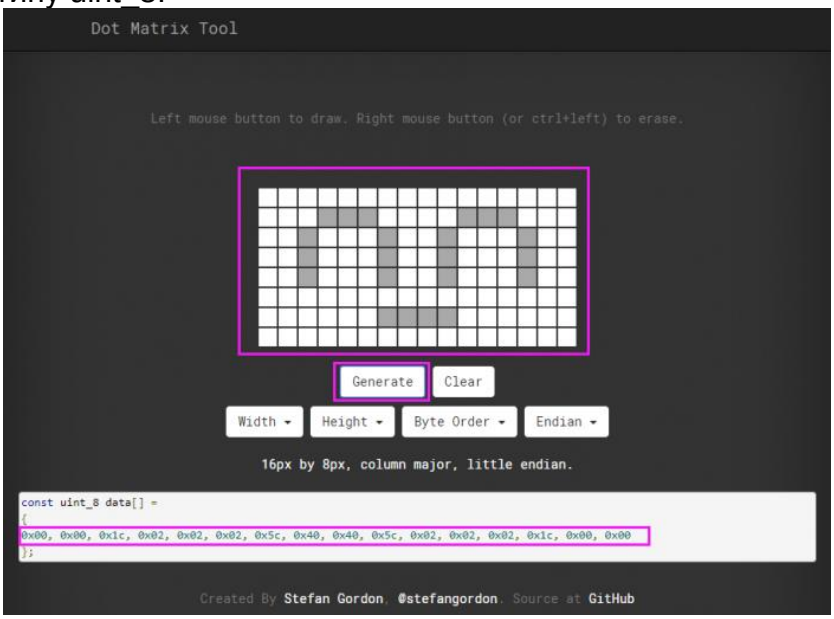



Рис. 8.4. Приклад генерування масиву, відповідного зображенню

Дані, які використовуються для керування 8x16 LED панеллю, представлені у вигляді послідовності байтів. Кожен байт представляє один стовпчик світлодіодів. Змінюючи програмно значення у масиві, що надсилається на контролер дисплея, можна змінювати виведене зображення (рис. 8.5).

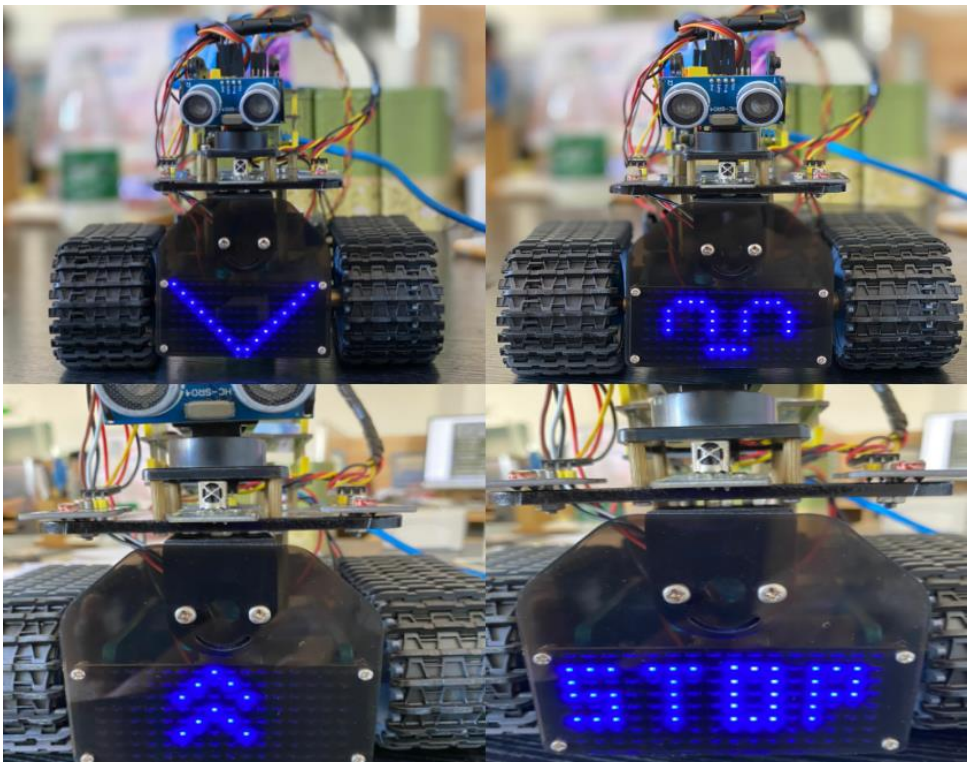



Рис. 8.5. Виведене на дисплей зображення

**План роботи**

1. Підключити дисплей до Arduino Uno та вивести на нього задане зображення.

2. Вивести на дисплей поточну відстань до перешкоди.

**Порядок виконання роботи**

1. Підключити дисплей до плати Arduino Uno через Sensor Shield згідно рис. 8.6.

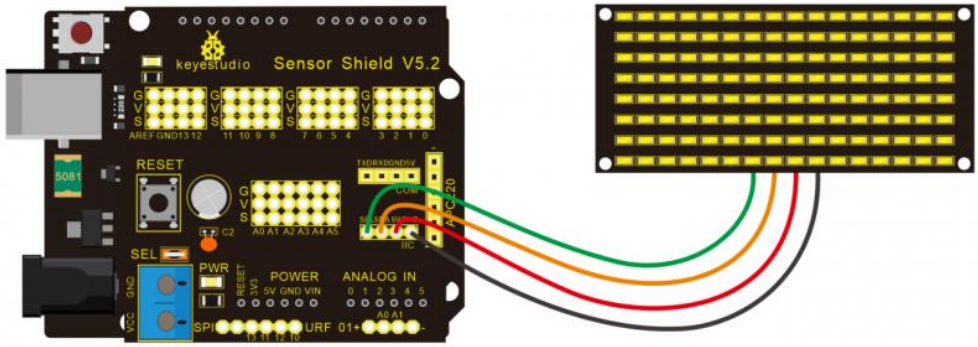



Рис. 8.6. Підключення світлодіодного матричного дисплея

2. Завантажити у мікроконтролер програму:

