Міністерство освіти і науки України

Національний університет водного господарства та природокористування

Кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

04-03-362М

МЕТОДИЧНІ ВКАЗІВКИ

до практичних робіт
з освітнього компоненту
«Мехатронні системи техніки в АПК»
для здобувачів вищої освіти другого (магістерського) рівня
за освітньо-професійною програмою «Агроінженерія»
спеціальності
208 «Агроінженерія»
денної форми навчання

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

Рівне – 2023

Методичні вказівки до практичних робіт з освітнього компоненту «Мехатронні системи техніки в АПК» для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Агроінженерія» спеціальності 208 «Агроінженерія» денної форми навчання [Електронне видання] / Реут Д. Т. – Рівне : НУВГП, 2023. – 52 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 208 «Агроінженерія»: Налобіна Олена Олександрівна, д.т.н., професор, професор кафедри будівельних, дорожніх, меліоративних, сільськогосподарських машин і обладнання

Зміст

Вступ	4
Практична робота 1. Основи програмування в середовищі Arduino IDE. Функції для роботи з вхідними та вихідними дискретними сигналами	4
Практична робота 2. Організація зчитування сигналів з датчиків	13
Практична робота 3. Використання акселерометра-гіроскопа	19
Практична робота 4. Дослідження роботи сервоприводів та реалізація циклограми	25
Практична робота 5. Програмування маніпулятора з дистанційним управлінням	31
Практична робота 6. Реалізація захисту і блокування роботи маніпулятора при виявленні перешкод у автоматичному режимі	39
Практична робота 7. Планування та моделювання польоту БПЛА в ArduPilot Mission Planner	46
Список рекомендованої літератури	52

Вступ

Практичні роботи з освітньої компоненти «Мехатронні системи техніки в АПК» дають змогу на практиці вивчити принципи функціонування мехатронних систем техніки, ознайомитись з технічними засобами, які використовуються в складі мехатронних систем, зокрема в АПК.

У практичних 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;

– використання ArduPilot Mission Planner для планування та моделювання польоту БПЛА.

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

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 ºС 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:

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.

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).

Таблиця 1.1

Основні характеристики Arduino Uno

Мікроконтролер	ATmega328P
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 (ATmega328P) з яких 0.5 KB використовуються завантажувачем
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Тактова частота	16 MHz

