Міністерство освіти і науки України

Національний університет водного господарства та природокористування

Кафедра агроінженерії

02-06-10М

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практичних робіт
з освітньої компоненти **«Робототехніка в машинобудуванні»**
для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Автомобільний транспорт»
спеціальності 274 «Автомобільний транспорт»
всіх форм навчання

Рекомендовано науково-
методичною радою з якості ННМІ
Протокол № 10 від 05 липня 2023 р.

Рівне – 2023

Методичні вказівки до виконання практичних робіт з освітньої компоненти «Робототехніка в машинобудуванні» для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Автомобільний транспорт» спеціальності 274 «Автомобільний транспорт» всіх форм навчання [Електронне видання] / Голотюк М. В., Реут Д. Т., Марчук Н. М. – Рівне : НУВГП, 2023. – 50 с.

Укладачі:

Голотюк М. В. – к.т.н., доцент кафедри агроінженерії;

Реут Д. Т. – к.т.н., доцент кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій;

Марчук Н. М. – к.т.н., доцент кафедри автомобілів та автомобільного господарства.

Відповідальний за випуск: Налобіна О. О., в. о. завідувача кафедри агроінженерії.

Керівник групи забезпечення спеціальності
208 «Агроінженерія» Налобіна О. О.

Методичні вказівки схвалено на засіданні кафедри агроінженерії Протокол № 1 від 04 липня 2023 року.

Схвалено науково-методичною радою з якості ННМІ
Протокол № 10 від 05 липня 2023 року.

ЗАГАЛЬНІ ПОЛОЖЕННЯ

Дисципліна «Робототехніка в машинобудуванні» є невід'ємним складником формування професійної компетентності студентів. Програма дисципліни передбачає комплексне вивчення робототехніки в машинобудуванні.

Робототехніка в машинобудуванні відносно самостійна дисципліна, яка дає загальне уявлення про розвиток техніки. Даний курс покликаний сприяти формуванню у студентів технічних спеціальностей загальної картини розвитку інженерної справи як цілісного процесу, який відбувається закономірно і проходить в органічному взаємозв'язку і взаємодії з історією суспільства.

Курс «Робототехніка в машинобудуванні» дозволить отримати знання в обсязі, достатньому для самостійного вирішення конструкторських та виробничо-технологічних завдань в галузі конструювання, проектування та сервісного обслуговування робототехнічних систем та комплексів, призначених для автоматизації виробничих (технологічних) процесів. Отримані вміння дозволяють використовувати інженерні методики, аналітичні та числові методи розрахунку для аналізу відомих та розробки нових механізмів, вузлів та комплексів обладнання робототехнічних систем.

Мета навчальної дисципліни «Робототехніка в машинобудуванні» є надання студентам знань і навиків в області розвитку та використання роботів і роботизованих технологічних комплексів (РТК), ознайомлення з методикою вибору та проектування РТК для вирішення технологічних задач виробництва..

У результаті вивчення навчальної дисципліни «Робототехніка в машинобудуванні» студент повинен *знати:* типові та сучасні конструкції промислових роботів, принципи їх функціонування та сфери використання, переваги і недоліки; інженерні методики розрахунку та особливості проектування промислових роботів, методи розрахунку їх основних вузлів і механізмів на міцність, жорсткість, стійкість та довговічність; методи кінематичних і динамічних розрахунків, комп'ютерного моделювання та аналізу механізмів робота.

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ВИКОНАННЯ ПРАКТИЧНИХ ЗАНЯТЬ

Практична робота 1. Основи програмування в середовищі Arduino IDE. Вивчення роботи з вхідними та вихідними дискретними сигналами

Мета роботи: ознайомитися з апаратною та програмною частинами середовища Arduino. Вивчити основи роботи з аналоговими і дискретними сигналами в Arduino IDE.

Теоретичні відомості

Arduino – це відкрита електронна платформа, що базується на простому у використанні апаратному та програмному забезпеченні.

Апаратна частина включає мікроконтролерну 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 пристрій на основі мікроконтролера ATmega328P. До його складу входить все необхідне для зручної роботи з мікроконтролером: 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 пристроєм 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. Максимальний струм, який може віддавати або споживати один вивід, становить 40 мА. Усі виводи з'єднані з внутрішніми підтягуючими резисторами (за замовчуванням відключеними) номіналом 20-50 кОм. Крім цього, деякі виводи Arduino Uno можуть виконувати додаткові функції:

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

Зовнішні переривання: виводи 2 і 3. Можуть служити джерелами переривань, що виникають при фронті, спаді або при низькому рівні сигналу на цих висновках.

ШІМ: виводи 3, 5, 6, 9, 10 і 11. За допомогою функції analogWrite() можуть виводити 8-бітові аналогові 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, приєднаний до виводу 13.

В Arduino Uno є 6 аналогових входів (A0 - A5), 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().

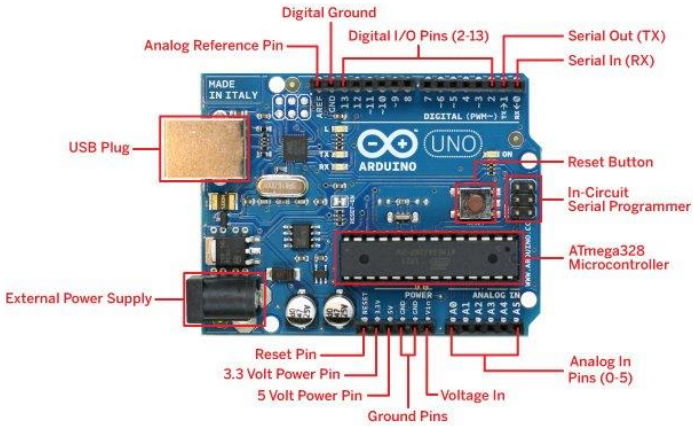


Рис. 1.1.Зовнішній вигляд та призначення виводів плати Arduino Uno

Таблиця 1.1

Основні характеристики Arduino Uno

Мікроконтролер	ATmega328
Робоча напруга	5V
Вхідна напруга (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 (ATmega328) з яких 0.5 KB використовуються завантажувачем
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Тактова частота	16 MHz

