МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
Кафедра “Підйомно-транспортного та робототехнічного обладнання"

МЕТОДИЧНІ ВКАЗІВКИ ДО ПРАКТИЧНИХ ЗАНЯТЬ

з дисципліни «Мехатроніка»

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 131 ПРИКЛАДНА МЕХАНІКА
Освітня програма – *Інженерія логістичних систем, Мехатроніка та промислові роботи*

Спеціальність – 133 ГАЛУЗЕВЕ МАШИНОБУДУВАННЯ
Освітня програма – *Підйомно-транспортні, будівельні дорожні машини і обладнання*

ОДЕСА 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
Кафедра “Підйомно-транспортного та робототехнічного обладнання"

Михайлов Євген Павлович

МЕТОДИЧНІ ВКАЗІВКИ ДО ПРАКТИЧНИХ ЗАНЯТЬ

з дисципліни «Мехатроніка»

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 131 ПРИКЛАДНА МЕХАНІКА
Освітня програма – *Інженерія логістичних систем, Мехатроніка та промислові роботи*

Спеціальність – 133 ГАЛУЗЕВЕ МАШИНОБУДУВАННЯ
Освітня програма – *Підйомно-транспортні, будівельні дорожні машини і обладнання*

Затверджено
на засіданні кафедри
підйомно-транспортного і
робототехнічного обладнання
Протокол № 1 від 26.08.2021 р.

ОДЕСА 2021

Методичні вказівки до практичних занять з дисципліни "Мехатроніка" для здобувачів першого (бакалаврського) рівня вищої освіти, спеціальності: 131 - Прикладна механіка, освітні програми: – Мехатроніка та промислові роботи, Інженерія логістичних систем, спеціальності: 133 - Галузеве машинобудування, освітня програма: – Підйомно-транспортні, будівельні дорожні машини і обладнання. / Укл.: Михайлов Є. П. – Одеса: ОП, 2021. - 38 с.

Укладач:
Михайлов Є.П. доц. кафедри підйомно-транспортного і робототехнічного обладнання

У методичних вказівках наведені практичні заняття, метою яких є ознайомлення з системами керування машин та обладнання для мехатронних систем, таких як робототехнічні та транспортні пристрої. У роботах розглянуті питання програмного керування окремими мехатронними пристроями та їхньої взаємодії.

Зміст

Заняття 1. Регульовані приводи постійного струму

Мета заняття: Назвати особливості використання двигунів постійного струму.
Визначити характеристики двигунів постійного струму згідно з метою використання.
Описати, як здійснити регулювання швидкості обертання двигунів постійного струму.

1. Теоретичні положення

Широке застосування в мехатронних пристроях, що використовуються в мобільних системах, наприклад, в мобільних роботах, знайшли двигуни постійного струму.

На рис. 1 наведений принцип дії двигуна постійного струму.

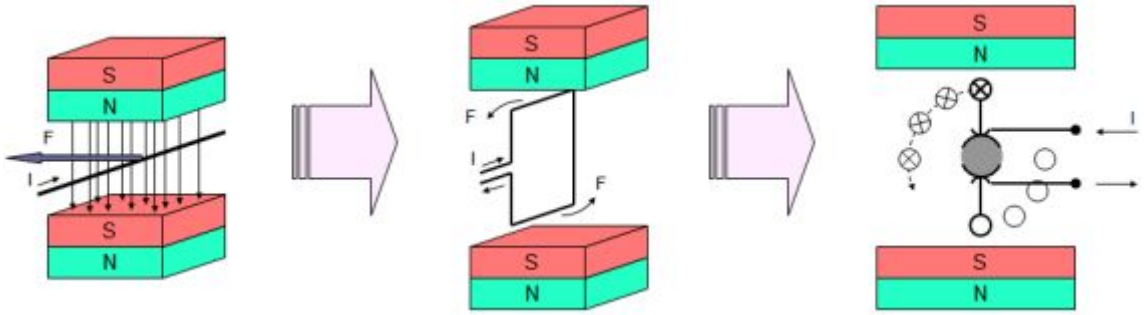


Рис. 1. Принцип дії двигуна постійного струму

Для підключення двигунів постійного струму у найпростішому випадку використовують реле або контактори. Оскільки двигун представляє собою індуктивність (рис. 2), то при вимиканні струм не може змінитися миттєво і на контакті виникає велика напруга.

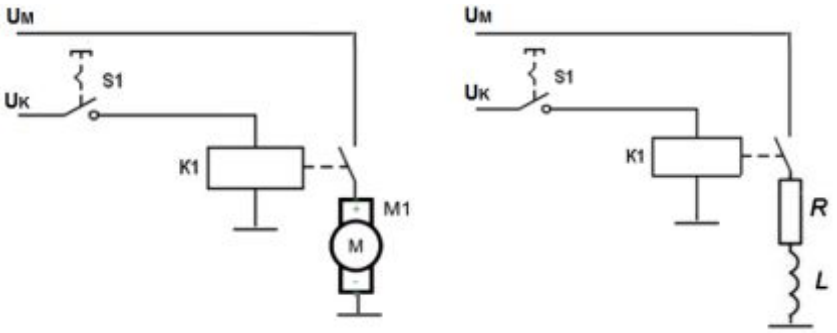


Рис. 2. Схема підключення двигуна постійного струму

Для усунення цього недоліку використовують захисні діоди (рис. 3), що замикають ланцюг при вимиканні двигуна.

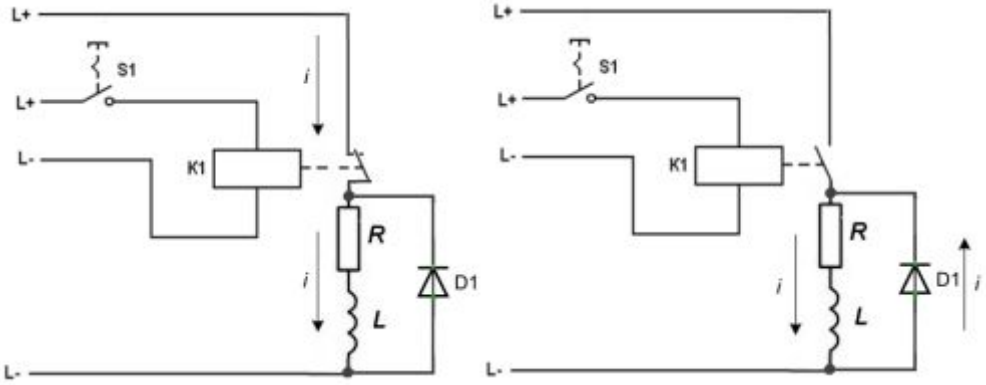
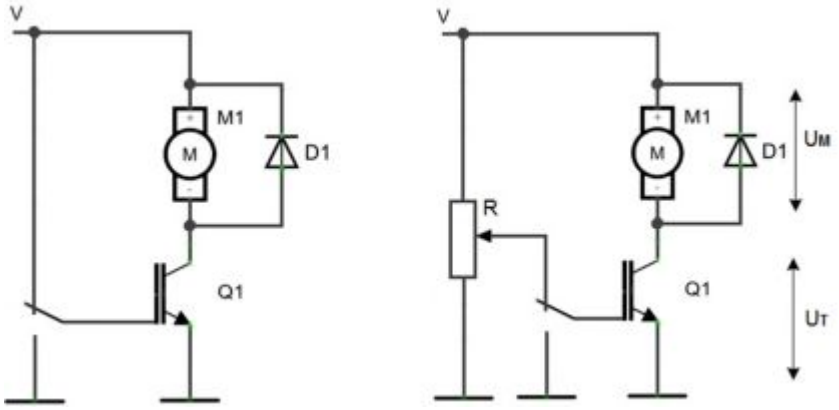


Рис. 3. Схема підключення захисних діодів

Сучасні системи керування замість реле найчастіше використовують електронні комутаційні схеми на IGBT-транзисторах (рис. 4, а), що поєднують позитивні характеристики біполярних та польових транзисторів, а саме, високий вхідний опір та великий вихідний струм



а) б)

Рис. 4. Комутаційна схема з використанням IGBT-транзистора без регулювання (а) таз регулюванням швидкості обертання (б)

Для керування двигуном постійного струму треба вирішувати дві задачі:

1) керування швидкістю обертання;

2) керування напрямком обертання.