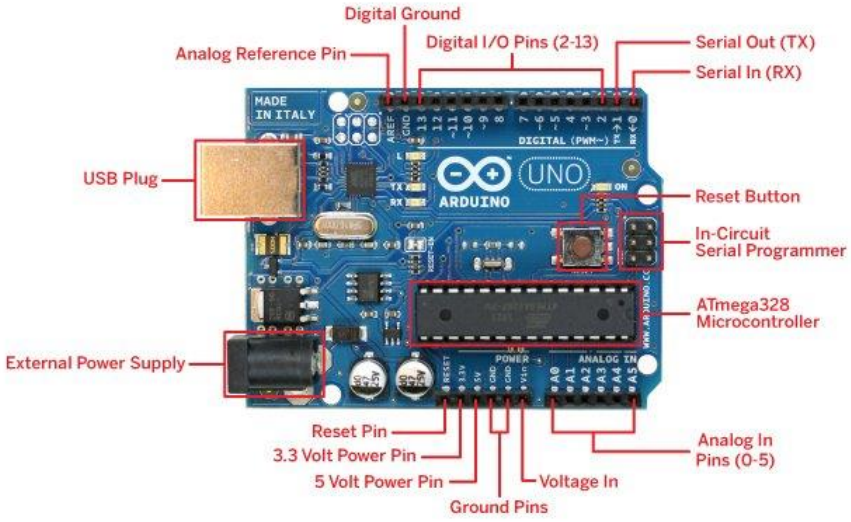


Рис. 1.1. Зовнішній вигляд та призначення виводів плати Arduino Uno

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

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

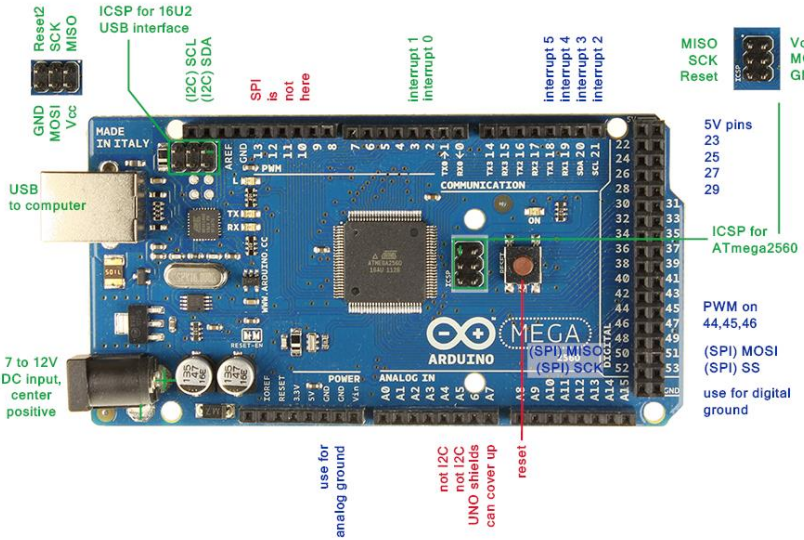


Рис. 1.2. Зовнішній вигляд та призначення виводів плати Arduino Mega 2560

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. 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.

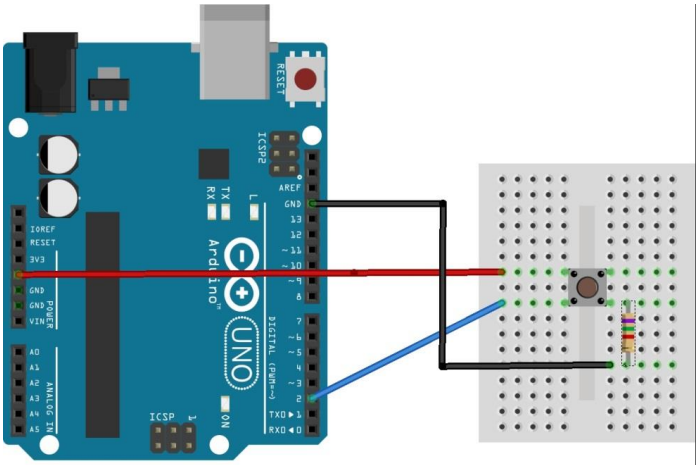


Рис. 1.3. Підключення кнопки до Arduino Uno

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

4. Скомпілювати програму та переконатися у відсутності помилок.

5. 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.

6. Ознайомитися зі схемою підключення ємнісної сенсорної кнопки (рис. 1.4). Зібрати її на макетній платі. Підключити сенсорну кнопку до 4 (send pin) і 6 (receive pin) дискретних виводів Arduino. Використати резистор 2,5 МОм.

7. Завантажити бібліотеку Capacitive Sensing за посиланням http://playground.arduino.cc/Main/CapacitiveSensor?from=Main.CapSense та інсталювати її.

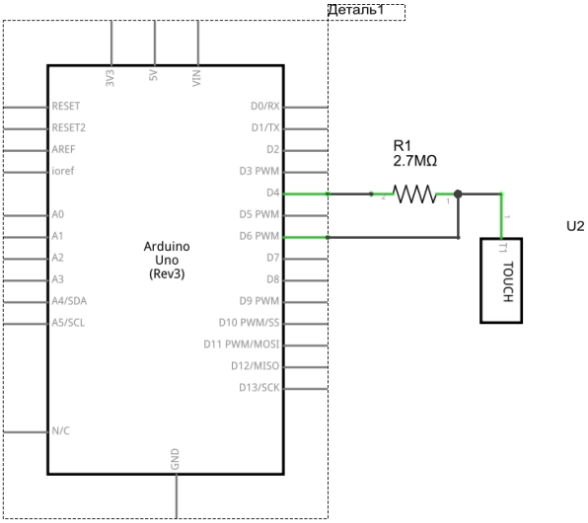


Рис. 1.4. Схема підключення ємнісної сенсорної кнопки

8. 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, змінюючи значення змінної range й оцінюючи силу натиснення або відстань до пальця, за яких спрацьовує кнопка.

9. 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, тривалості паузи, скважності (коефіцієнта заповнення) тощо.