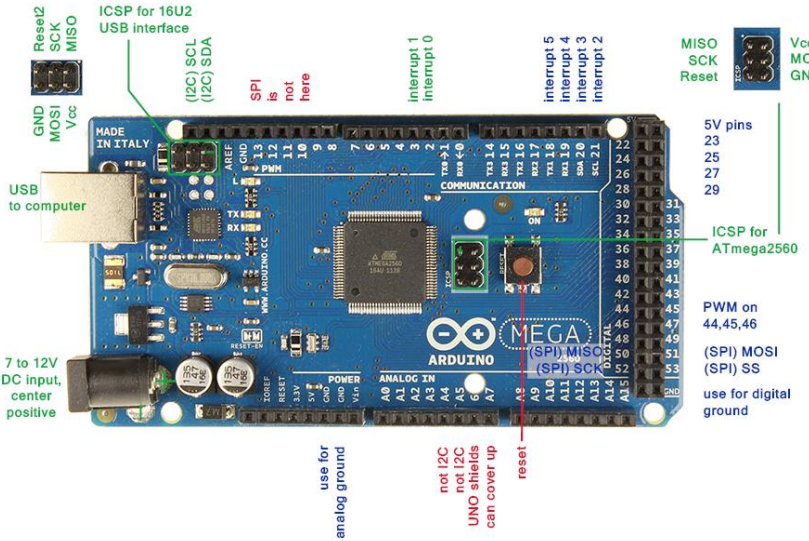


Рис. 1.2. Зовнішній вигляд та призначення виводів плати Arduino Mega 2560

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

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. Взаємодія з ним здійснюється за оригінальним протоколом STK500.

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.

План роботи

1. Ознайомитися з будовою та призначенням елементів плати Arduino Uno.

2. Ознайомитися з принципами програмування в середовищі Arduino IDE.

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

Порядок виконання роботи

1. Ознайомитися з теоретичними відомостями.

2. Завантажити середовище Arduino IDE.

3. Ознайомитись зі схемою підключення механічної кнопки (рис. 1.3.). Зібрати її на макетній платі. Використати підтягуючий резистор на 7,5 кОм.

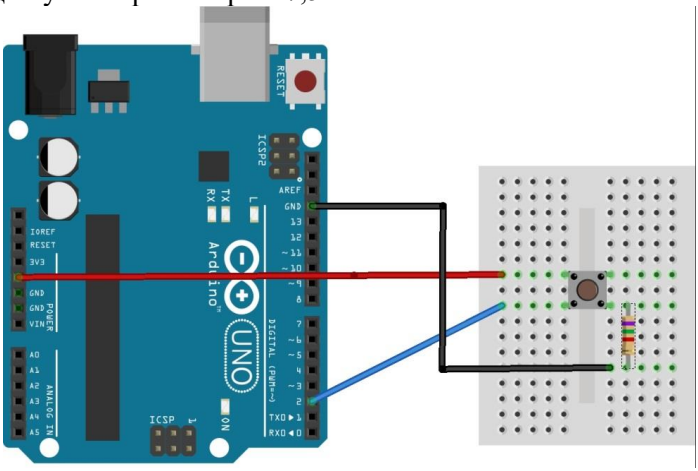


Рис. 1.3. Підключення кнопки до Arduino Uno

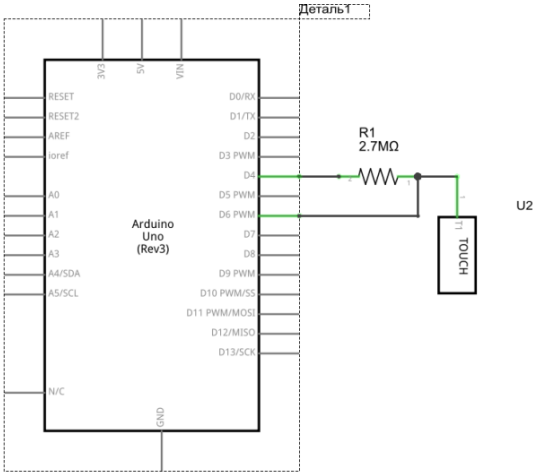
У середовищі Adruino IDE написати програму, що запалюватиме світлодіод при кожному натисканні кнопки.

Скомпілювати програму та переконатися у відсутності помилок.

Підключити мікроконтролерну плату до USB-порта комп'ютера.

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

4. Ознайомитися зі схемою підключення ємнісної сенсорної кнопки. Зібрати її на макетній платі. Підключити сенсорну кнопку до 4 (send pin) і 6 (receive pin) дискретних виводів Arduino. Використати резистор 2,5 МОм.



Завантажити бібліотеку Capacitive Sensing http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense та інсталювати її. Написати програму, що запалюватиме світлодіод при кожному натисканні кнопки.

```
   #include <CapacitiveSensor.h>
   CapacitiveSensor  key1 = CapacitiveSensor(4, 6);
// підключена сенсорна кнопка між 4-м і 6-м пінами
   int range = 100; // порогове значення, що
визначає натискання кнопки. Підбирається
експериментально
   byte button; // побітове позначення натиснутих
кнопок
   void setup()
   {
   pinMode(13,OUTPUT);
   }
   void loop()
   {
    long total = key1.capacitiveSensor(30);
```

```
 button = 0;
   if (total > range) {button=1;}
if (button==1) digitalWrite(13, HIGH);
    else digitalWrite(13, LOW);
}
```

Завантажити програму в мікроконтролер.

Поекспериментувати з чутливістю спрацювання кнопки.

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

Контрольні запитання

1. Яка функція налаштовує режим роботи піна на вхід?
2. Яка функція дозволяє зчитати стан цифрового входу?
3. Яка функція дозволяє змінити стан цифрового виходу?

Практична робота 2. Організація зчитування сигналів з давачів

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

Теоретичні відомості

Мікроконтролер 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 захвату 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, скважності і т.д.