unsigned char start01[] =

```
        {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0
x08,0x04,0x02,0x01};
        unsigned char front[] =
        {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0
x00,0x00,0x00,0x00};
        unsigned char back[] =
        {0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0
x00,0x00,0x00,0x00};
        unsigned char left[] =
        {0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0
x44,0x28,0x10,0x00};
        unsigned char right[] =
        {0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0
x00,0x00,0x00,0x00};
        unsigned char STOP01[] =
        {0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,
0x3E,0x0A,0x0E,0x00};
        unsigned char clear[] =
        {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00};
        #define SCL_Pin  A5  //Set clock pin to A5
        #define SDA_Pin  A4  //Set data pin to A4
        void setup(){
          pinMode(SCL_Pin,OUTPUT);
          pinMode(SDA_Pin,OUTPUT);
          //очистка дисплея
          matrix_display(clear);
        }
        void loop(){
          matrix_display(start01);  //показати зображення з масиву start01
          delay(2000);
          matrix_display(front);    // показати зображення з масиву front
          delay(2000);
          matrix_display(STOP01);  //показати зображення з масиву
STOP01
          delay(2000);
          matrix_display(clear);      //очистка дисплея
          delay(2000);
        }

        void matrix_display(unsigned char matrix_value[])
        {
          IIC_start();  //формування старт-біта
          IIC_send(0xc0);  //передача адреси
```

```
  for(int i = 0;i < 16;i++) //для кожного з 16 байт
  {
    IIC_send(matrix_value[i]); //надіслати дані в I2C
  }
  IIC_end();   //формування кінця передачі
  IIC_start();
  IIC_send(0x8A);  //ширина імпульсів 4/16
  IIC_end();
}

void IIC_start()
{
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}

void IIC_send(unsigned char send_data)
{
  for(char i = 0;i < 8;i++)  //Each byte has 8 bits
  {
      digitalWrite(SCL_Pin,LOW);
      delayMicroseconds(3);
      if(send_data & 0x01)  //виділення молодшого біта
      {
        digitalWrite(SDA_Pin,HIGH);
      }
      else
      {
        digitalWrite(SDA_Pin,LOW);
      }
      delayMicroseconds(3);
      digitalWrite(SCL_Pin,HIGH);
      delayMicroseconds(3);
      send_data = send_data >> 1;  //побітовий зсув праворуч
  }}
void IIC_end()
{
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
```

```
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);}
```

3. Доповнити програму формуванням зображень, що означають: 1) очікування на нову команду; 2) підтримання дистанції до об’єкта попереду; 3) об’їзд перешкод; 4) низький заряд акумуляторів.

4. Написати програму, що виводить на дисплей ваше ім’я. Перевірити її роботу.

5. Реалізувати таймер, що виводить на дисплей кількість секунд 0…60 з моменту запуску програми.

6. Написати програму, яка забезпечує вимірювання відстані за допомогою ультразвукового датчика й виведення її числового значення на дисплей.

7. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. Які засоби виведення інформації користувачу ви знаєте?
2. Що являє собою світлодіодна матриця?
3. Яка роль контролера (драйвера) дисплея?
4. Як кодується стан пікселів дисплея, використаного в лабораторній роботі?
5. Який інтерфейс використовується для передачі даних між мікроконтролером і контролером дисплея?
6. Які переваги і недоліки світлодіодних матричних індикаторів?

## Лабораторна робота №9. Керування рухом гусеничної платформи

Мета роботи: навчитись керувати приводами гусеничного робота для руху його вперед, назад, ліворуч, праворуч.

### Теоретичні відомості

У роботі Keyestudio Mini Tank Robot приводами є колекторні двигуни постійного струму, що живляться від 2 літій-іонних акумуляторів типорозміру 18650. Для зміни напруги на приводах використовується силова мікросхема – драйвер двигунів STMicroelectronics L298P.

Драйвер двигунів L298P з режимом керування швидкістю за допомогою широтно-імпульсної модуляції). L298P Shield (рис. 9.1) надає можливість використовувати два колекторні двигуни постійного струму, підключивши їх до клем Motor A та Motor B.

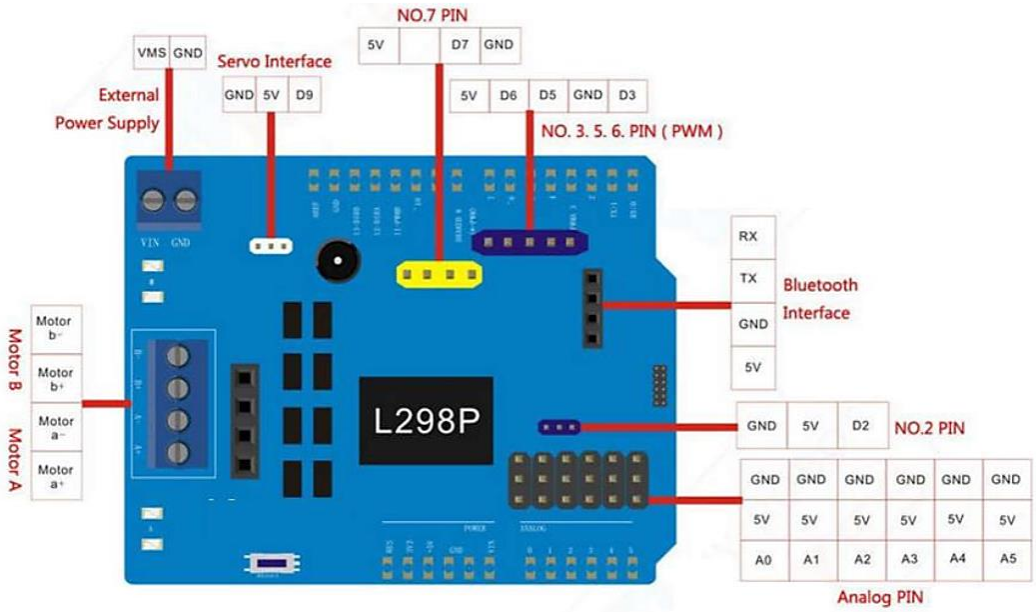



Рис. 9.1. Модуль драйвера L298P Shield

Характеристики:

1) Вхідна напруга логічної частини: DC 5 В.
2) Вхідна напруга провідної частини: DC 7-12 В.
3) Робочий струм логічної частини: <36 мА.
4) Робочий струм силової частини: <2А на канал.
5) Максимальна розсіювана потужність: 25 Вт (T=75 ℃).
6) Робоча температура: -25 ℃ ~ + 130 ℃.
7) Рівень вхідного сигналу керування: високий рівень 2,3 В <$V_{in}$ <5 В, низький рівень -0,3 В <$V_{in}$ <1,5 В.

На платі розміщено 8 високошвидкісних діодів Шоткі для захисту від стрибків напруги при закритті силових транзисторів. Також на платі розміщено перемичку "PWR". Ця перемичка використовується для вибору живлення для L298P. У випадку, коли перемичка "PWR" встановлена у положення "ON", плата L298P Shield отримує живлення від внутрішнього джерела 5 В від Arduino. У випадку, коли перемичка "PWR" встановлена у положення "OFF", живлення для плати L298P Shield надходить із зовнішнього джерела.

ШІМ-сигнал керування швидкістю лівого двигуна повинен надходити на контакт 11, правого – контакт 3. Напрямок обертання задається логічним рівнем на контактах 13 (лівий привод) та 12 (правий привод).

Для полегшення підключення датчиків, сервоприводів й інших електронних компонентів на L298P Shield зверху встановлюється Arduino Sensor Shield V5 (рис. 9.2).

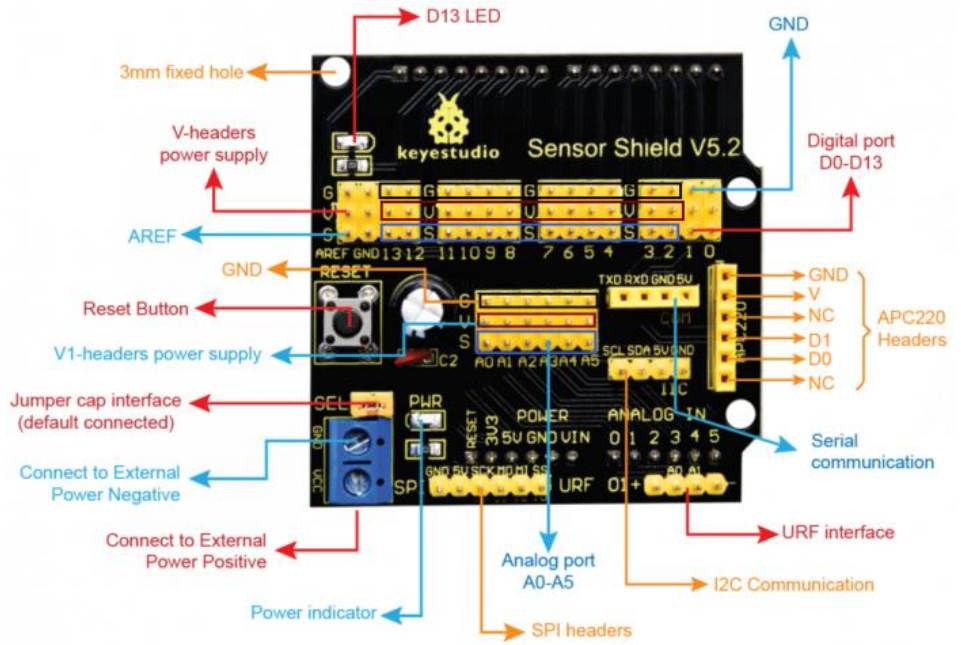



Рис. 9.2. Призначення елементів Sensor Shield V5

**План роботи**

1. Підключити двигуни через драйвер двигуна до Arduino Uno.
2. Запрограмувати і перевірити роботу приводів у різних режимах.

**Порядок виконання роботи**

1. Використовуючи інформацію в теоретичних відомостях, підключити двигуни до плати з драйвером двигунів, а її – до плати Arduino Uno. Перед підключенням впевнитись, що попередня програма, наявна в мікроконтролері, не використовує контакти керування двигунами.

2. Завантажити в контролер робота програму:

```
#define ML_Ctrl 13  //контакт контролю напрямку лівого приводу
#define ML_PWM 11     //контакт контролю швидкості лівого приводу
#define MR_Ctrl 12  //контакт контролю напрямку правого приводу
#define MR_PWM 3     // контакт контролю швидкості правого приводу

void setup()
{
  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
}

void loop()
{
  //вперед
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,200);//задати тривалість імпульсів ШІМ 200 з 255 для лівого привода
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,200);  //задати тривалість імпульсів ШІМ 200 з 255 для правого привода
  delay(2000);
   //назад
  digitalWrite(ML_Ctrl,HIGH);
  analogWrite(ML_PWM,200);
  digitalWrite(MR_Ctrl,HIGH);
  analogWrite(MR_PWM,200);
  delay(2000);
   //ліворуч
  digitalWrite(ML_Ctrl,HIGH);
  analogWrite(ML_PWM,200);
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,200);
  delay(2000);
   //праворуч
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,200);
  digitalWrite(MR_Ctrl,HIGH);
  analogWrite(MR_PWM,200);
  delay(2000);
```

```
  //зупинка
  analogWrite(ML_PWM,0);
  analogWrite(MR_PWM,0);
  delay(2000);
}
```

3. Перевірити відповідність рухів робота програмі.