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

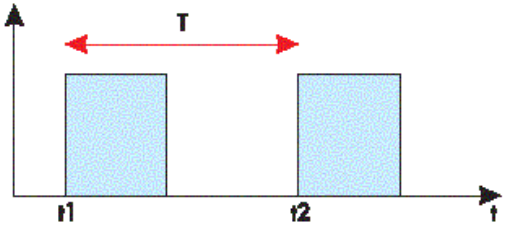


Рис. 2.1. Вимірювання періоду

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

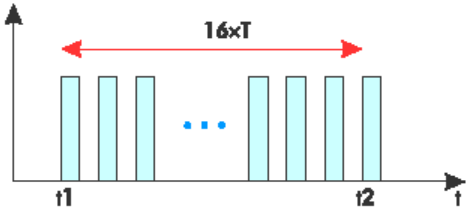


Рис. 2.2. Вимірювання періоду з усередненням

Приклад 3. Вимірювання тривалості імпульсу (рис.2.3)

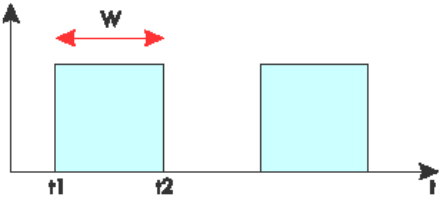


Рис. 2.3. Вимірювання тривалості імпульсу

Приклад 4. Вимірювання скважності імпульсів (рис. 2.4)