Для керування швидкістю обертання треба змінювати напругу, що подається на двигун.

При використанні електронні схеми на IGBT-транзисторах це можна зробити шляхом зміни вхідної напруги. Але при цьому напруга живлення буде поділятися між мотором та транзистором (рис. 4, б).

Для усунення цього недоліку використовують сигнали з широтно-імпульсною модуляцією.

У широтно-імпульсному перетворювачі частота проходження імпульсів постійна, а тривалість імпульсів плавно змінюється, що забезпечує регулювання, як це показано на рис. 5.

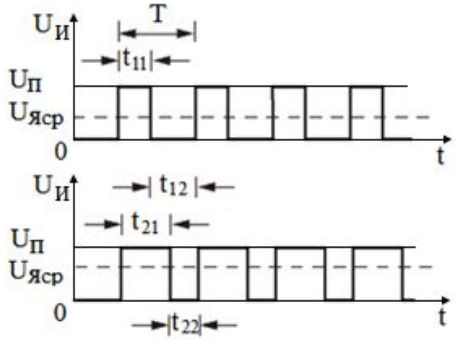
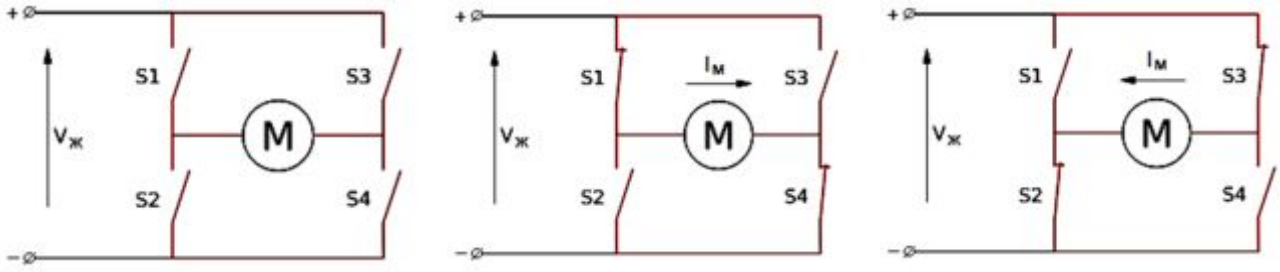


Рис. 5. Сигнал з широтно-імпульсною модуляцією

Для керування напрямком обертання у двигунів постійного струму використовують так званий Н-міст

Схема, що наведена на рис. 6, а, дає можливість здійснювати реверсивне керування.

Якщо замкнути ключі (контакти реле, транзистори, тиристори тощо) S1 та S4, двигун обертається в одну сторону - прямий напрямок (рис. 6, б), якщо замкнути ключі S2 та S3, двигун обертається в іншу сторону - зворотний напрямок (рис. 6, в).



а) б) в)

Рис. 6. Керування напрямком обертання двигунів постійного струму за допомогою Н-моста

Схема, що наведена на рис. 7, представляє собою Н-міст на IGBT-транзисторах.

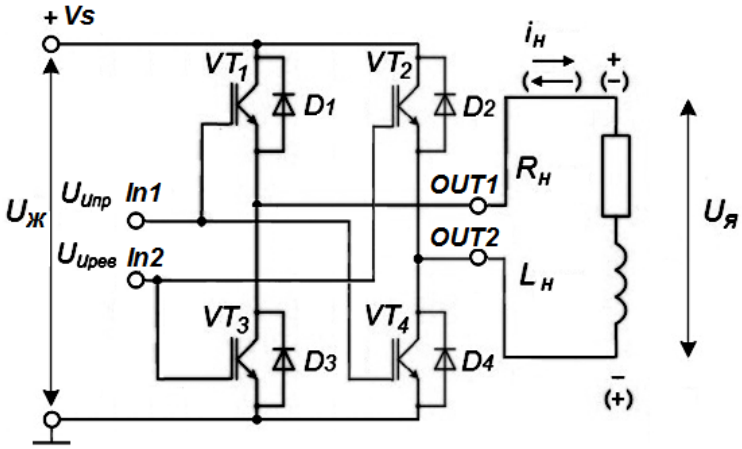


Рис. 7. Н-міст на біполярних транзисторах

Імпульсна напруга $Ui_{пр}$ відкриває ключові транзистори VT_1, VT_4, і струм через навантаження (двигун) тече в прямому напрямку, імпульсна напруга $Ui_{рев}$ відкриває ключові транзистори VT_2, VT_3, і струм через навантаження (двигун) тече в зворотному напрямку (реверс).

Сигнали на виходах OUT1 та OUT2 мають ті ж значення, що й на відповідних входах In1 та In2, тому включення обертання двигуна в одну сторону на вхід In1 високий рівень сигналу, а на вхід In2 низький, а для обертання в іншу сторону, навпаки.

Для регулювання швидкістю обертання двигуна використовують сигнал з широтно-імпульсною модуляцією (ШІМ). Цей сигнал можна подавати на один з входів.

На рис. 8 наведені силові модулі для мікроконтролерів Arduino, що мають виходи з ШІМ Motor Driver L298N (а), Motor Driver L9110S (б), Motor Shield L293D (в).

а) б) в)

Рис. 8. Силові модулі для мікроконтролерів Arduino, що мають виходи з ШІМ Motor Driver L298N (а), Motor Driver L9110S (б), Motor Shield L293D (в).

2. Приклади розв'язання задач з теми заняття.

Розглянемо, як працюють модулі управління електродвигунами на прикладі драйвера двигуна Motor Driver L298N. Модуль виконаний на основі мікросхеми L298N, яка представляє собою два мостових драйвера, і може бути використаний для управління двома двигунами постійного струму (DC моторів). (рис. 9).

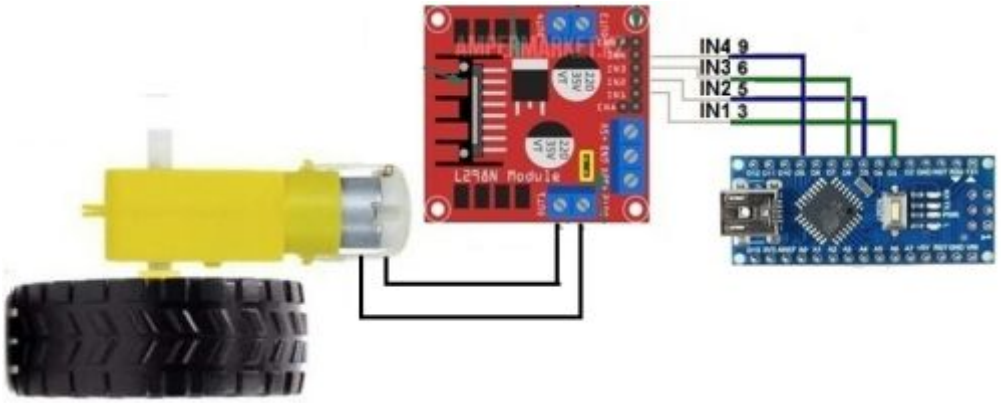


Рис. 9. Приклад підключення DC мотора до контролера

Розглянемо, як здійснюється керування двигуном постійного струму за допомогою контролерів Arduino.