Приклад 1. Вимірювання періоду дискретного сигналу (рис. 2.5)

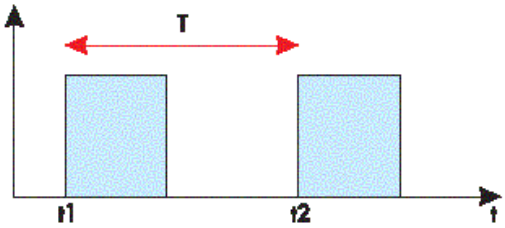


Рис. 2.5. Вимірювання періоду

Приклад 2. Вимірювання періоду з усередненням результату (рис. 2.6)

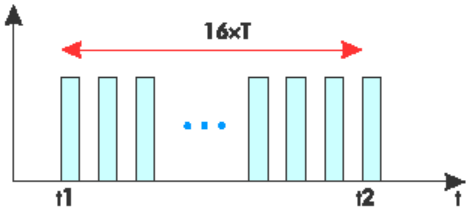


Рис. 2.6. Вимірювання періоду з усередненням

Приклад 3. Вимірювання тривалості імпульсу (рис.2.7)

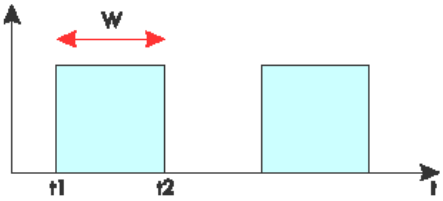


Рис. 2.7. Вимірювання тривалості імпульсу

Приклад 4. Вимірювання скважності імпульсів (рис. 2.8)

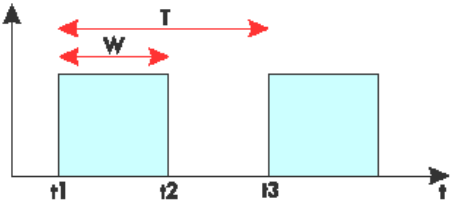


Рис. 2.8. Вимірювання скважності імпульсів

Аналогово-цифровий перетворювач (АЦП, англ. Analog-todigital conventer, ADC) - пристрій, що перетворює вхідний аналоговий сигнал в цифровий код (цифровий сигнал).

У мікроконтролері ATmega328P є 10-розрядний АЦП. Вхідна напруга конвертується в 10-розрядне двійкове значення. Мінімальне значення відповідає 0, а максимальне - опорній напрузі. 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

$$ADC = \frac{V_{in} \cdot 1024}{V_{ref}}$$

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.

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.

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 МК наступну програму:

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

int maximumRange = 200; // максимальна відствнь
int minimumRange = 0; // Minimum range needed
float duration, distance; // Duration used to calculate distance

void setup() {
 Serial.begin (9600);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
 pinMode(LEDPin, OUTPUT); // Use LED indicator (if required)
}

void loop() {
/* The following trigPin/echoPin cycle is used to determine the distance of the nearest object by bouncing soundwaves off of it. */
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);

 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);

 digitalWrite(trigPin, LOW);
 duration = pulseIn(echoPin, HIGH);

 //Calculate the distance (in cm) based on the speed of sound.
 distance = duration/58.2;
```

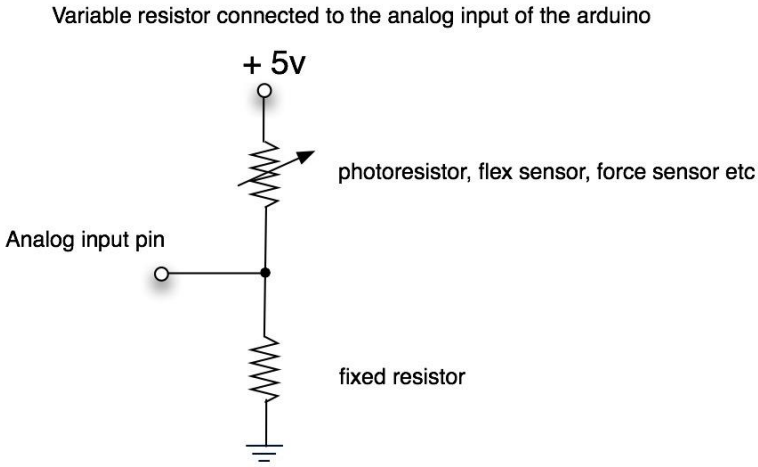
```
   if (distance >= maximumRange || distance <=
minimumRange){
   /* Send a negative number to computer and
Turn LED ON to indicate "out of range" */
   Serial.println("error");
   Serial.print("\n\r");
   digitalWrite(LEDPin, HIGH);
   delay(50);
   }
   else {
   /* Send the distance to the computer using
Serial protocol, and
   turn LED OFF to indicate successful
reading. */
   Serial.println(distance);
   Serial.print("\n\r");
   digitalWrite(LEDPin, LOW);
   delay(250);
   }

   //Delay 50ms before next reading.
   delay(150);
  byte l[255];
    byte count=0;
    int i;
    if (Serial.available()>0){
    while(Serial.available()>0){
      digitalWrite(LEDPin, HIGH);
     l[count] = Serial.read();
     count++;}
    for(i=0;i<count;i++){
      Serial.write(l[i]);};
      Serial.println();
      Serial.print("\n\r");
      delay(500);}
  }
```

Відкрити монітор порту. Піднести руку навпроти давача

на відстані 20 – 50 см. Спостерігати за виміряним значенням відстані, що виводиться на екран.

2. Підключити підстроювальний або змінний резистор за схемою подільника напруги (потенціометра).



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.

```
  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 лабораторної роботи. Звіт повинен містити: назву та мету лабораторної 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?

Практична робота 3. 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 смартфонах, автомобільній промисловості для управління подушками безпеки, і в охоронної 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 (рис. 3.1, 3.2).

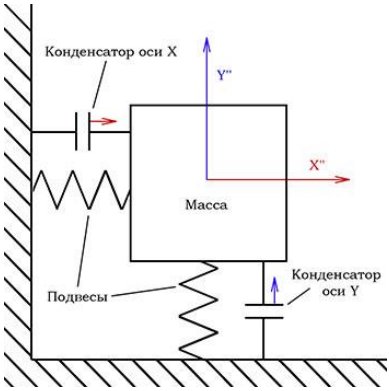


Рис. 3.1 Принцип дії ємнісного акселерометра

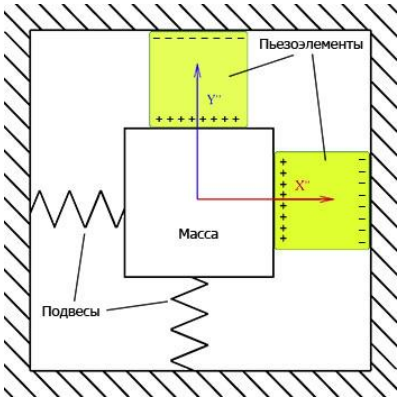


Рис. 3.2. Принцип дії п'єзоелектричного акселерометра

Модуль акселерометра-гіроскопа GY-521

Рис. 3.3. Зовнішній вигляд модуля акселерометра-гіроскопа GY-521

Модуль GY-521 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.