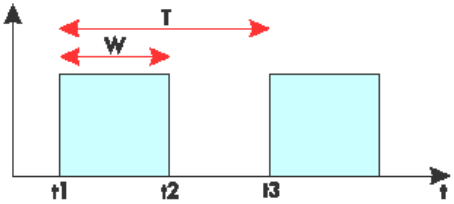


Рис. 2.4. Вимірювання скважності імпульсів

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

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 // 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);
}
```

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.

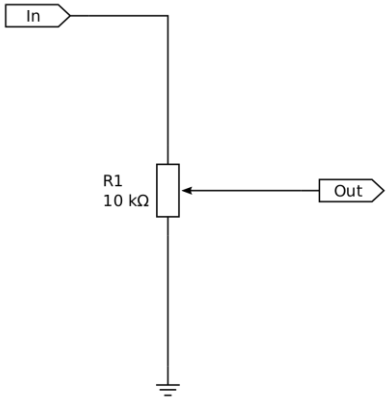


Рис. 2.6. Схема підключення змінного резистора

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 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?

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.

Модуль акселерометра-гіроскопа GY-521

Модуль GY-521 (рис. 3.1) 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.

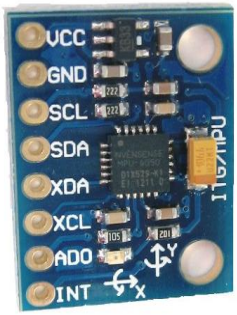


Рис. 3.1. Зовнішній вигляд модуля акселерометра-гіроскопа GY-521

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 °/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. 3.2.

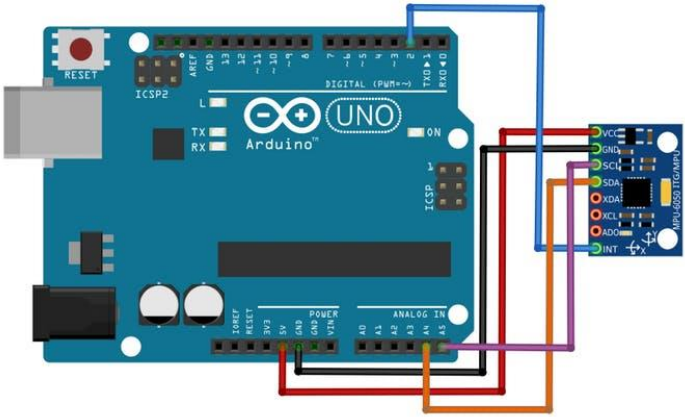


Рис. 3.2. Підключення модуля акселерометра до Arduino Uno

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?

Практична робота 4. 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. 4.1) найчастіше розуміють механізм з електродвигуном, якому можна скомандувати повернутися в заданий кут і утримувати це положення. Однак, це не зовсім повне визначення. Якщо сказати повніше, сервопривід - 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 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.

Рис. 4.1. Сервоприводи

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

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. 4.2).

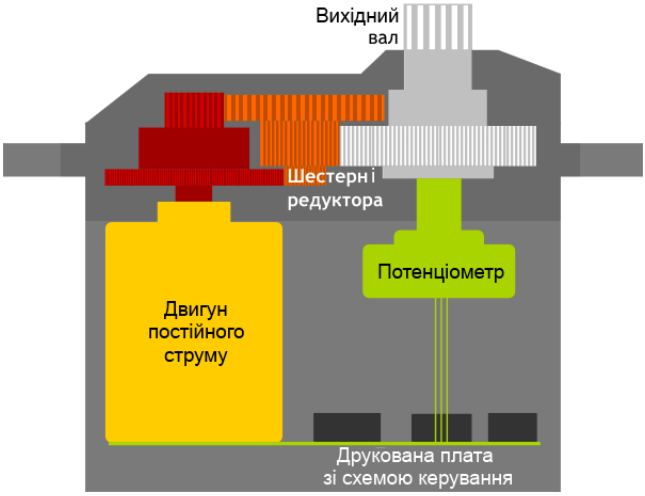


Рис. 4.2. Будова сервопривода

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.

4.3).

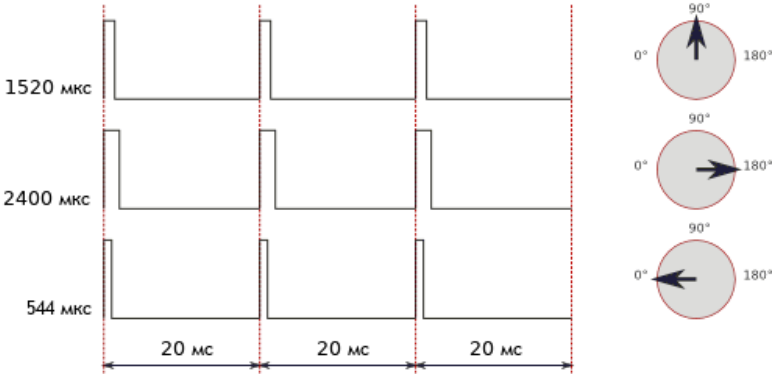


Рис. 4.3. Залежність кута повороту сервопривода від трвалості керуючих імпульсів

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 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 (рис. 4.4), для підключення більш потужних – слід використовувати зовнішнє джерело живлення.

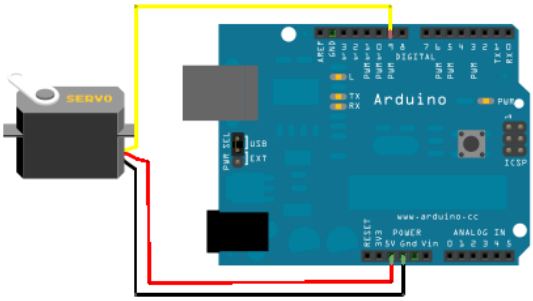