Керування без регулювання швидкості здійснюється за допомогою функції:

digitalWrite(pin, value);

де:

pin – номер виходу контролера,

value – змінна типу bit, яка задає логічний рівень HIGH, або LOW (включити або виключити) на заданому цифровому виводі pin.

Далі наведений приклад програми керування двигуном постійного струму, що підключений до двох виходів контролера Arduino NANO.

Згідно з програмою двигун виконує такі дії

- обертання на протязі 2 с в одну сторону,
- зупинка на 1 с,
- обертання на протязі 2 с в другу сторону,
- зупинка на 1 с,
- потім все повторюється.

Приклад програми керування двигуном постійного струму

```
const int IN1 = 3;
const int IN2 = 5;  //підключення входів модуля до відповідних контактів
void setup() {
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT); }
void loop() {
digitalWrite(IN1, HIGH);  //обертання вперед,
digitalWrite(IN2, 0);
delay(2000);      //затримка 2 с
digitalWrite(IN1, LOW);    //зупинка
digitalWrite (IN2, LOW);
delay(1000);      //затримка 1 с
}
```

Для керування швидкістю обертання сигнал з ШІМ можна подати на один з входів.

Розглянемо, як здійснюється керування двигуном постійного струму за допомогою контролерів Arduino, які мають можливість видавати сигнали з широтно-імпульсною модуляцією (ШІМ).

Управління виходом ШІМ контролерів Arduino при використанні програмної оболонки IDE здійснюється за допомогою функції analogWrite ().

Ця функція має два аргументи: номер виходу, на який виводиться сигнал ШІМ, і число в діапазоні від 0 до 255, яке задає пропорційну тривалість імпульсу ШІМ та у даній програмі змінює швидкість обертання мотору.

Регулювання швидкості здійснюється за допомогою функції:

analogWrite(pin, value);

де:

pin – номер виходу контролера, який підтримує ШІМ, при цьому треба враховувати, що у контролерів Arduino UNO та Arduino NANO ШІМ підтримують виходи 3, 5, 6, 9. 10, 11.

value – змінна типу byte, яка задає тривалість імпульсу, та має значення значення від 0 до 255.

Для керування швидкістю обертання двигунів можна використовувати потенціометр, сигнал з яких подається на аналогові входи (рис. 10).

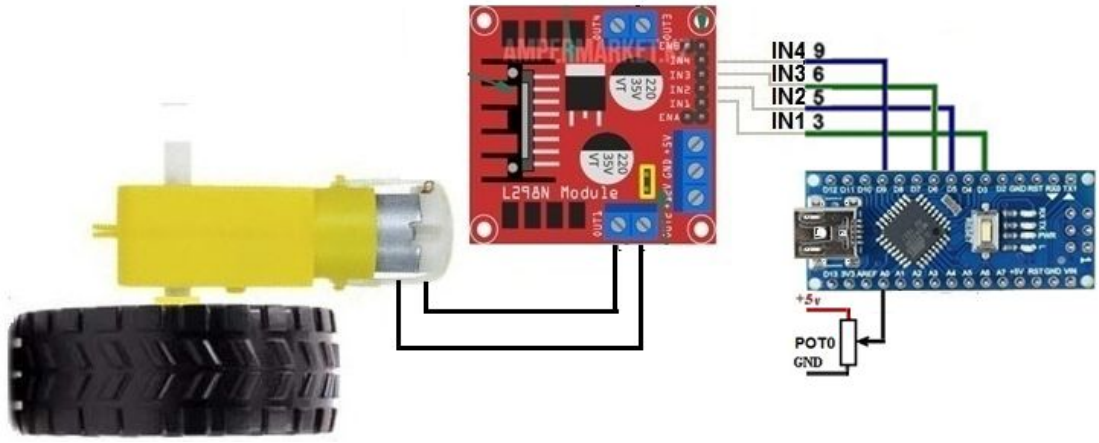


Рис. 10. Керування швидкістю обертання за допомогою потенціометра

Оскільки аналоговий вхід видає значення від 0 до 1023, а для регулювання швидкістю за допомогою ШІМ треба задавати значення від 0 до 255, здійснюється масштабування значення за допомогою функції

map(value, fromLow, fromHigh, toLow, toHigh),

яка перетворює значення змінної з одного діапазону в інший, а саме значення змінної value, що дорівнює fromLow, буде перетворено в число toLow, а значення fromHigh - в toHigh.

Всі проміжні значення value масштабируются відносного нового діапазону [toLow; toHigh].

Програма керування двигунами, яка здійснює обертання в прямому та зворотному напрямку з регулюванням швидкості обертання окремо для обертання в прямому та зворотному напрямках (потенціометр POT0). Значення потенціометрів виводиться на монітор за допомогою функції Serial.println(var) значення змінної speed після масштабування.

```
const int IN1 = 3;
const int IN2 = 5;  //підключення входів модуля до відповідних контактів
int speed = 0;      //швидкість обертання
#define  POT0 0     //потенціометр

void setup()  {
pinMode(IN1, OUTPUT);
```

```
pinMode(IN2, OUTPUT);
Serial.begin(9600);  }

void loop()   {
speed = analogRead(POT0);
speed = map(speed, 0, 1023, 0, 255);
Serial.println(speed);
analogWrite(IN1, speed);  //обертання вперед
analogWrite(IN2, 0);
delay(2000);      //затримка 2 с
analogWrite(IN1, 0);    //зупинка
analogWrite(IN2, 0);
delay(1000);      //затримка 1 с
analogWrite(IN1, 0);  //обертання назад
analogWrite(IN2, speed);
delay(2000);      //затримка 2 с
analogWrite(IN1, 0);    //зупинка
analogWrite(IN2, 0);
delay(1000);      //затримка 1 с  }
```

Завдання 1

Скласти програму керування двох двигунів постійного струму з реверсом з використання чотирьох виходів з ШІМ для наведеної схеми (рис. 11).

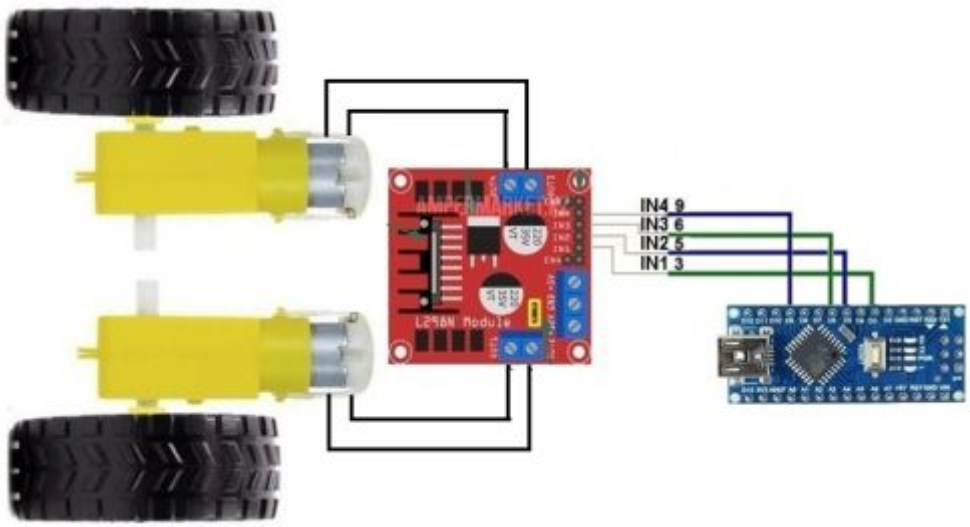


Рис. 11. Схема керування двох двигунів постійного струму з реверсом з використання чотирьох виходів з ШІМ

Завдання 2

Скласти програму керування двох двигунів постійного струму з реверсом та регулюванням швидкості обертання з використання чотирьох виходів з ШІМ для наведеної схеми (рис. 12).