Необхідну напругу живлення для модуля MPU6050 регулює вбудований стабіллізатор 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 °/с
- Діапазон вимірювань акселерометра: ± 2, 4, 8, 16 g
- Крок між контактами: 2.54 мм
- Вбудований 16-бітний АЦП
- Розмір плати: 20 мм х 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);
        }

        // If you want, you can set accelerometer
offsets
        // mpu.setAccelOffsetX();
        // mpu.setAccelOffsetY();
        // mpu.setAccelOffsetZ();

        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. 3.4.

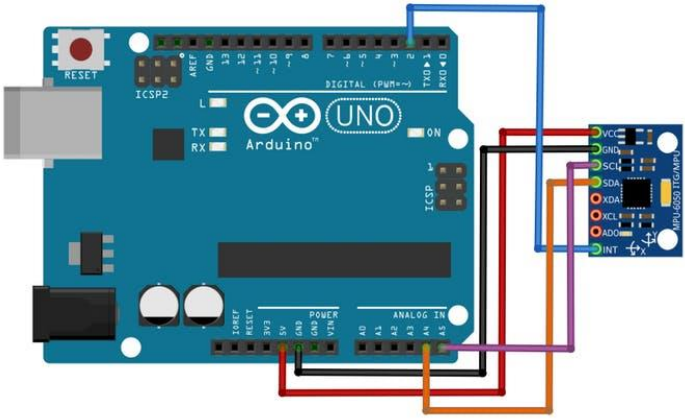


Рис. 3.4. Підключення модуля акселерометра до Arduino Uno

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?

Практична робота 4. Реалізація програмного керування двигуном 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. 4.1.

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).

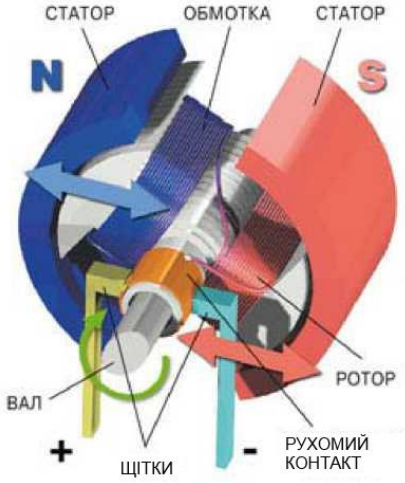


Рис. 4.1. Будова колекторного ДПС з двополюсним статором і двополюсним ротором

Принцип ШІМ пояснюється на рис. 4.2. При ШІМ вихідний сигнал являє собою імпульсний сигнал постійної частоти і змінної скважності. Скважність – це відношення періоду проходження імпульсів T до їх тривалості τ:

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

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.

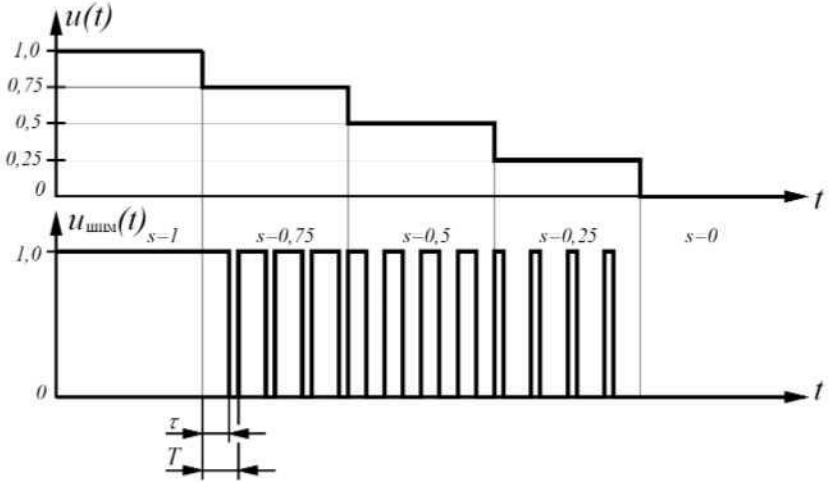


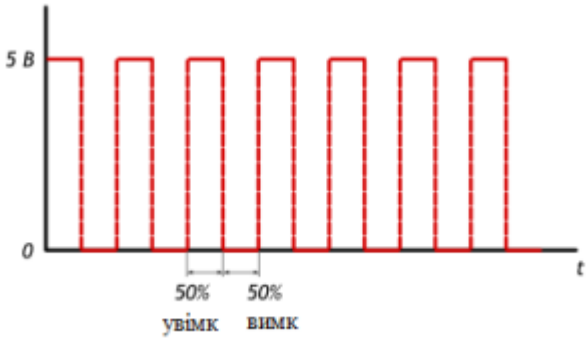
Рис. 4.2. Широтно-імпульсна модуляція

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 інший – модулюючий безперервний аналоговий сигнал. Частота вихідних імпульсів ШІМ відповідає частоті «зубів» пилки. В ту частину періоду, коли сигнал на позитивному вході вище сигналу на негативному вході, на виході виходить одиниця, в іншу, коли сигнал на позитивному вході нижче сигналу на негативному вході - нуль.

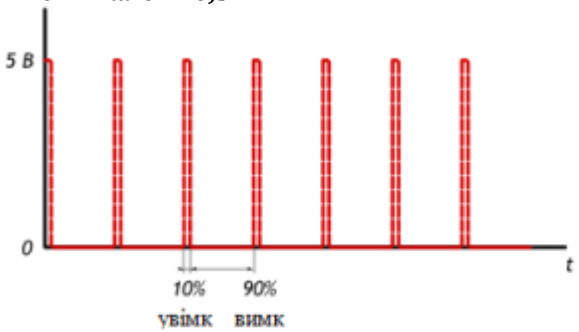
Цифрова ШІМ. У двійковій цифровій техніці, виходи в якій можуть приймати тільки одне з двох значень, наближення бажаного середнього рівня вихідного сигналу за допомогою ШІМ є абсолютно природним. Схема така ж проста: пилкоподібний сигнал генерується *N*-бітовим лічильником.

Розглянемо кілька прикладів при напрузі живлення Vcc рівній 5 вольтам.

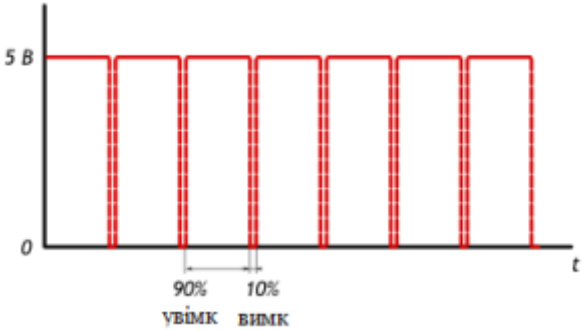
50% — еквівалент 2,5 В



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.