Рис. 4.4. Підключення сервопривода малої потужності до мікроконтролерної плати Arduino Uno

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.

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. 4.4, використовуючи сервопривод SG90.

2. Завантажте програму з прикладу бібліотеки Servo.

Напишіть програму для реалізації наступної циклограми (рис. 4.5).

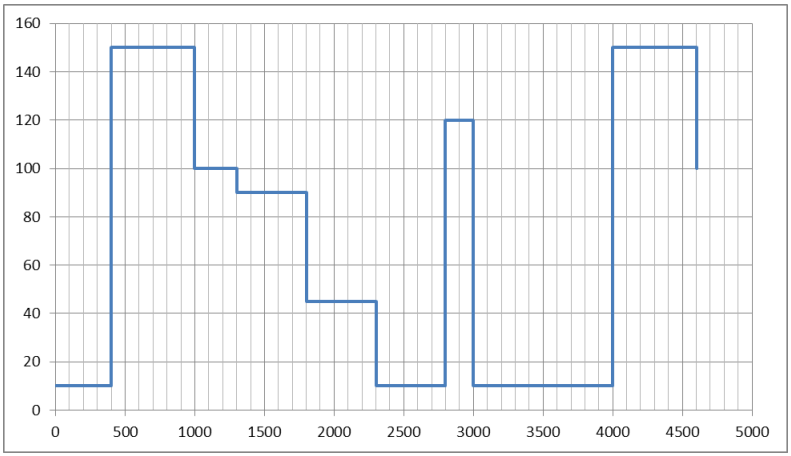


Рис. 4.5. Задана циклограма роботи сервопривода

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

3. Для того, щоб мати змогу змінювати швидкість обертання використайте бібліотеку VarSpeedServo. Завантажити в мікроконтролер приклад 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?

Практична робота 5. Програмування маніпулятора з дистанційним управлінням

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

Теоретичні відомості

У процесі механізації й автоматизації виробничих процесів у АПК 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 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 переробної промисловості АПК.

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. 5.1).

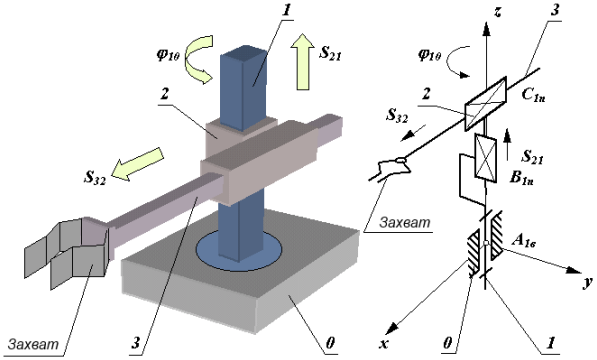


Рис. 5.1. Структурна і функціональна схеми промислового маніпулятора

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. 5.2).

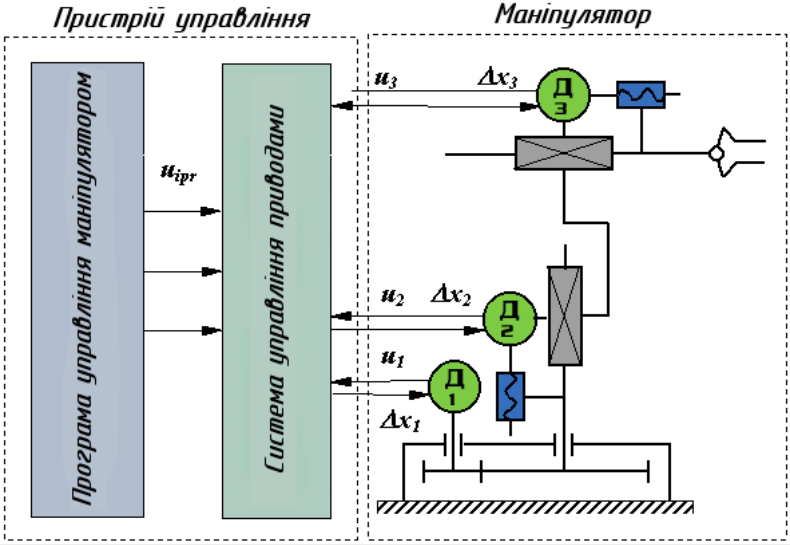


Рис. 5.2. Функціональна схема промислового робота

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.

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. 5.3, 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.

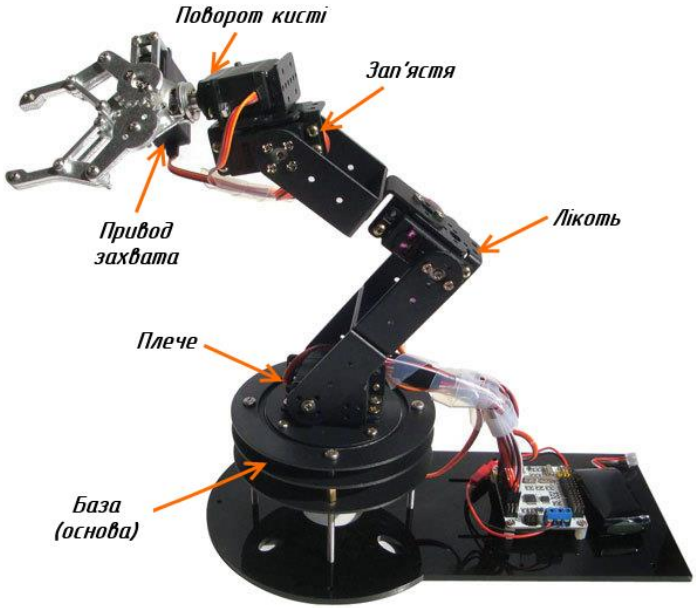


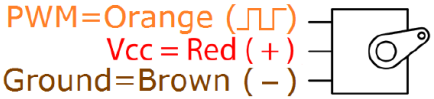
Рис. 5.3. Зовнішній вигляд досліджуваного маніпулятора

Параметри сервоприводів MG996R (рис. 5.4) подані нижче.

Рис. 5.4. 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. 5.1.

Таблиця 5.1

№ 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?

Практична робота 6. 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. 6.1).

Рис. 6.1. 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. 6.2. Призначення контактів датчика руху HC-SR501