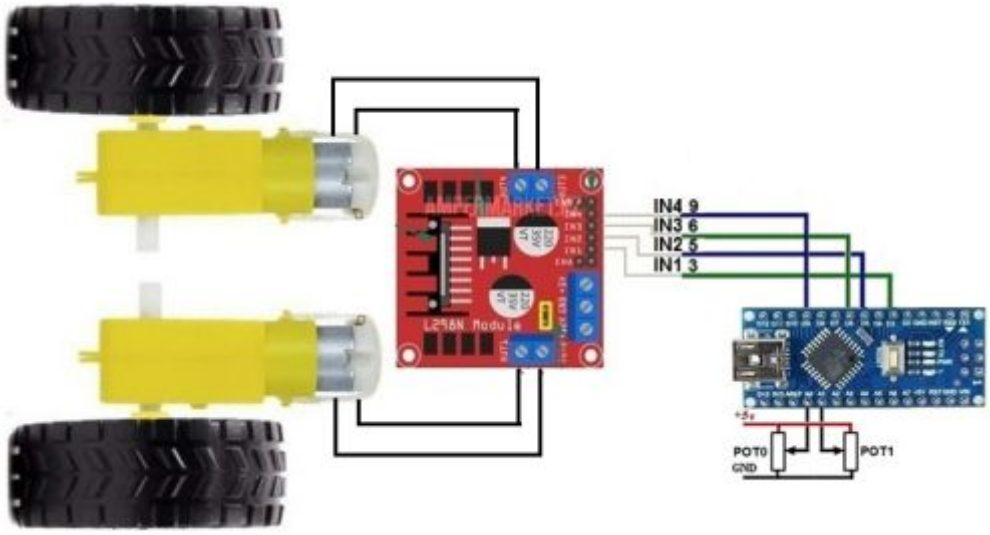


Рис. 11. Схема керування двох двигунів постійного струму з реверсом та регулюванням швидкості обертання з використання чотирьох виходів з ШІМ

Завдання 3

Перевірити роботу програм за допомогою емулятора UnoArduSim.

Контрольні запитання за темою заняття.

1. Визначити, як здійснюється регулювання швидкості обертання двигунів постійного струму?
2. Описати, які недоліки мають схеми з підсилювачем напруги?
3. Визначити, як здійснюється регулювання швидкості обертання за допомогою ШІМ?
4. Описати, як здійснюється напрямок обертання двигунів постійного струму?
5. Назвати, що представляє собою так званий Н-міст?

Заняття 2. Регульовані приводи крокових двигунів

Мета заняття: Назвати особливості використання крокових двигунів.
Визначити характеристики крокових двигунів згідно з метою використання.
Описати, як здійснити регулювання швидкості обертання крокових двигунів.

1. Теоретичні положення

Високу точність позиціонування без датчиків зворотного зв'язку (швидкості або положення) можна здійснити за допомогою крокових двигунів.

Принцип роботи крокового двигуна заснований на використанні такої конструкції, при якої один вхідний імпульс повертає ротор на визначений кут. Імпульси поступають послідовно на різні обмотки, що забезпечує обертання з постійною швидкістю.

Спрощена схема уніполярного крокового двигуна наведена на рис. 1.

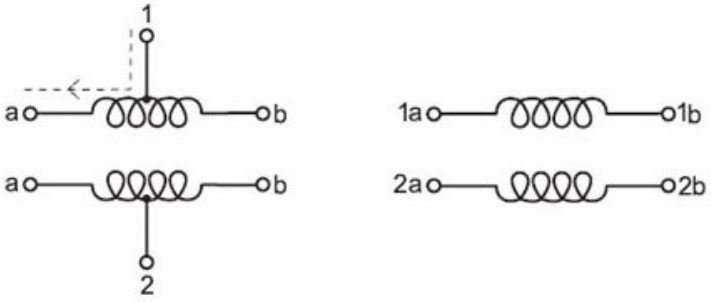


Рис. 1. Спрощена схема уніполярного крокового двигуна

На рис. 2 показаний принцип роботи крокового двигуна. Послідовність подачі сигналів, яка потребується для обертання двигуна наведена на рис.2 а, б.

Контролер формує послідовність імпульсів зі змінним періодом повторення, що дозволяє керувати швидкістю обертання двигуна. Оскільки контролер видає послідовність імпульсів на один вихід потрібна схема, яка сформує послідовну видачу імпульсів на контакти 1а, 1б, 2а, 2б.

Ці функції виконує силовий перетворювач, який забезпечує також потрібну напругу та струм на виході (рис. 2 в). Діоди на виходах перетворювача потрібні для замикання струму котушок двигуна при відключенні.

Змінюючи період проходження імпульсів можна змінювати швидкість обертання крокового двигуна по заданому закону.

Позицію задає кількість імпульсів, які поступають на кроковий двигун.

Швидкість обертання крокових двигунів визначається частотою проходження імпульсів, а їх кількість дозволяє повернути вал двигуна на певний кут, що дає можливість здійснити позиціонування без датчика зворотного зв'язку (датчика кута повороту).

Силові модулі двигунів постійного струму для мікроконтролерів Arduino можна використовувати також для крокових двигунів, для чого треба підключити один кроковий двигун на чотири виходи.

Силові модулі для мікроконтролерів Arduino, які були розглянуті у попередньому занятті, можна використовувати також для крокових двигунів (один кроковий двигун замість двох двигунів постійного струму)

2. Приклади розв'язання задач з теми заняття

Розглянемо кроковий двигун 28BYj-48 з драйвером ULN2003 (рис. 2).

У 4-кроковому режимі він може здійснювати 2048 кроків, у 8-кроковому 4096 кроків. Живлення 5 В, струм споживання 160 мА.

Двигун має умонтований редуктор з передавальним числом 1:64, тобто один крок він зробить на 5,625 градусів.

Крутний момент становить 34 мН.м.
Середня швидкість 15 об / хв.

Рис.2. Підключення крокового двигуна до контролера

Керування кроковим двигуном 28BYj-48 з драйвером ULN2003 за допомогою контролера Arduino Uno. Схема підключення наведена на рис. 3.

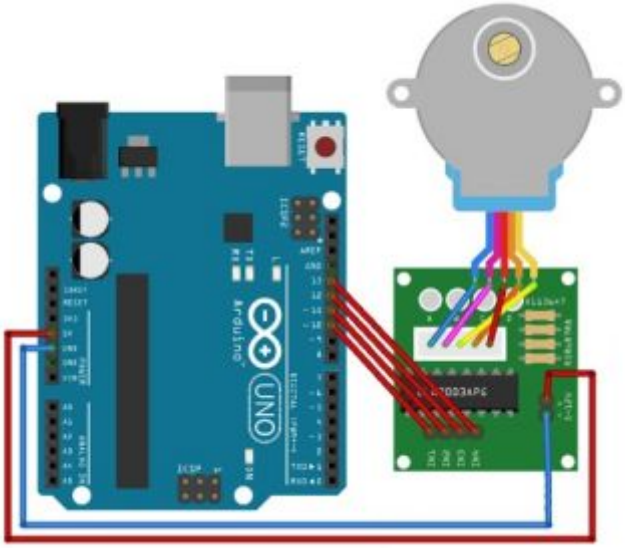


Рис. 3. Схема підключення крокового двигуна 28BYj-48 з драйвером ULN2003 до контролера Arduino Uno

Контакти модуля ULN2003 IN1, IN2, IN3 и IN4 підключені до відповідних цифрових контактів D10, D11, D12 и D13 у режимі виходів на Arduino.