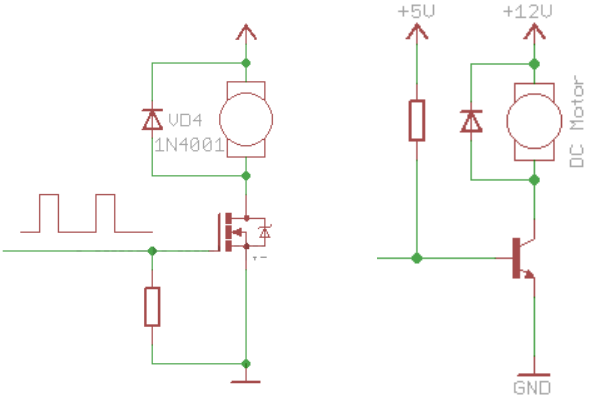
Функція analogWrite() не має нічого спільного з аналоговими входами і функцією analogRead(). Синтаксис:

```
analogWrite(pin, value)
```

Параметри

`pin`: номер ШІМ-виходу

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



а) б)

Рис. 4.3. Схема керування швидкістю обертів ДПС з використанням: а) польового транзистора; б) біполярного транзистора і діода

Найпростіша схема керування двигуном постійного

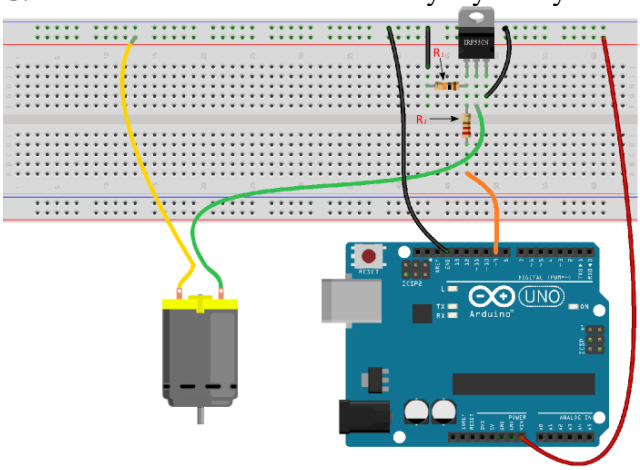
струму складається з польового транзистора, на затвор якого подається ШІМ сигнал. Транзистор в даній схемі виконує роль електронного ключа, який комутує один з виводів двигуна на землю. Транзистор відкривається на момент тривалості імпульсу.

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



Резистор R1 підтягує до "землі" затвор транзистора. Номінал не принциповий - можна використовувати будь-які резистори в діапазоні від 1 до 10 кОм. Резистор R2 служить для захисту виводу мікроконтролера. Діапазон, приблизно, від 10 до 500 Ом.

4. В середовищі Adruino IDE створити нову програму.

```
int led = 9; // ШІМ-вивід, до якого підключений затвор транзистора
int speed = 0; // ця змінна відповідає за швидкість обертання
int fadeAmount = 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   +   fadeAmount;   //   Коли
швидкість стає максимальною / мінімальною -
починаємо її знижувати / підвищувати
      if (speed == 0 || speed == 255)
      {
      fadeAmount = -fadeAmount;
      }
      // 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?

Практична 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 найчастіше розуміють механізм з електродвигуном, який можна попросити повернутися в заданий кут і утримувати це положення. Однак, це не зовсім повне визначення. Якщо сказати повніше, сервопривід - це привід 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 заданому завданню. Тобто сервопривод отримує на вхід значення керуючого параметра, наприклад, кут повороту. Блок управління порівнює це значення зі значенням на своєму датчику. 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. Найбільш поширені сервоприводи, які утримують заданий кут і сервоприводи, що підтримують задану швидкість обертання.

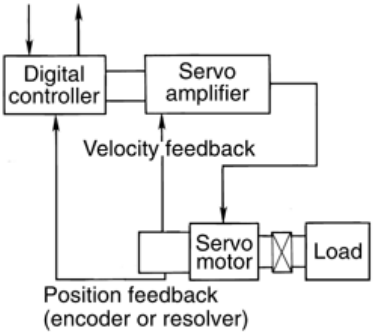


Рис. 5.1. Структурна схема сервопривода з давачем швидкості обертів

Рис. 5.2. Сервоприводи

Отже, сервопривод – це регульований редукторний електродвигун. Він зазвичай складається з приводного механізму з двигуном постійного струму, плати управління і потенціометра, котрий забезпечує зворотний зв'язок.

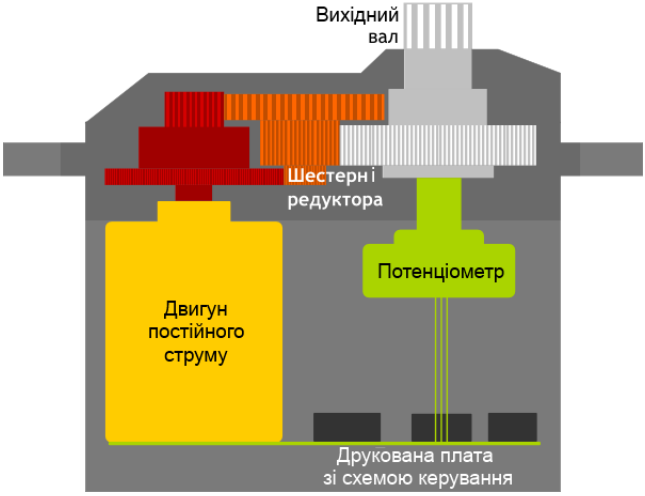


Рис. 5.3. Будова сервопривода

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. Коли сигнал

надходить в керуючу схему, наявний у ній генератор імпульсів виробляє свій імпульс, тривалість якого визначається через потенціометр. Інша частина схеми порівнює тривалість двох імпульсів. Якщо тривалість різна, включається електродвигун. Напрямок обертання визначається тим, який з імпульсів коротший. Якщо довжини імпульсів рівні, електродвигун зупиняється. Найчастіше в аматорських сервоприводів імпульси виробляються 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.

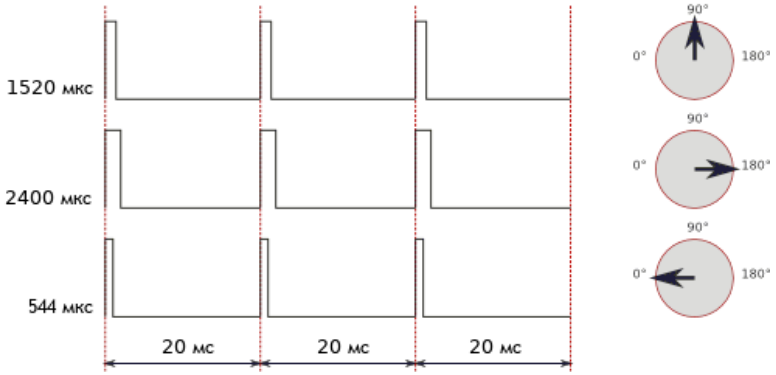


Рис. 5.4. Залежність кута повороту сервопривода від трвалості керуючих імпульсів

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 стандартних. Також варто відзначити, що це всього лише загальноприйняті довжини. 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 і на зниженій потужності, а при сильному завищенні частоти може перегрітися і вийти з ладу.

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

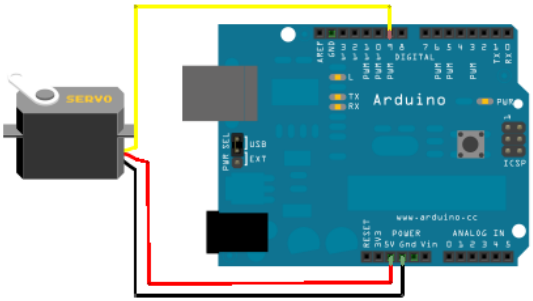


Рис. 5.5. Підключення сервопривода малої потужності до мікроконтролерної плати Arduino Uno

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

attach() – приєднує змінну до конкретного виводу плати. Можливі два варіанти синтаксису для цієї функції: 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.

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 була приєднана змінна до конкретного виводу. Синтаксис наступний: servo.attached(), повертається true, якщо змінна була приєднана до якого-небудь виводу, або false в зворотному випадку.

detach() - виконує дію, зворотну дії attach(), тобто від'єднує змінну від виводу, до якого вона була приписана. Синтаксис наступний: servo.detach().

Приклад програми з використанням бібліотеки Servo