Активний рівень сигналу на виході утримується впродовж часу, що задається підстроювальним резистором (рис. 6.2.). 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 программу, яка в автоматичному режимі (не чекаючи даних від ПК) здійснюватиме переміщення маніпулятора між як мінімум трьома 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(interru
ptPin), 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?

Практична робота 7. Планування та моделювання польоту БПЛА в ArduPilot Mission Planner

Теоретичні відомості

ArduPilot Mission Planner використовується для керування та налаштування автономних польотів з використанням автопілотної системи ArduPilot. ArduPilot – це відкрите програмне забезпечення для автопілотів, розроблене спеціально для безпілотників, дронів та інших безпілотних рухомих платформ.

ArduPilot Mission Planner дозволяє планувати місії, встановлювати параметри автопілота, керувати БПЛА та візуалізувати дані польоту. Ця програма забезпечує повний

спектр функцій для керування автономними польотами, зокрема налаштування шляхів, керування точками маршруту, планування місій з додатковими параметрами (наприклад, часом, висотою і т. д.), режими автопілота та інші функції. Підтримується керування додатковим обладнанням, наприклад камерою (збільшення, напрям, вкл./викл), розприскувачем [4], сервоприводами, що можуть змінювати положення клапана чи люка розвантаження.

ArduPilot Mission Planner може працювати з різними типами автопілотів, такими як ArduPilot Mega (APM), Pixhawk і Cube. Він підтримує підключення до автопілота через USB, бездротове з'єднання або за допомогою телеметричних модулів.

Автопілот ArduPilot (контролер на борту БПЛА) також підтримує різноманітні типи дронів, включаючи квадрокоптери, гексакоптери, вертольоти, літаки та багато інших.

ArduPilot надає широкий спектр функцій і можливостей для автономного польоту. Він підтримує точне GPS-позиціонування, автоматичні режими стабілізації, навігацію, збереження маршрутів, планування місій, автоматичний зліт і посадку, управління камерами, телеметрію і багато іншого.

План роботи

1. Запланувати в ArduPilot Mission Planner політ БПЛА під керуванням ArduPilot за простим маршрутом.