Для керування кроковим двигуном використовується бібліотека Stepper Library в Arduino IDE.

На рис.4 наведена послідовність подавання імпульсів для крокового режиму, яку використовує бібліотека Stepper. Кроковий режим - це коли дві з чотирьох обмоток живляться на кожному кроці.

Рис.4. Послідовність подавання імпульсів для крокового режиму

Бібліотека Stepper має наступні функції.

Функція встановлення параметрів крокового двигуна:

Stepper myStepper = Stepper(steps, pin1, pin2, pin3, pin4);

де myStepper – ім'я крокового двигуна,

steps - кількість кроків на одне обертання,

pin1, pin2, pin3, pin4 - виходи контролера, що використовуються для керування двигуном (у залежності від типа двигуна використовують два або чотири виходи).

Функція встановлення швидкості обертання двигуна myStepper:

myStepper.setSpeed(speed);

де speed - швидкість обертання у кількості обертів за хвилину.

Кількість кроків, на яке буде обертатися двигун, задає функція

myStepper (позначка "–" означає, що двигун буде обертатися в іншу сторону):

myStepper.step(steps);

де steps - кількість кроків.

Функція myStepper.step(steps); блокує виконання програми, а саме, вона очікує закінчення обертання двигуна, перш ніж передати управління в наступний рядок коду.

Наприклад, якщо швидкість обертання встановлена на 1 обертання за хвилину), і викликано step (200) на двигуні з повним оборотом в 200 кроків, то виконання цієї функції займе цілу хвилину.

Тому краще організовувати управління так, щоб швидкість була висока, а за один виклик step () робилося лише кілька кроків, для уникнення тривалого блокування виконання коду скетчу (програми).

Швидкість переміщення встановлює функція затримки в циклі:

delay(time), де time - затримка у мілісекундах.

Далі наведений приклад програми, яка здійснює одне обертання в одну сторону та одне в іншу.

Між циклами обертання затримка 1,0 с.

Програма задає в циклі переміщення на 1 крок та затримку на 5 мс, а остаточна кількість кроків та затримки встановлює функція встановлення кількості циклів:

```
for (int i=1; i<=2048; i++)
{
…
}
```

```
#include <Stepper.h>
// Призначаються виводи для IN1-IN3-IN2-IN4
// для викоританняя Stepper с 28BYJ-48
// IN1 - 10
// IN2 - 11
// IN3 - 12
// IN4 - 13
Stepper stepper(64, 10, 12, 11, 13);
void setup()
{
stepper.setSpeed(500);
}

void loop()
{
//одне обертання вперед
for (int i=1; i<=2048; i++)  //рух вперед
{
stepper.step(1);
```

```
delay (5);
}
delay (1000);
//одне обертання назад
for (int i=1; i<=2048; i++)//рух назад
{
stepper.step(-1);
delay (5);
}
delay (1000);
}
```

Завдання 1

Скласти програму керування кроковим двигуном з реверсом без регулювання швидкості обертання для контролера Arduino.

Звдання 2

Скласти програму керування кроковим двигуном з переміщення на встановлену кількість градусів з реверсом та регулюванням швидкості обертання шляхом встановлення відповідного значення для контролера Arduino.

Завдання 3

Перевірити роботу програм за допомогою емулятора UnoArduSim.

Контрольні запитання за темою заняття.

1. Описати, які засоби використовують для регулювання крокових двигунів?
2. Визначити, як здійснюється регулювання швидкості обертання крокових двигунів?
3. Визначити, як здійснюється зміна напрямку обертання крокових двигунів?
4. Описати, як здійснюється позиціонування за допомогою крокових двигунів?
5. Назвати, що представляють собою силові модулі крокових двигунів?

Заняття 3. Регульовані сервоприводи

Мета заняття: Назвати особливості використання сервоприводів.
Визначити характеристики сервоприводів згідно з метою використання.
Описати, як здійснити регулювання швидкості обертання сервоприводів.

1. Теоретичні положення

Сервопривід – це привід з управлінням через негативний зворотній зв'язок, що дозволяє точно керувати параметрами руху.

Сервоприводом є будь-який тип механічного приводу, що має в складі давач (положення, швидкості, зусилля і т.п.) і блок управління приводом, що автоматично підтримує необхідні параметри на давачі і пристрою згідно заданому зовнішньому значенню.

Підключення до Arduino

Багато сервоприводів можуть бути підключені до Arduino безпосередньо. Для цього від них іде шлейф з трьох проводів:

червоний – живлення – *підключається до контакту 5V або безпосередньо до джерела живлення*

коричневий або чорний – *земля (GND контакт Arduino)*

жовтий або білий – *сигнал; підключається до цифрового виходу Arduino.*

Можна генерувати керуючі імпульси самостійно, але це настільки поширена задача, що для її спрощення існує стандартна бібліотека *Servo*.

Обмеження по живленню

Непотужний сервопривід під час роботи споживає не більше 100 мА. При цьому Arduino здатне видавати струм до 500 мА.

Якщо в проекті необхідно використовувати кілька сервоприводів або потужні сервоприводи, то треба забезпечити підключення сервоприводів в контур з додатковим живленням.

Бібліотека *Servo* дозволяє здійснювати програмне керування сервоприводами. Для цього створюється об'єкт класу Servo.

Функції бібліотеки Servo

Управління здійснюється наступними функціями:

attach () – закріпляє привід до конкретного піну. Можливі два варіанти синтаксису для цієї функції:

servo.attach (pin) і

servo.attach (pin, min, max),

при цьому pin – номер піна, до якого приєднують сервопривід,

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 функції:

servo.write (angle), де angle – кут, на який повинен обернутися сервопривід.

writeMicroseconds () – віддає команду надіслати сервоприводу імпульс певної довжини, є низькорівневим аналогом попередньої команди. Синтаксис функції :

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

read () – читає поточне значення кута, в якому знаходиться сервопривід. Синтаксис функції :

servo.read (), повертається ціле значення від 0 до 180.

attached () – перевірка, чи був приєднаний об'єкт до конкретного піну. Синтаксис функції :

servo.attached (), логічна одиниця повертається, якщо об'єкт був приєднаний до якого-небудь піну, або нуль в противному випадку.

detach () – виконує дію, зворотну дії *attach ()*, тобто від'єднує об'єкт від піна, до якого він був приписаний.

Синтаксис функції :

servo.detach ().

Бібліотека VarSpeedServo

Бібліотека VarSpeedServo-master.h Arduino дозволяє використовувати до 8 сервоприводів, що рухаються асинхронно (оскільки вона використовує переривання).

Крім того, можна встановити швидкість переміщення та додатково почекати (заблокувати), поки робота сервоприводу не буде завершена, а також створити послідовності ходів, які виконуються асинхронно.

Ця бібліотека є адаптацією стандартної бібліотеки Arduino Servo.h та має такі можливості:

- підтримує до 8 сервоприводів
- дозволяє одночасний, асинхронний рух всіх сервоприводів;
- може бути встановлена швидкість руху;
- функція write () ініціює переміщення і може додатково дочекатися завершення переміщення, перш ніж повернутися;
- на сервоприводи може бути відправлена послідовність ходів (де кожен хід має позицію і швидкість).