4. Підібрати порогове значення скважності ШІМ-сигналу, менше якого приводи не обертаються. Побудувати графік залежності швидкості робота від коефіцієнта заповнення ШІМ.

5. Визначити тривалість включення приводів, потрібну для повороту на 30°, 45°, 90°, 120°, 180°.

6. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. Які приводи можуть використовуватись і мехатроніці?

2. Як змінити напрямок обертання колекторного двигуна постійного струму?

3. Як змінити швидкість обертання колекторного двигуна постійного струму?

4. Яка силова мікросхема використовується для керування напругою на двигуні в лабораторній роботі?

5. Яке призначення контактів мікросхеми-драйвера двигуна, що використовується в лабораторній роботі?

6. Навіщо потрібні швидкодіючі захисні діоди?

7. Яка функція використовується для зміни швидкості обертання двигуна?

8. Чим програмно відрізняється керування швидкістю правого привода й лівого?

## Лабораторна робота №10. Дистанційне керування роботом через IR-приймач

Мета роботи: навчитись використовувати інфрачервоний пульт й інфрачервоний приймач для передачі команд керування роботом.

**Теоретичні відомості**

Інфрачервоний пульт дистанційного керування всюдисущий у повсякденному житті. Використовується для управління різними побутовими приладами, такими як телевізори, музичні центри, відеомагнітофони та приймачі супутникового сигналу. Інфрачервона система дистанційного керування складається з передавача (випромінювача інфрачервоного випромінювання) та інфрачервоного приймального модуля, а також мікроконтролера, здатного декодувати прийнятий сигнал.

Інфрачервоний несучий сигнал частотою 38 кГц, який випромінює пульт дистанційного керування, формується мікросхемою

кодування на пульті дистанційного керування. Він складається з преамбули, адреси, інвертованої адреси, даних й інвертованих даних. Логічна 1 та логічний 0 відрізняються часовим інтервалом між імпульсами. Адреса одного й того ж пульта дистанційного керування не змінюється. Поле даних може містити код натиснутої кнопки. Коли натискається кнопка дистанційного керування, пульт дистанційного керування надсилає інфрачервоний сигнал. Коли IR-приймач отримує сигнал, програма розшифровує несучий сигнал і визначає, яка клавіша натиснута. MCU декодує отриманий сигнал 0/1, оцінюючи, яка клавіша натиснута на пульті дистанційного керування.

Модуль інфрачервоного приймача (рис. 10.1) містить головку інфрачервоного приймача, яка об'єднує прийом, підсилення та демодуляцію. Його вбудована мікросхема після демодуляції виводить прийнятий код у вигляді TTL-сумісного сигналу. Інфрачервоний приймальний модуль у лабораторній роботі має три контакти: сигнальну лінію, VCC і GND.

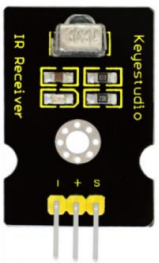



Рис. 10.1. Модуль інфрачервоного приймача

Коди кнопок пульта дистанційного керування, що використовується в лабораторній роботі, наведені на рис. 10.2.

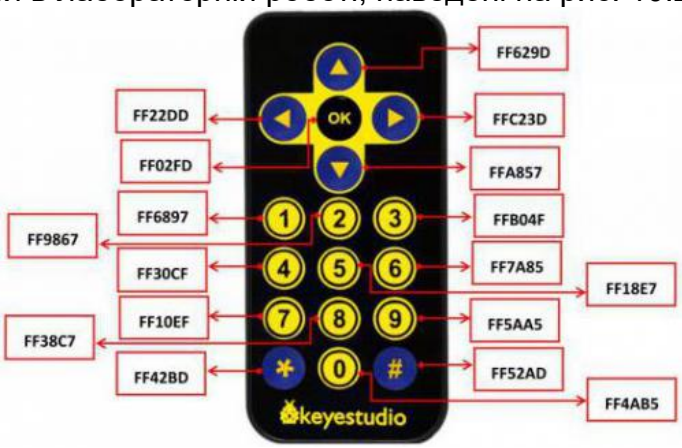



Рис. 10.2. Коди кнопок пульта дистанційного керування

**План роботи**

1. Ознайомитись із складовими інфрачервоної системи дистанційного керування й кодами кнопок.

2. Запрограмувати керування дискретним навантаженням за командами з пульта.

3. Запрограмувати керування роботом за командами з пульта.

**Порядок виконання роботи**

1. Завантажити бібліотеку IRremoteTank з https://www.dropbox.com/sh/yfip8hguyfmi54m/AABiWhMX0TiZnUiSaz4hs9-Ka/3.%20Tutorial%20-Arduino/2.%20Libraries?dl=0&subfolder_nav_tracking=1 та встановити у середовище Arduino.

2. Підключити інфрачервоного приймач до контролера робота, вихідний сигнал подати на аналоговий вхід A0.

3. Завантажити програму, яка надсилатиме у послідовний інтерфейс дані, прийняті від ІЧ-пульта:

```
#include <IRremoteTank.h>
int RECV_PIN = A0;
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
        {
         Serial.begin(9600);
         irrecv.enableIRIn(); //ввімкнення прийому
        }
 void loop() {
          if (irrecv.decode(&results))//якщо декодування завершилось успішно, зчитати отримані дані
          {
            Serial.println(results.value, HEX);//представити у шістнадцятковому форматі отримані дані й надіслати відповідний рядок символів у послідовний інтерфейс
             irrecv.resume(); // відновити прийом для отримання наступних даних
          }
          delay(100);
        }
```

4. Модифікувати програму з попереднього завдання так, щоб при натисканні обраної кнопки вмикалось навантаження (запалювався світлодіод), а при натисканні іншої – вимикався.