2. Змоделювати зліт й політ БПЛА в ArduPilot Mission Planner.

3. Запланувати й змоделювати політ БПЛА над заданою ділянкою для задач спостереження.

Порядок виконання роботи

1. Запустити ArduPilot Mission Planner.

2. Перейти в режим «План» для планування маршруту (1 на рис. 7.1). Обрати режим відліку висоти (абсолютна над рівнем моря, від поверхні землі, відносно місця старту), обравши потрібний варіант зі списку (2 на рис.7.1). Задати не менше 4 точок маршруту над ділянкою з координатами 50.682537 пн.ш., 26.278592 сх.д.

Для додавання точки маршруту клацніть по точці на карті з потрібними координатами, в контекстному меню оберіть «Вставити Wp» (3 на рис.7.1). Також точки маршруту можна

додавати, клацнувши по «+» на лінії між двома точками. Список доданих точок відобразиться нижче (4 на рис.7.1). Для точки маршруту можна обрати іншу роль (наприклад, точка ввімкнення камери, зміни швидкості, ввімкнення розбризкувача тощо). За потреби в цьому ж списку можна змінити висоту літального апарату в точці (5 на рис.7.1). Отриманий маршрут зберегти в файл.

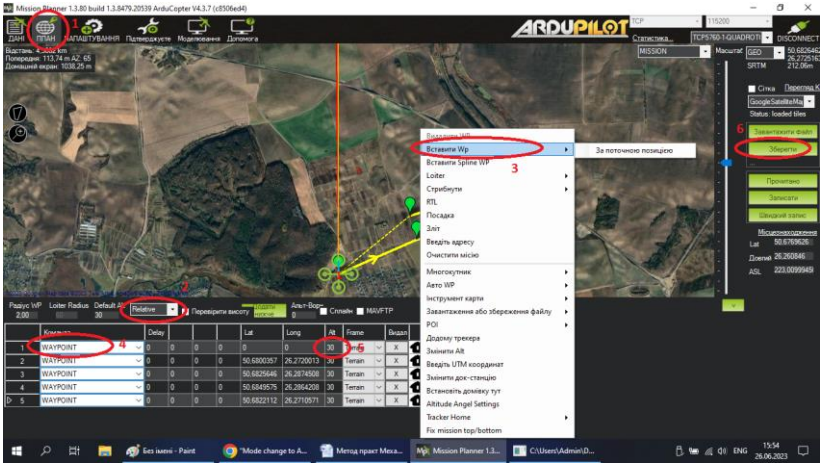


Рис. 7.1. Планування маршруту польоту в ArduPilot Mission Planner

3. Перейти в режим «Моделювання» (1 на рис. 7.2). Перемістити базу (2 на рис. 7.2) ближче до земельної ділянки, над якою планувався політ (за замовчуванням у версії 1.3.80 вона знаходиться в Австралії). В списку «Модель» обрати вид безпілотного апарату під керуванням ArduPilot, який планується використовувати – квадрокоптер (3 на рис. 7.3). Лишається завантажити й запустити симулятор для відповідного типу безпілотних апаратів – обираємо «мультикоптер» (4 на рис. 7.3) і в діалоговому вікні з запитанням, яку версію симулятора використовувати, відповідаємо «Stable» (стаблільну). Зачекати, доки завантажиться і запуститься симулятор.

Рис. 7.2. Налаштування моделювання в ArduPilot Mission Planner

4. Перейти в режим «Дані». Ліворуч на вкладці «Дії» змінити режим на QSTABILIZE або Stabilize (рис. 7.3), підтверити зміну натиканням кнопки поряд «Режим набору», натискнути «Арм» (запустити двигуни), після чого клацнути правою кнопкою по карті й обрати «Зліт».