Функції бібліотеки VarSpeedServo

- **attach(pin)** - приєднує серводвигун до входу / виходу;
- **write(value)** - встановлює кут серво в градусах. (Невірний кут, розглядається як імпульс в мікросекундах);
- **write(value, speed)** - speed змінює швидкість руху в нову позицію 0 = повна швидкість, 1-255 від повільної до швидкої;
- **write(value, speed, wait)** - wait - логічне значення, яке, якщо true, викликає блокування виклику наступної функції до завершення переміщення;
- **read()** - отримує останню записану ширину імпульсу серво як кут між 0 і 180;
- **wait ();** // очікування закінчення руху;

Приклад програми

```
#include <VarSpeedServo.h>
VarSpeedServo myservo;    // складає servo object
void setup() {
myservo.attach(2);  // підлючає серво до pin 2 контролера
myservo.write(0, 10, true);   // переміщення на вихідне положення
                    //(0 градусів) зі швидкістю 10, поки не завершиться рух
}
void loop() {
myservo.write(0, 50, true);    // переміщення на 0 градусів,
delay(500);
myservo.write(90, 50, true);          // переміщення на 90 градусів,
delay(500);
myservo.write(180, 50, true);         // переміщення на 180 градусів,
delay(500);
myservo.write(90, 50, true);          // переміщення на 90 градусів,
delay(500);
}
```

Потенціометр

Для керування зміною положення сервоприводу можна використати потенціометр (рис. 1). Він має три контакти, що підключаються наступним чином. Два крайні контакти (як правило) це живлення і земля, а середній – інформаційний.

Під’єднуємо:

- живлення потенціометра –> 5 V Arduino,
- земля –> GND Arduino,
- інформаційний –> аналоговий пін (0) Arduino.

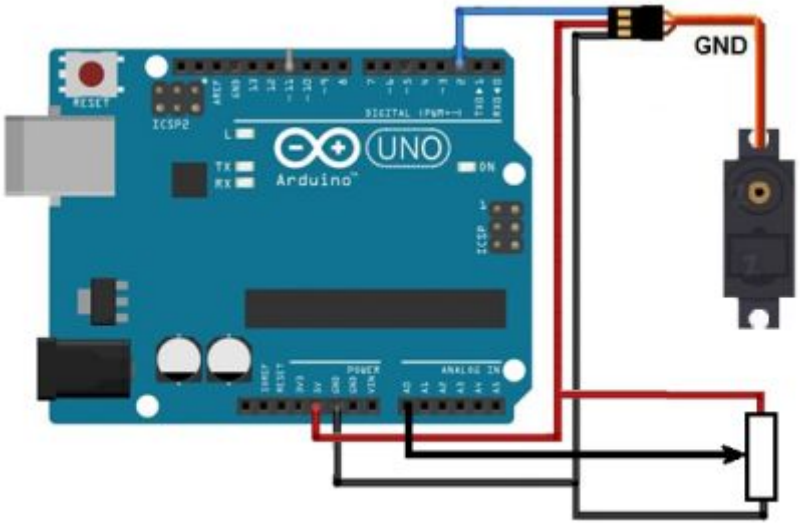


Рис. 1. Керування зміною положення сервоприводу за допомогою потенціометру

2. Приклад розв’язання задач з теми заняття.

Далі наведена програма, де задається обертання сервоприводу на кут, що встановлює потенціометр, який підключений до аналогового входу 0.

Оскільки аналоговий вхід видає значення від 0 до 1023, а для кута повороту треба задавати значення від 0 до 180, здійснюється масштабування значення

val = analogRead(0)*10/57,

Або можна використати функцію map(value, fromLow, fromHigh, toLow, toHigh), яка перетворює значення змінної з одного діапазону в інший, а саме значення змінної value, що дорівнює fromLow, буде перетворено в число toLow, а значення fromHigh - в toHigh. Всі проміжні значення value масштабируются відносного нового діапазону [toLow; toHigh].

Текст програми (скетчу):

```
#include <Servo.h>
Servo myservo;                  // сервоприводу призначається ім'я myservo
int val=0;                  // змінна з ім'ям val для керування сервоприводом
void setup() {
myservo.attach(2);     // сигнал управління підключений до виходу 2
myservo.write(val);    // встановлення початкового положення
}
void loop() {
val = analogRead(0)*10/57;  // зчитування та масштабування сигналу
                            // з аналогового входу 0
myservo.write(val);         // видача керуючого сигналу
}
```

Для послідовної зміни значення можна використовувати функцію циклу.

Для збільшення значення:

```
for (int i=1; i<=180; i=i+10) {
val=i
delay(10);       // затримка в мс
}
```

Для зменшення значення:

```
for (int i=180; i>=1; i=i-10)  {
val=i
delay(10);        // затримка в мс
}
```

Приклад програми:

```
#include <Servo.h>
Servo myservo;                  // сервоприводу призначається ім'я myservo
int val=0;                  // змінна з ім'ям val для керування сервоприводу
void setup() {
myservo.attach(2);     // сигнал управління підключений до виходу 2
myservo.write(val);    // встановлення початкового положення
delay(1000); }
void loop() {
val=0;
for (int i=0; i<=180; i=i+20) {
val=i;
myservo.write(val);    // видача керуючого сигналу
delay(500);                // затримка в мс
} }
```

Завдання 1

Скласти програму керування сервоприводом для одноразового послідовного повороту від 0 до 180° з кроком 10° та затримкою на кожному кроці на 2 с.

Завдання 2

Скласти програму керування сервоприводом для періодичного послідовного повороту від 0 до 180° та від 180° до 0 з кроком 10° та затримкою на кожному кроці на 1 с.

Завдання 3

Перевірити роботу програм за допомогою емулятора UnoArduSim.

Контрольні запитання за темою заняття.

1. Описати, які засоби використовують для регулювання сервоприводів?
2. Визначити, як здійснюється регулювання швидкості обертання сервоприводів?
3. Назвати, в якому діапазоні здійснюється обертання сервоприводів?
4. Описати, як здійснюється позиціонування за допомогою сервоприводів?
5. Назвати бібліотеки, що здійснюють керування сервоприводами?

Заняття 4. Регульовані приводи як мехатронні пристрої

Мета заняття: Назвати особливості використання регульованих приводів.
Визначити характеристики регульованих приводів згідно з метою використання.
Описати, як здійснити регулювання параметрів регульованих приводів на прикладі маніпулятора.

1. Теоретичні положення

Розглянемо мехатронну систему на прикладі маніпулятора з чотирма ступенями руху, що має у своєму складі чотири регульованих приводи на основі сервоприводів, які підключені до чотирьох виходів контролера Arduino NANO (рис. 1).

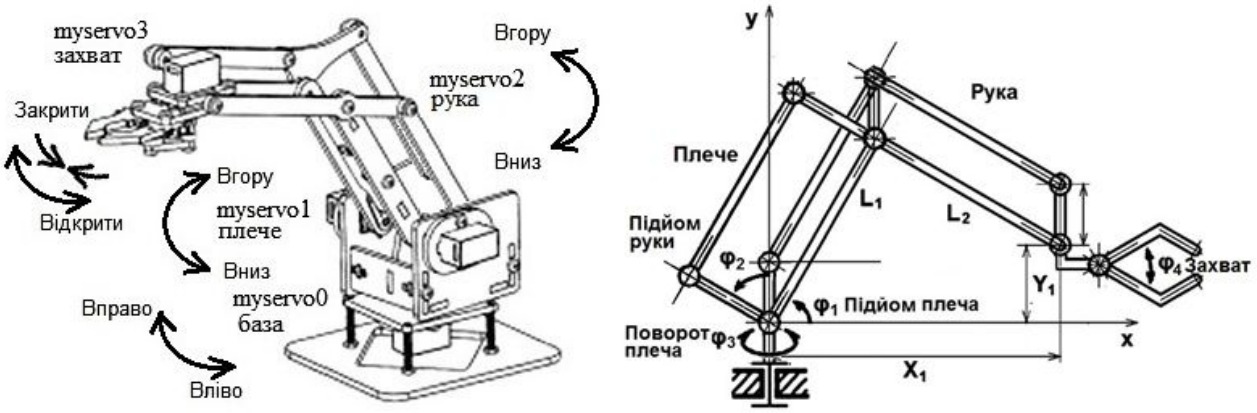


Рис. 1. Маніпулятор SNARM 4dof

Для переміщення ланок маніпулятора використовуються сервоприводи MG-90S, що забезпечують поворот валу від 0 до 180°.