5. Об’єднати програму з попереднього пункту з програмою з попередньої лабораторної роботи так, щоб робот виконував команди «вперед», «назад», «ліворуч», «праворуч», «стоп». Асоціації кодів кнопок і виконуваних дій можна взяти з таблиці https://wiki.keyestudio.com/Ks0428_keyestudio_Mini_Tank_Robot_V2#Project_13:_IR_Remote_Robot_Tank.

```
#include <IRremoteTank.h>
```

```
IRrecv irrecv(A0);
decode_results results;
long ir_rec;
unsigned char start01[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned char front[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char back[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char left[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00};
unsigned char right[] = {0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char STOP01[] = {0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00};
unsigned char clear[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
#define SCL_Pin  A5
#define SDA_Pin  A4
#define ML_Ctrl 13
#define ML_PWM 11
#define MR_Ctrl 12
#define MR_PWM 3
#define servoPin 9
int pulsewidth;

void setup(){
  Serial.begin(9600);
  irrecv.enableIRIn();

  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  matrix_display(clear);
```

```
  matrix_display(start01);
  pinMode(servoPin, OUTPUT);
  procedure(90);
 }

 void loop(){
  if (irrecv.decode(&results))
  {
   ir_rec=results.value;
   String type="UNKNOWN";
   String typelist[14]={"UNKNOWN", "NEC", "SONY", "RC5", "RC6",
"DISH", "SHARP", "PANASONIC", "JVC", "SANYO", "MITSUBISHI",
"SAMSUNG", "LG", "WHYNTER"};
   if(results.decode_type>=1&&results.decode_type<=13){
    type=typelist[results.decode_type];
   }
   Serial.print("IR TYPE:"+type+"  ");
   Serial.println(ir_rec,HEX);
   irrecv.resume();
  }

  if (ir_rec == 0xFF629D)
  {
   Car_front();
   matrix_display(front);
  }
  if (ir_rec == 0xFFA857)
  {
   Car_back();
   matrix_display(front);
  }
  if (ir_rec == 0xFF22DD)
  {
   Car_T_left();
   matrix_display(left);
  }
  if (ir_rec == 0xFFC23D)
  {
   Car_T_right();
   matrix_display(right);
  }
  if (ir_rec == 0xFF02FD)
  {
   Car_Stop();
```

```
    matrix_display(STOP01);
  }
  if (ir_rec == 0xFF30CF)
  {
    Car_left();
    matrix_display(left);
  }
  if (ir_rec == 0xFF7A85)
  {
    Car_right();
    matrix_display(right);
 }
}
void procedure(int myangle) {
  for (int i = 0; i <= 50; i = i + (1)) {
    pulsewidth = myangle * 11 + 500;
    digitalWrite(servoPin,HIGH);
    delayMicroseconds(pulsewidth);
    digitalWrite(servoPin,LOW);
    delay((20 - pulsewidth / 1000));
  }
}
void matrix_display(unsigned char matrix_value[])
{
  IIC_start();
  IIC_send(0xc0);
   for(int i = 0;i < 16;i++)
  {
     IIC_send(matrix_value[i]);
  }
  IIC_end();

  IIC_start();
  IIC_send(0x8A);
  IIC_end();
}
void IIC_start()
{
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
```

```
}
void IIC_send(unsigned char send_data)
{
  for(char i = 0;i < 8;i++)
  {
      digitalWrite(SCL_Pin,LOW);
      delayMicroseconds(3);
      if(send_data & 0x01)
      {
        digitalWrite(SDA_Pin,HIGH);
      }
      else
      {
        digitalWrite(SDA_Pin,LOW);
      }
      delayMicroseconds(3);
      digitalWrite(SCL_Pin,HIGH);
      delayMicroseconds(3);
      send_data = send_data >> 1;
  }
}
void IIC_end()
{
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
}
void Car_front()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,200);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,200);
}
void Car_back()
{
  digitalWrite(MR_Ctrl,HIGH);
  analogWrite(MR_PWM,200);
  digitalWrite(ML_Ctrl,HIGH);
```

```
  analogWrite(ML_PWM,200);
}
void Car_left()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,255);
  digitalWrite(ML_Ctrl,HIGH);
  analogWrite(ML_PWM,255);
}
void Car_right()
{
  digitalWrite(MR_Ctrl,HIGH);
  analogWrite(MR_PWM,255);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,255);
}
void Car_Stop()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,0);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,0);
}
void Car_T_left()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,255);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,180);
}
void Car_T_right()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,180);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,255);
}
```

6. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. Які компоненти є в інфрачервоній системі дистанційного керування?
2. Яка несуча частота використовується в лабораторній роботі?
3. Як приймач розрізняє логічну 1 та 0 в даній роботі?

4. Як в програмі організувати виконання дій при натисканні різних кнопок на пульті дистанційного керування?

5. В якому випадку робот зможе передавати дані інфрачервоним каналом назад у пульт?

## Лабораторна робота №11. Дистанційне керування роботом через Bluetooth

Мета роботи: навчитись використовувати Bluetooth для передачі команд керування роботом.

### Теоретичні відомості

Модуль HM-10 (рис. 11.1) дозволяє передавати дані до пристроїв, оснащених Bluetooth. Він підтримує UART, який є широко використовуваним протоколом для передачі даних.