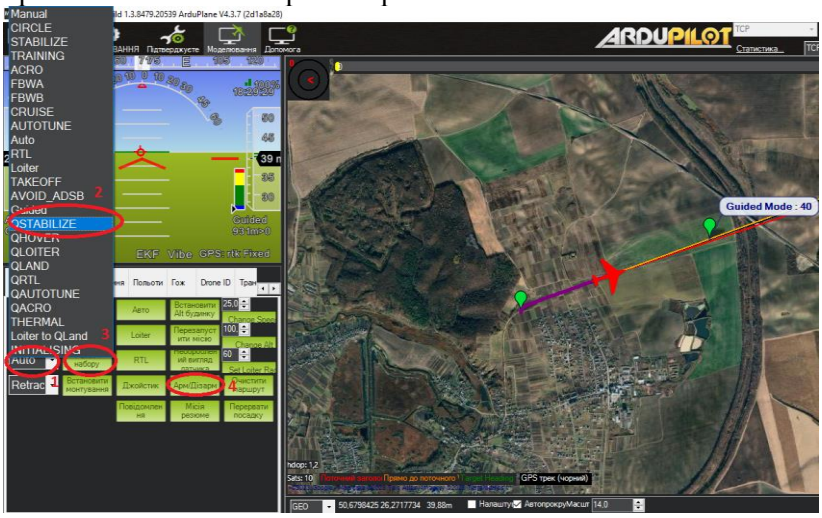


Рис. 7.3. Моделювання зльоту в ArduPilot Mission Planner

Увага! Якщо після ввімкнення двигунів впродовж декількох секунд не надходить жодних команд, двигуни вимикаються і відображається статус «Disarmed».

Далі змінюємо режим на «Auto», в якому літальний апарат виконуватиме наперед задану місію з обльоту ранішезаданих точок.

В режимі «Guided» БПЛА виконуватиме команди (летіти в точку, змінити висоту) від ArduPilot Misson Planner у реальному часі.

5. Вибрати довільну земельну ділянку з врахуванням обежень для літальних апаратів масою до 20 кг згідно пункту 4 розділу II. Регулювання використання повітряного простору Авіаційних правил України «Правила використання повітряного простору України» [5] (рис. 7.4).

Рис. 7.4. П’ятикілометрова зона обмеження польотів безпілотних літальних апаратів навколо Міжнародного аеропорту «Рівне» в ArduPilot Mission Planner

6. На вибраній ділянці побудувати полігон (многокутник), в контекстному меню обрати «Авто WP» – «спостереження/опитування (сітка)» (рис. 7.5). Прийняти параметри сітки за замовчуванням.

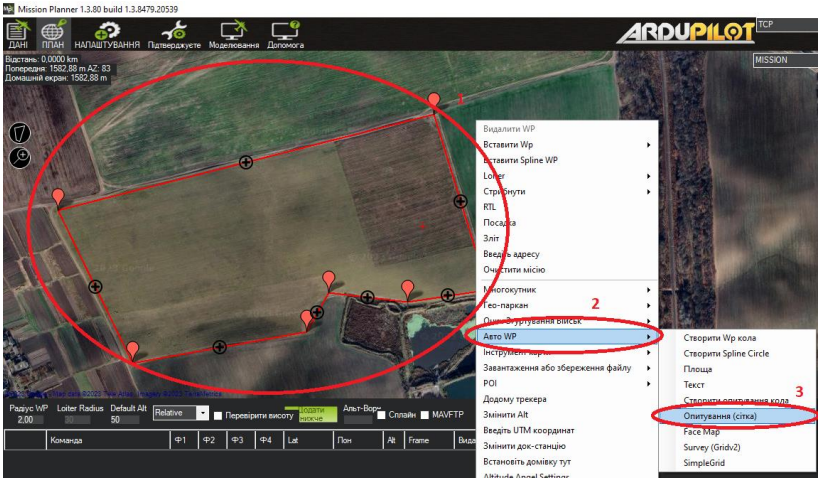


Рис. 7.5. Вибір ділянки для автоматичного створення точок маршруту для спостереження за нею

ArduPilot Mission Planner побудує план польоту БПЛА над ділянкою (рис. 7.6). Змоделювати політ згідно вказівок п.3-4.

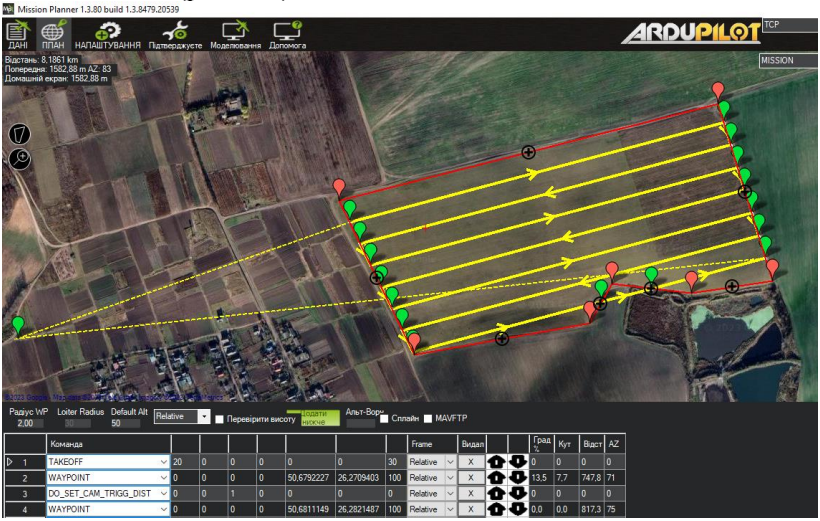


Рис. 7.6. План руху БПЛА над ділянкою

7. Зробити висновки. Звіт повинен містити: титульний

лист; тему, мету роботи; порядок виконання; скріншоти в режимі моделювання польоту БПЛА; висновки.

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. Crop Sprayer - Copter documentation - ArduPilot: веб-сайт. URL: https://ardupilot.org/copter/docs/sprayer.html (дата звернення: 26.06.2023).

5. Авіаційні правила України «Правила використання повітряного простору України»: затверджені наказом Державної авіаційної служби України і Міністерства оборони України від 11 травня 2018 року № 430/210. URL: https://zakon.rada.gov.ua/laws/show/z1056-18#Text (дата звернення: 26.06.2023).