```
#include <Servo.h>
Servo myservo;
void setup()
{
  myservo.attach(9);
}
void loop()
{
  myservo.write(90);//           встановлюємо
сервопривод в середнє положення
  delay(500);
  myservo.write(0);    //    встановлюємо
сервопривод в крайнє лвве положення
  delay(500);
  myservo.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.

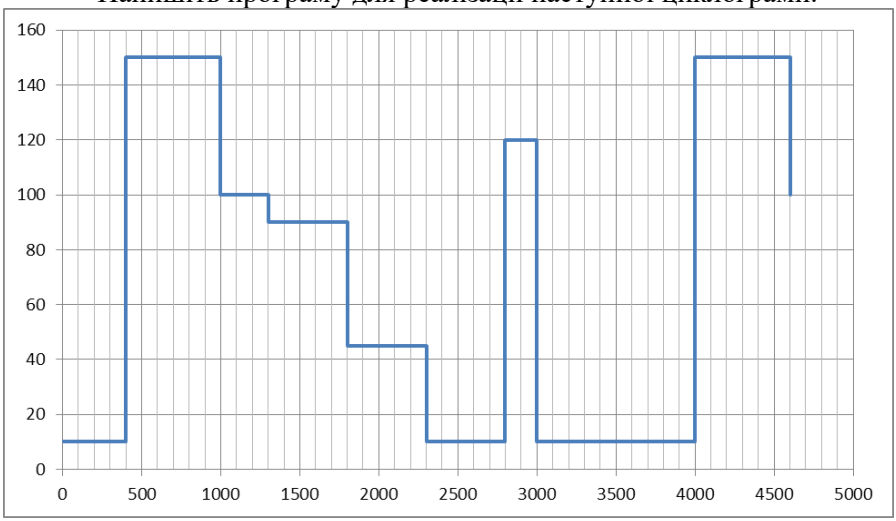
3. Програмно обмежити швидкість повороту вала сервопривода.

Порядок виконання роботи

1. Складіть схему за рис. 5.5, використовуючи сервопривод SG90.

2. Завантажте програму з прикладу бібліотеки Servo.

Напишіть програму для реалізації наступної циклограми.



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

3. Для того, щоб мати змогу змінювати швидкість обертання використайте бібліотеку VarSpeedServo. Завантажити в МК приклад VarSpeedServo, змінюючи значення швидкості. Переконатися в правильності роботи програми та точності позиціювання сервопривода.

4. 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. Що таке сервопривод?

2. Назвіть основні елементи сервопривода.

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

4. Яка бібліотека дає змогу обмежувати швидкість повороту сервопривода?

Практична робота 6. Дослідження роботи маніпулятора з дистанційним управлінням

Мета роботи: Ознайомитися з принципом управління роботом-маніпулятором з 6-ма ступенями свободи. Навчитися створювати програми керування для дистанційного управління маніпулятором.

Теоретичні відомості

У процесі автоматизації промислових підприємств важливим є 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 технологічних операцій, певною мірою зменшити вплив шкідливих факторів на персонал, скоротити площу виробничих приміщень і 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 промислового виробництва. Гнучкі автоматизовані виробництва, які створюються на базі промислових роботів, дозволяють вирішувати задачі автоматизації на підприємствах із широкою номенклатурою продукції при дрібносерійному і штучному виробництві.

Необхідність дослідження та вдосконалення систем управління маніпуляційними роботами насамперед зумовлена їх широким застосуванням. Подібні пристрої використовуються в будівельній галузі (крани-маніпулятори), металургії (прокатні стани, кувальні маніпулятори), гірничодобувній промисловості (бурильні машини), хімічній промисловості (маніпулятори для роботи з токсичними і радіоактивними матеріалами), суднобудівній та авіаційній галузях (зварювальні та складальні маніпулятори, маніпулятори для різання металів) тощо.

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).

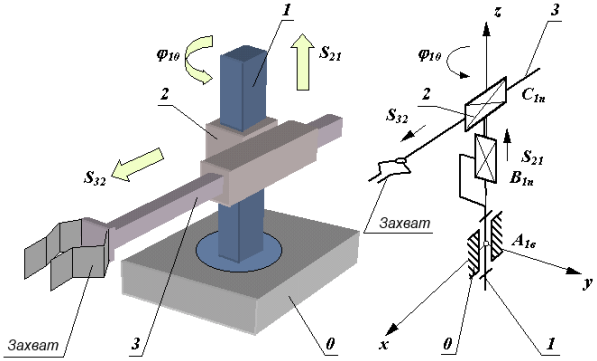


Рис. 6.1. Структурна і функціональна схеми промислового маніпулятора

Механізм цього маніпулятора відповідає циліндричній системі координат. У цій системі ланка *1* може обертатися відносно ланки *0* (відносне кутове переміщення φ_{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 Δx_i, що надходять до неї з давачів зворотного зв'язку (рис. 6.2).