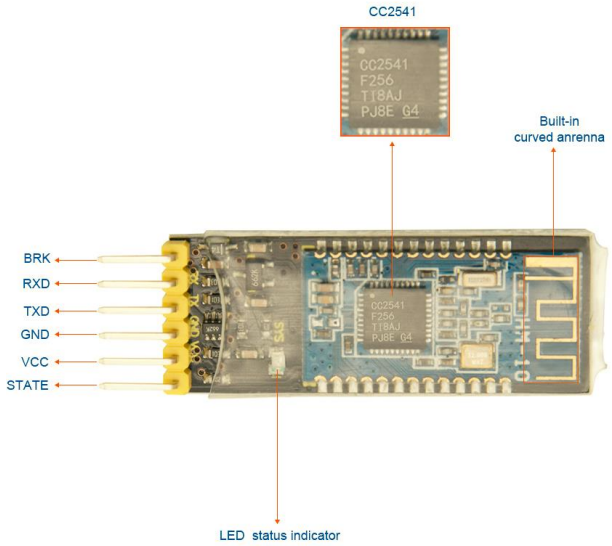



Рис. 11.1. Bluetooth-модуль HM-10

Модуль HM-10 працює в стандарті Bluetooth 4.0, варіант Bluetooth Low Energy. Модуль має два режими роботи: режим зв'язку і режим скидання. У режимі зв’язку модуль активується для встановлення комунікації або обміну даними з іншими пристроями. Під час цього режиму модуль може використовувати різні протоколи чи інтерфейси для передачі інформації. У режимі скидання модуль може повертатися до початкового стану або очікувати на нові команди чи події. Цей режим може бути використаний для відновлення до заводських налаштувань або підготовки до нового циклу роботи.

Модуль зазвичай постачається з налаштуваннями за замовчуванням, але може бути налаштований для відповідності конкретним вимогам, і може взаємодіяти з іншими пристроями через введення команд, що дозволяє налаштовувати його роботу за

допомогою відповідних команд. Заводська прошивка модуля перетворює його в «прозорий» міст UART-BLE.

Для зв'язку робота з BLE-модуля з іншим пристроєм на останньому має бути встановлена програма, що підтримує обмін даними через Bluetooth Low Energy: BLE Scanner для смартфону або BLE Serial для ПК.

**План роботи**

1. Налаштувати передачу команд зі смартфону до робота.
2. Запрограмувати робот для дистанційного керування за командами з іншого пристрою.

**Порядок виконання роботи**

1. Написати програму, яка у відповідь на отримання послідовним інтерфейсом символу (команди) 'L' перемикатиме стан світлодіода (світить-погашений):

```
char ble_val; //character variables, used to save the value of Bluetooth reception
const int led_pin=13;
void setup() {
  Serial.begin(9600);
  pinMode(led_pin, OUTPUT);
}
void loop() {
  if(Serial.available() > 0)  //judge if there is data in buffer area
  { ble_val = Serial.read();  //read the data from serial buffer
    if(ble_val=='L') digitalWrite(led_pin, !digitalRead(led_pin));
  }}
```

Прошити її в плату-контролер робота та перевірити роботу.

2. Підключити модуль HM-10 до контролера (контакти RX, TX, GND, 5V у роз'ємі COM) й надіслати команду через Bluetooth з телефону або комп'ютера. Впевнитись у виконанні команди.

3. Модифікувати програму дистанційного керування з попередньої лабораторної роботи й завантажити у контролер робота:

```
unsigned char start01[] = {0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
unsigned char front[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x12,0x09,0x12,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char back[] = {0x00,0x00,0x00,0x00,0x00,0x24,0x48,0x90,0x48,0x24,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char left[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x44,0x28,0x10,0x44,0x28,0x10,0x44,0x28,0x10,0x00};
```

```
unsigned char right[] = {0x00,0x10,0x28,0x44,0x10,0x28,0x44,0x10,0x28,0x44,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char STOP01[] = {0x2E,0x2A,0x3A,0x00,0x02,0x3E,0x02,0x00,0x3E,0x22,0x3E,0x00,0x3E,0x0A,0x0E,0x00};
unsigned char clear[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
#define SCL_Pin  A5
#define SDA_Pin  A4
#define ML_Ctrl 13
#define ML_PWM 11
#define MR_Ctrl 12
#define MR_PWM 3
char bluetooth_val;
void setup(){
  Serial.begin(9600);
  pinMode(SCL_Pin,OUTPUT);
  pinMode(SDA_Pin,OUTPUT);
  matrix_display(clear);
  matrix_display(start01);
  pinMode(ML_Ctrl, OUTPUT);
  pinMode(ML_PWM, OUTPUT);
  pinMode(MR_Ctrl, OUTPUT);
  pinMode(MR_PWM, OUTPUT);
}

void loop(){
  if (Serial.available())
  {
    bluetooth_val = Serial.read();
    Serial.println(bluetooth_val);
  }
  switch (bluetooth_val)
  {
     case 'F':
        Car_front();
        matrix_display(front);
        break;
     case 'B':
        Car_back();
        matrix_display(back);
        break;
```

```
    case 'L':
      Car_left();
      matrix_display(left);
      break;
    case 'R':
      Car_right();
      matrix_display(right);
     break;
    case 'S':
      Car_Stop();
      matrix_display(STOP01);
      break;
  }
}
void matrix_display(unsigned char matrix_value[])
{
  IIC_start();
  IIC_send(0xc0);
  for(int i = 0;i < 16;i++)
  {
    IIC_send(matrix_value[i]);
  }
  IIC_end();
  IIC_start();
  IIC_send(0x8A);
  IIC_end();
}
void IIC_start()
{
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
}
void IIC_send(unsigned char send_data)
{
  for(char i = 0;i < 8;i++)
  {
     digitalWrite(SCL_Pin,LOW);
     delayMicroseconds(3);
     if(send_data & 0x01)
     {
```

```
      digitalWrite(SDA_Pin,HIGH);
    }
    else
    {
      digitalWrite(SDA_Pin,LOW);
    }
    delayMicroseconds(3);
    digitalWrite(SCL_Pin,HIGH);
    delayMicroseconds(3);
    send_data = send_data >> 1;
  }
}
void IIC_end()
{
  digitalWrite(SCL_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,LOW);
  delayMicroseconds(3);
  digitalWrite(SCL_Pin,HIGH);
  delayMicroseconds(3);
  digitalWrite(SDA_Pin,HIGH);
  delayMicroseconds(3);
}
void Car_front()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,200);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,200);
}
void Car_back()
{
  digitalWrite(MR_Ctrl,HIGH);
  analogWrite(MR_PWM,200);
  digitalWrite(ML_Ctrl,HIGH);
  analogWrite(ML_PWM,200);
}
void Car_left()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,255);
  digitalWrite(ML_Ctrl,HIGH);
  analogWrite(ML_PWM,255);
}
```

```
void Car_right()
{
  digitalWrite(MR_Ctrl,HIGH);
  analogWrite(MR_PWM,255);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,255);
}
void Car_Stop()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,0);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,0);
}
void Car_T_left()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,255);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,180);
}
void Car_T_right()
{
  digitalWrite(MR_Ctrl,LOW);
  analogWrite(MR_PWM,180);
  digitalWrite(ML_Ctrl,LOW);
  analogWrite(ML_PWM,255);
}
```