Для керування маніпулятором використовуються контролери Arduino.

При використанні контролерів Arduino для спрощення керування сервоприводом використовується бібліотека Servo, яка була розглянута на попередньому занятті.

Розглянемо, як здійснити керування маніпулятором у ручному режимі.

Для керування переміщенням ланок маніпулятора використовується пульт керування у вигляді 4 потенціометрів.

Схема підключення маніпулятора до контролера Arduino Nano наведена на рис.2.

Оскільки для керування у ручному режимі використовуються потенціометри, що підключені до аналогових входів контролера, які видають значення від 0 до 1023, треба здійснити зміну масштабу, щоб отримати значення від 0 до 180.

Крім того для керування деякими ланками треба обмежувати діапазон, наприклад, для захвата треба подавати значення від 100 (захват відкритий) до 180 (захват закритий).

Для зчитування у змінну значення сигналу з аналогового входу pin використовується функція analogRead().

Функція має такий синтаксис: value = analogRead(pin); де value – змінна.

Для зміни масштабу можна використати функцію пропорційного перетворення значень від одного діапазону до другого map () та функція обмеження діапазону constrain ().

Далі наведений фрагмент програми ручного керування ланкою маніпулятора з сервоприводом myservo0 за допомогою значення `val0`, що отримане з аналогового входу POT0.

```
val0 = analogRead(POT0);    //зчитування сигналу з аналогового входу
val0 = map(val0, 0, 1023, 0, 180); //перетворення масштабу
myservo0.write(val0); //запис сигналу керування до сервоприводу
```

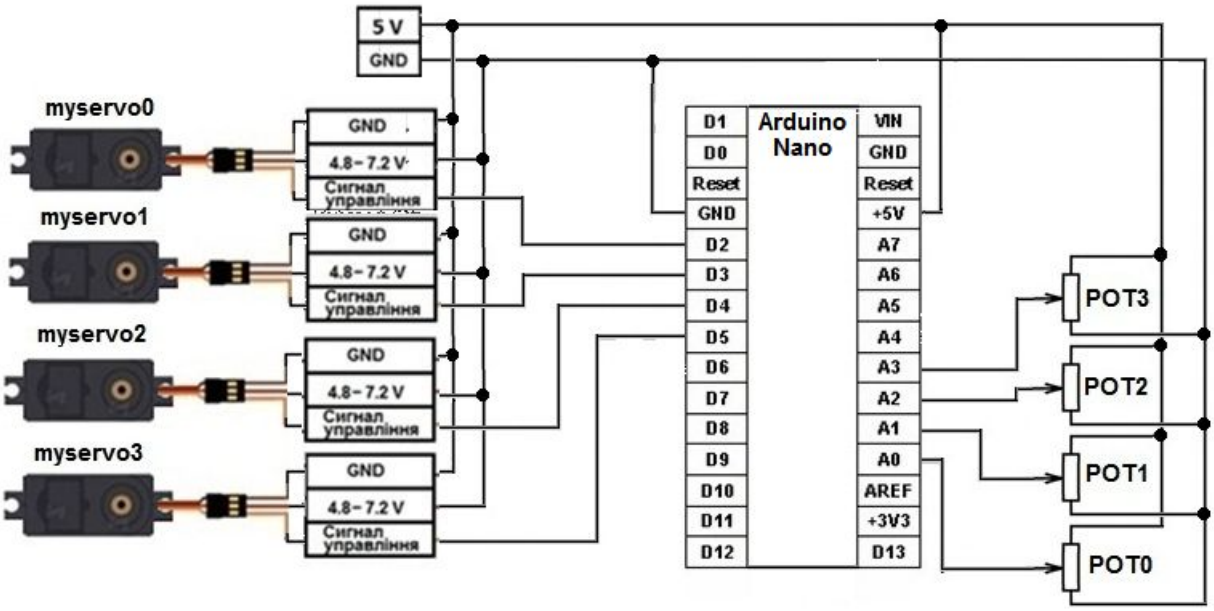


Рис. 3. Схема підключення маніпулятора та пульта керування до контролера Arduino Nano

Для перевірки значень для керування приводами val0, val1, val2, val3, що були отримані з потенціометрів, використовується функція виводу даних керування на дисплей (Монитор порта), що має такий вигляд.

```
void setup() {
…
Serial.begin(9600);  }
void loop() {
…
// значення для керування приводами
Serial.print("AO\t");//такстове повідомлення A0 та табуляція
Serial.print(val0);  //значення змінної val0
Serial.print("\t");  //табуляція
Serial.print("A1\t");
Serial.print(val1);
Serial.print("\t");
Serial.print("A2\t");
Serial.print(val2);
Serial.print("\t");
Serial.print("A3\t");
Serial.println(val3);   //значення змінної val3 та перенос строки
delay(100);  }
```

На дисплеї отримаємо значення у такому вигляді, наведеному на рис. 4.

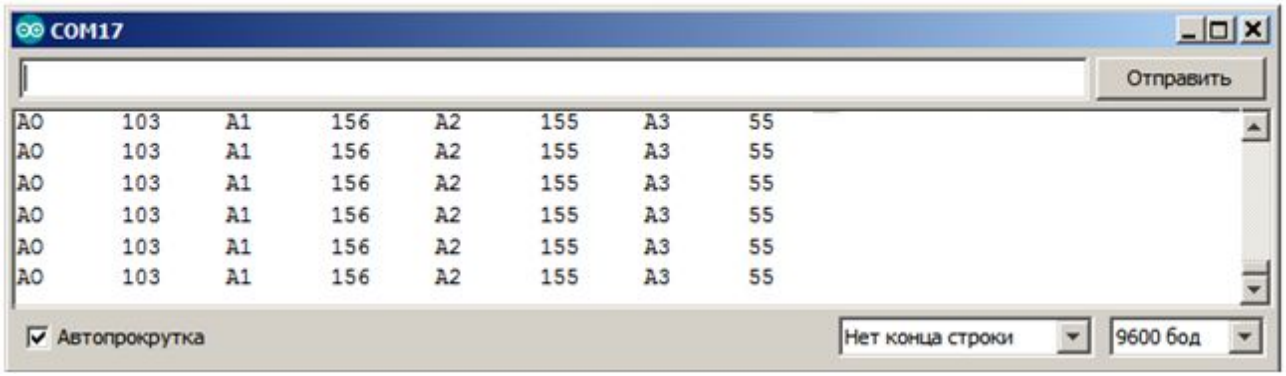


Рис. 4. Представлення даних положення окремих ланок на дисплеї

2. Приклади розв'язання задач з теми заняття.

Приклад програми керування у ручному режимі наведений у Додатку.

Завдання 1

Скласти програму керування маніпулятором в автоматичному режимі для послідовного переміщення робочого органу у визначену позицію та повернення у вихідне положення.