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. Кроковим 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 складається з двох етапів:

- 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).

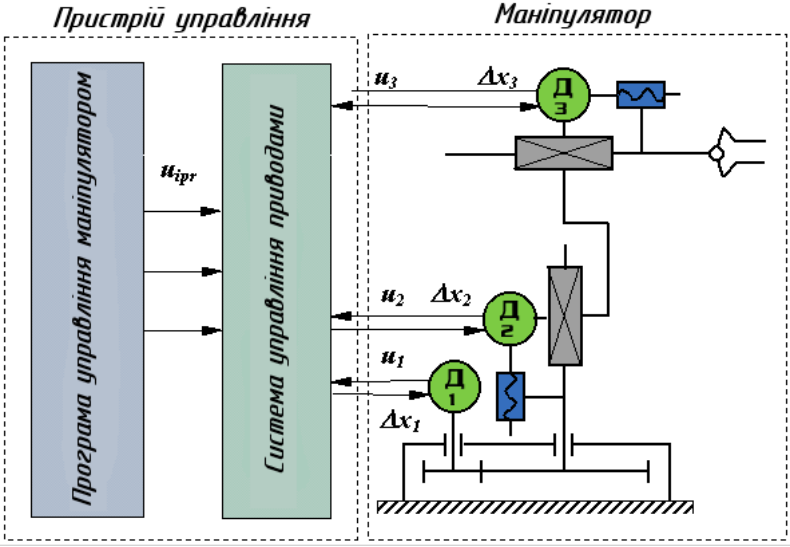


Рис. 6.2. Функціональна схема промислового робота

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 (Δr_M<1 мм), середньою (0,1 мм < Δr_M< 1 мм) і високою (Δr_M< 0,1 мм) точністю позиціювання.

Опис лабораторної установки

Досліджуваний маніпулятор, зображений на рис. 6.3, має 6 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.

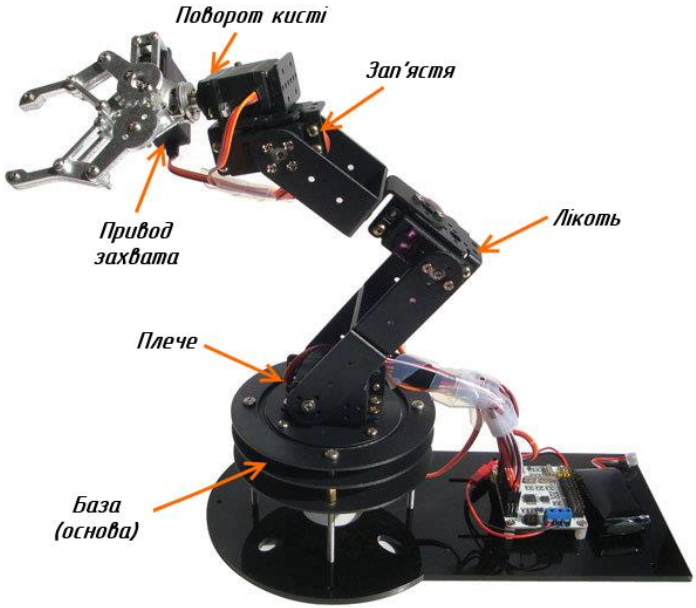


Рис. 6.3. Зовнішній вигляд досліджуваного маніпулятора

Параметри сервоприводів MG996R подані нижче.

Рис. 6.4. Зовнішній вигляд та розміри сервопривода MG996R

Основні характеристики:

З'єднувальний провід: довжина 300мм.

Розміри: 40.7 мм х 19,7 мм х 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.

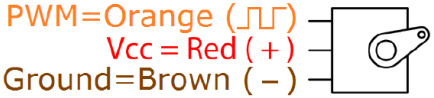
Робоча напруга: 4.8 – 7.2V

Редуктор : металеві шестерні.

Робочий струм при русі: 500 – 900 мА (6 В).

Струм при стабільному навантаженні: 2.5 А (6 В).

Підключення:



Для живлення сервоприводів у лабораторній 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 лабораторної 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 myservo1, myservo2, myservo3, myservo4, myservo5, myservo6;   // 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);

        myservo1.attach(2,740,2365);   //   attaches
the servo on pin 2 to the servo object
        myservo2.attach(3,760,2365);   //  attaches
the servo on pin 3 to the servo object
        myservo3.attach(4,720,2365);   //  attaches
the servo on pin 4 to the servo object
        myservo4.attach(5,720,2300);   //  attaches
the servo on pin 5 to the servo object
        myservo5.attach(10,670,2300);   // attaches
the servo on pin 10 to the servo object
        myservo6.attach(11,720,2300);   // attaches
the servo on pin 11 to the servo object
      oa1 = myservo1.read();
      oa2 = myservo2.read();
      oa3 = myservo3.read();
      oa4 = myservo4.read();
      oa5 = myservo5.read();
      oa6 = myservo6.read();
      }

      void loop()
      {
        if (a1 != oa1){setPosServo(myservo1,a1);oa1
= a1;printserial(); }
        if (a2 != oa2){setPosServo(myservo2,a2);oa2
= a2;printserial(); }
        if (a3 != oa3){setPosServo(myservo3,a3);oa3
= a3;printserial(); }
        if (a4 != oa4){setPosServo(myservo4,a4);oa4
= a4;printserial(); }
        if (a5 != oa5){setPosServo(myservo5,a5);oa5
= a5;printserial(); }
        if (a6 != oa6){setPosServo(myservo6,a6);oa6
= a6;printserial(); }
        if   (serialflag   ==   1)   {printserial();
```