4. Після перевірки працездатності у попередню програму додати ще одну команду – поворот на кут 120°. Налагодити роботу системи.

5. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. Як підключити модуль HM-10 до контролера робота?

2. Які програми можна використовувати для надсилання даних роботу через Bluetooth.

3. Який модуль мікроконтролера в платі Arduino Uno задіюється для передачі даних у Bluetooth-модуль?

4. Які переваги та недоліки віддаленого керування через Bluetooth порівняно з інфрачервоним?

## Лабораторна робота №12. Програмування робота, що підтримує дистанцію до об’єкта

Мета роботи: навчитись програмувати робота для вимірювання дистанції до об’єкта попереду та регулювання швидкості приводів.

### Теоретичні відомості

Задача підтримування дистанції до транспортного засобу попереду в безпечних межах шляхом зміни швидкості виникає як в пілотованих, так і безпілотних рухомих платформах (рис. 12.1). Для визначення відстані можуть використовуватись радарні, інфрачервоні, ультразвукові датчики відстані.

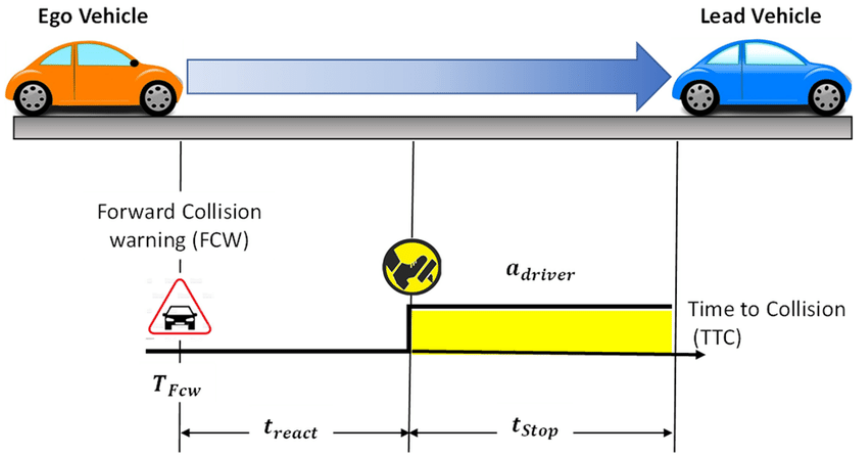



Рис. 12.1. Ілюстрація утримання дистанції в безпечних межах

У лабораторній роботі буде використано ультразвуковий сенсор HC-SR04, що генерує ультразвукову хвилю, яка повертається до нього після відбивання від об'єкта (рис. 12.2). Час, що витрачається хвилею на повернення, рівний відстані до об'єкта, поділеній на швидкість звуку в повітрі.


Рис. 12.2. Принцип вимірювання дистанції датчиком HC-SR04

### План роботи

1. Запрограмувати робота для вимірювання відстані до перешкоди попереду.

2. Запрограмувати зміну швидкості робота залежно від дистанції до об’єкта попереду.

**Порядок виконання роботи**

1. Написати програму для вимірювання відстані й позиціювання сервопривода датчика в напрямку вперед, використовуючи напрацювання з лабораторної роботи 1 або https://wiki.keyestudio.com/Ks0428_keyestudio_Mini_Tank_Robot_V2#Project_12:_Ultrasonic_Follow_Tank.

2. Додати умови зміни швидкості руху в програму керування гусеничним роботом з лабораторної роботи 9 залежно від дистанції, виміряної датчиком, щоб реалізувати позиційний регулятор швидкості.

3. Змінити закон керування швидкістю з позиційного на інший (на вибір), який забезпечуватиме плавну зміну швидкості без різких ривків.

4. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. В чому полягає задача керування дистанцією за транспортним засобом попереду?

2. Які датчики застосовуються для визначення дистанції?

3. Як забезпечувалась зміна швидкості робота з точки зору апаратної частини?

4. Як забезпечувалась зміна швидкості робота з точки зору програмної частини?

5. На що впливає вибір закону керування швидкістю?

## Лабораторна робота №13. Об'їзд перешкод роботом

Мета роботи: навчитись програмувати рух автономного робота з об'їздом перешкод.

### Теоретичні відомості

Поворотний ультразвуковий сенсор робота може використовуватись не лише для вимірювання дистанції до транспортного засобу спереду, а й для вимірювання відстаней до об’єктів у передній півплощині, адже сервопривод в основі сенсора може повертатись на кут 0…180°. Проте якість оцінки обстановки за допомогою ультразвукових датчиків гірша, ніж, наприклад, інфрачервоних лідарів, оскільки ультразвуковий промінь поширюється конусом із значно більшим кутом розходження при вершині, ніж в оптичних лазерних системах. Датчик HC-SR04 характеризується кутом розходження приблизно 15°.

Ще однією проблемою в оцінці відстані до перешкод активними скануючими датчиками є відбивання променя розташованими під гострим кутом гранями перешкод вбік від приймача датчика. Це сприймається датчиком як відсутність відбитого сигналу, а тому робиться висновок при значну відстань до перешкоди, яка знаходиться набагато ближче, але не відбиває хвилі від датчика назад до нього. Відбитий від площини ультразвук вловлюється датчиком HC-SR04 при куті падіння до 30-35°. Це варто враховувати при русі робота й формуванні карти прохідності. Один і той самий об’єкт може бути виявлений або ні залежно від ракурсу, звідки вимірюється відстань.

### План роботи

1. Розробити програму для руху робота з уникненням перешкод з вимірюванням відстані в трьох напрямках.

2. Запрограмувати робота для руху з уникненням перешкод з вимірюванням відстані з кроком 15°.

### Порядок виконання роботи

1. Розробити програму, що періодично повертає сервопривод датчика відстані в положення 10°, 90°, 170° та вимірює відстані. Робот повертається і рушає в тому напрямку, в якому відстань найбільша, і рухається, доки відстань до перешкоди попереду не стане менше 25 см. Тоді робот зупиняється і повторює цикл вимірювання відстані й руху. Підключення компонентів наведено в https://wiki.keyestudio.com/Ks0428_keyestudio_Mini_Tank_Robot_V2#Project_11:_Ultrasonic_Avoid_Tank. Перевірити роботу робота.

2. Модифікувати програму, щоб відстань вимірювалась у всіх досяжних сервоприводом напрямках з кроком 15°, а не лише в трьох. Решта алгоритму залишається незмінною. Перевірити роботу робота.

3. Розробити програму, що прийматиме через Bluetooth заданий

початковий напрямок (кут) руху, здійснюватиме поворот робота і рухатиме його в заданому напрямку. Коли відстань до перешкоди попереду буде менше 25 см, обрати новий напрямок руху аналогічно пункту 2. Після повороту і початку руху в заданому напрямку проїхати 35 см (вести підрахунок тривалості руху в новому напрямку, на базі значення швидкості з лабораторної роботи 9 розраховувати пройдену відстань) і зупинитись (рис. 13.1). Розрахувати відстань (відхилення, зсув) до початкової прямої, якою рухався робот (використати синус кута повороту). Виводити її на дисплей. Базуючись на раніше збереженому куті попереднього повороту, здійснити поворот в бік початкової прямої, розрахувати відстань, яку треба пройти в новому напрямку, щоб вийти на початкову пряму. Почати рух в новому напрямі. Після виходу на початкову пряму скоректувати напрям руху, щоб він відповідав заданому на початку.

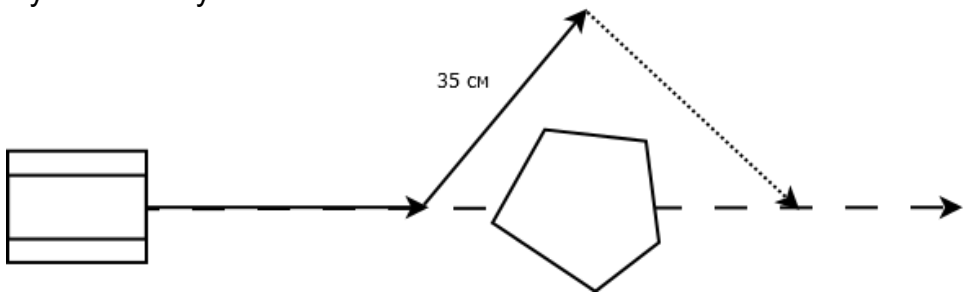



Рис. 13.1. Схема об'їзду перешкоди

4. Передбачити в програмі, що друга перешкода може трапитись, поки не закінчено об'їзд першої (робот не вийшов на початкову пряму). В цьому випадку після зупинки робота при виявленні перешкоди зберігати розрахункову відстань до прямої і використовувати її під час наступних розрахунків відхилення від початкової прямої.

5. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; висновки.

**Контрольні запитання**

1. Які засоби виявлення перешкод роботом ви знаєте?

2. Які переваги та недоліки ультразвукових сканерів у виявленні перешкод?

3. Які переваги та недоліки лідарів у виявленні перешкод?

4. Який кут розходження ультразвукового променя датчика HC-SR04? На що він впливає? До яких ефектів може призвести таке значення кута?

5. Який максимальний кут падіння ультразвукового променя на площину допустимий для успішного вимірювання відстані датчиком HC-SR04?

## Список рекомендованої літератури

1. Ловейкін В. С., Ромасевич Ю. О., Крушельницький В. В. Мехатроніка : підручник. К., 2020. 404 с.

2. Margolis Michael. Arduino Cookbook. O'Reilly Media, 2011. 662 p.

3. Evans B. Arduino programming notebook. First edition. 2007. 38 p. URL: https://playground.arduino.cc/uploads/Main/arduino_notebook_v1-1.pdf.

4. Ks0428 keyestudio Mini Tank Robot V2. Keyestudio Wiki. URL: https://wiki.keyestudio.com/Ks0428_keyestudio_Mini_Tank_Robot_V2