```
serialflag = 0;}
        //printserial(); delay(200);
      }

      void  setPosServo(VarSpeedServo  myservo,int
angle){
        int i = 1;
        int oldangle = myservo.read();
        if (oldangle > angle) {i=-1;}
        for (int a = oldangle; a!= angle; a += i ){
          myservo.slowmove(a,15);     // tell servo
to go to position in variable 'pos'
          delay(15);
        }
      }
      void serialEvent() {
        byte myservo;
        while (Serial.available()) {
          // get the new byte:
          byte inByte = (byte)Serial.read();
          if ((inByte >= 201) && (inByte<=206)  ) {
          myservo = inByte - 200;
          }
          if (inByte <= 180) {
            if (myservo == 1) {a1 = inByte;}
            if (myservo == 2) {a2 = inByte;}
            if (myservo == 3) {a3 = inByte;}
            if (myservo == 4) {a4 = inByte;}
            if (myservo == 5) {a5 = inByte;}
            if (myservo == 6) {a6 = inByte;}
            serialflag = 1;
            myservo = 0;
          }
        }
      }

      void printserial() {
        delay(20);
        Serial.print(myservo1.read()); //  Movement
Data
        Serial.print(",");
        Serial.print(myservo2.read()); //  Movement
```

```
Data
        Serial.print(",");
        Serial.print(myservo3.read()); //  Movement
Data
        Serial.print(",");
        Serial.print(myservo4.read()); //  Movement
Data
        Serial.print(",");
        Serial.print(myservo5.read()); //  Movement
Data
        Serial.print(",");
        Serial.print(myservo6.read()); //  Movement
Data
        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. Завантажити середовище Processing. Відкрити скетч Scrollbar1.pde та запустити його на виконання.

8. З дозволу викладача ввімкнути живлення стенда.

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

10. Встановити 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. Вимкніть живлення стенда.

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

13. Зробити висновки. Звіт повинен містити: титульний лист; тему, мету роботи; порядок виконання; створені програми; заповнену таблицю 3.2; 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. Як класифікують промислові роботи за точністю позиціювання?

7. Що таке ступені свободи маніпулятора?

Практична робота 7. Реалізація захисту і блокування роботи маніпулятора при виявленні перешкод

Мета роботи: навчитися програмувати зупинку роботи мехатронної системи у процесі виявлення в робочій зоні тіла людини або іншої перешкоди за допомогою датчиків руху. Навчитися налаштовувати чутливість 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. Різниця напруг з двох елементів опрацьовується мікросхемою BISS0001, в результаті на виході датчика при вияленні 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 лабораторного 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.

3. У програму в попередній роботі додати перевірку сигналу з датчика руху та зупинку переміщення маніпулятора у випадку виявлення руху.

4. Після перевірки програми та правильності підключення датчика викладачем завантажити програму в плату керування маніпулятором та перевірити правильність роботи.

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?

Практична робота 8. Розроблення програмного забезпечення роботизованої ділянки

Мета роботи: ознайомитися з принципами програмно-логічного управління роботизованими ділянками. Написати програму для автоматичного управління маніпулятором.

Теоретичні відомості

У процесі програмування роботизованих ділянок виробництва потрібно враховувати взаємний вплив кожної ланки у ділянці на попередню, наприклад, неможливість здійснення зупинки одного конвеєра без зупинки попереднього конвеєра, з якого в нормальному режимі роботи повинні забиратись об'єкти на конвеєр, який необхідно зупинити.

Приклад автоматизованої виробничої ділянки показано на рис. 8.1. Заготовка, що прибуває у ділянку на вхідному конвеєрі, переміщається у робочу зону верстата. При цьому слід передбачити зупинку конвеєра до того часу, поки маніпулятор не захопить заготовку. Робот виконує спочатку операцію переміщення заготовки на вхід у верстат. Потім відбувається обробка заготовки у верстаті. Після цього заготовка переміщається на вихід верстата. Потім, якщо робот вільний, він переносить оброблену заготовку на вихідний конвеєр виробничої ділянки.

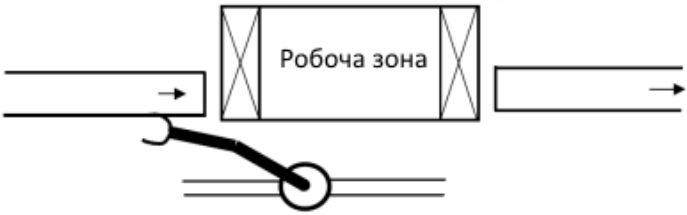


Рис. 8.1. Схема досліджуваної роботизованої ділянки

План роботи

1. Ознайомитися з будовою роботизованої ділянки в лабораторній установці.

2. Написати та відлагодити програму для керування роботизованою ділянкою.

Порядок виконання роботи

1. Ознайомитися з схемою підключення приводів роботизованої ділянки до плати Arduino Mega та силовими елементами для керування приводами.

2. Написати програму для керування маніпулятором роботизованої ділянки.

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

4. З дозволу викладача увімкнути живлення стенда та відлагодити роботу програми.

5. Написати та відлагодити програму для керування конвеєром або рухомою кареткою роботизованої ділянки.

6. На підставі розроблених програм створити програму, що керуватиме роботою всієї ділянки: і робота-маніпулятора, і конвеєра або рухомої каретки.

7. Перевірити правильність роботи програми та відлагодити її.

8. Додати в програму функцію блокування роботи всієї механічної складової при виявленні руху датчиком HC-SR501.

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. У чому відмінність розробки програм керування окремими мехатронними елементами та роботизованими ділянками?

2. Якими способами можна виявити надходження об'єкта на початок ділянки?

3. Які способи керування двигном постійного струму ви знаєте?

4. Чим спричинено використання зворотньо включених діодів параллельно якорям двигунів постійного струму під час керування їх швидкістю за допомогою ШІМ сигналу, що надходить на затвор польового транзистора?

Список рекомендованої літератури

1. Синтез робототехнічних систем в машинобудуванні / Л. Є. Пелевін, К. І. Почка, О. М. Гаркавенко та ін. К. : Інтерсервіс, 2016. 258 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. Методичні вказівки до виконання практичних робіт з навчальної дисципліни «Робототехніка в машинобудуванні» для здобувачів вищої освіти першого (бакалаврського) за освітньо-професійною програмою «Галузеве машинобудування» спеціальності 133 «Галузеве машинобудування» всіх форм навчання [Електронне видання] / Голотюк М. В., Реут Д. Т., Марчук Н. М. Рівне : НУВГП, 2021. 35 с.

5. Ніколайчук В. М. Основи робототехніки : навч. посіб. Рівне : НУВГП, 2008. 76 с. URL: http://ep3.nuwm.edu.ua/2243/.

6. Пелевін Л. Є., Балака М. М., Аржаєв Г. О. Механотронні системи гідропневмоавтоматики. К. : Аграр Медіа Груп, 2014. 192 с.