Позицію, в яку здійснюється переміщення визначити за допомогою ручного режиму шляхом зчитування даних з Монітора порта.

Завдання 2

Перевірити роботу програми керування маніпулятором в ручному режимі за допомогою емулятора UnoArduSim.

Контрольні запитання за темою заняття.

1. Описати, які засоби можна використовувати для ручного керування сервоприводів?
2. Визначити, як здійснюється обмеження діапазону отриманих даних?
3. Назвати, що обмежує діапазони обертання сервоприводів, які використовує маніпулятор?
4. Описати, як здійснюється керування за допомогою потенціометрів?
5. Описати, як здійснюється вивід даних на дисплей (Монитор порта)?

Заняття 5. Безконтактні датчики положення

Мета заняття: Назвати особливості використання безконтактних датчиків.
Визначити характеристики безконтактних датчиків згідно з метою використання.
Описати, як здійснити визжначення параметрів безконтактних датчиків.

1. Теоретичні положення

Безконтактні датчики визначають наявність і властивості об'єкта дистанційно без фізичного контакту.

Безконтактні датчики положення можна розділити на дискретні (цифрові) і аналогові.

У першому випадку вихідний сигнал має два стани, що визначає наявність або відсутність об'єкта, який призвів до спрацьовування датчика.

У другому випадку вихідний сигнал видає інформацію про відстань до об'єкту або про відстань, на яку перемістився об'єкт. В останньому випадку розрізняють лінійні і кутові переміщення.

За принципом визначення відстані до об'єкта безконтактні датчики діляться індуктивні, ємнісні, оптичні й ультразвукові.

Індуктивні датчики відносяться до датчиків параметричного типу. Вони можуть ставитися як до контактних, так і безконтактним датчикам, так як принцип їх дії яких заснований на зміні індуктивності обмотки з сердечником внаслідок переміщення сердечника або впливу металевого об'єкта на магнітне поле (рис. 1).

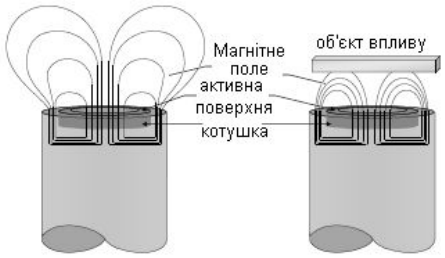


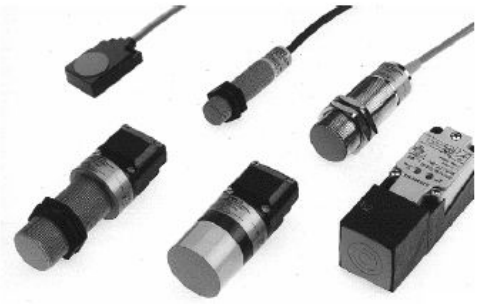
Рис. 1. Принцип дії індуктивних датчиків

В датчиках з площинними чутливими елементами впливаючий металевий елемент проходить біля котушок на заданій відстані. Зовнішній вигляд таких датчиків наведено на рис. 2, а. У датчиках із щілинними чутливими елементами (рис. 2, б) металева пластина (впливаючий елемент) проходить в щілині між котушками чутливого елемента.

а) б)

Рис. 2. Датчики з площинними (а) та щілинними (б) чутливими елементами

Ємкісні датчики відрізняються тим, що впливаючий елемент не повинен обов'язково бути металевим. Дію цих датчиків засновано на перетворенні вхідної величини у зміну ємності конденсатора (рис. 3).



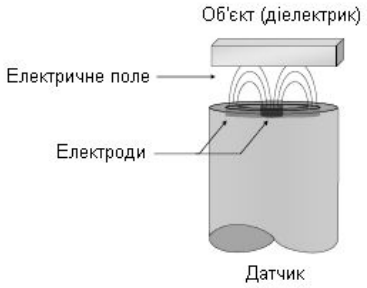
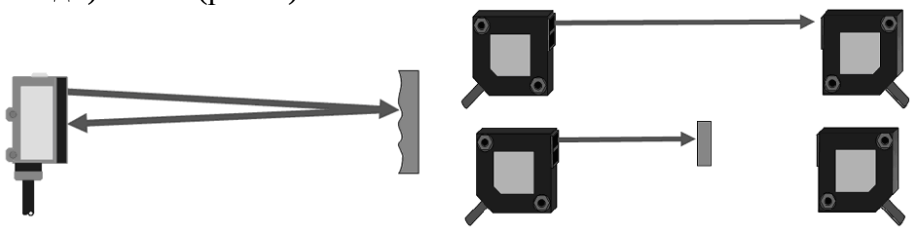


Рис. 3. Ємкісні датчики

В оптичних (фотоелектричних) датчиках зміна вихідного параметра (срум, напруга) відбувається залежно від зміни сили світла, яке падає на датчик. Оптичні датчики можуть працювати на відбивання (датчики відбиваючої дії) (рис. 4, а) або на проходження (датчики однонаправленої дії) світла (рис. 4).



а) б)

Рис. 4. Оптичні датчики на відбивання (а) та проходження (б) світла

В ультразвукових датчиках зміна вихідного параметра (струм, напруга) відбувається залежно від зміни ультразвуку, яке падає на приймач датчика. Ультразвукові датчики також можуть працювати на відбивання (датчики відбиваючої дії) або на проходження (датчики однонаправленої дії).

2. Приклад розв'язання задач з теми заняття.

Індуктивні датчики положення мають характеристику спрацьовування, що наведена на рис. 5.

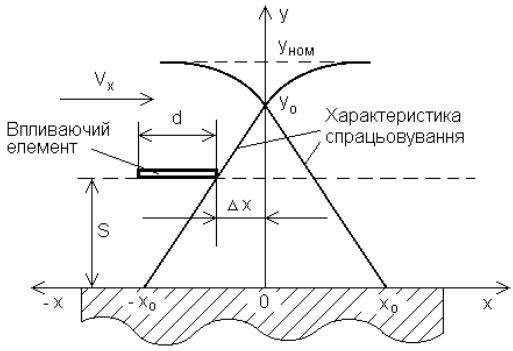


Рис. 5. Характеристика спрацьовування індуктивного датчика

Ця характеристика дає можливість знайти найменшу довжину впливаючого елемента, яка визначає точність позиціонування, в залежності від швидкості проходження та відстані від поверхні датчика до елемента впливання, що приводить до його спрацювання.

Максимальна швидкість проходження V_x, мінімальна довжина елемента впливання d, відстань S, на якому елемент впливання проходить повз датчика, відстань спрацьовування від центра датчика Δx та максимальний час спрацьовування устрою керування t_s пов'язані такими співвідношеннями:

$$\Delta x \approx \frac{x_0(y_0 - S)}{y_0}; \qquad V_x = \frac{2\Delta x + d}{t_S}; \qquad d = V_x \cdot t_s - 2 \cdot \Delta x;$$

Завдання 1

Знайти мінімальну довжину елемента впливання d та відстань спрацьовування від центра датчика Δx, що визначає помилку позиціонування, для таких значень параметрів:

Варіант	V, м/с	X_0, мм	Y_0, мм	S, мм	t_s, мс
1	1,0	5	5	3	10
2	1,5	10	10	6	15
3	2,0	15	15	10	20
4	2,5	20	20	10	15

Контрольні запитання за темою заняття.

1. Описати, які особливості мають безконтактні датчики положення?
2. Визначити, як і особливості мають індуктивні вимірювальні перетворювачі?
3. Назвати, які особливості мають ємкісні вимірювальні перетворювачі?
4. Описати яку характеристику спрацьовування мають індуктивні датчики положення?
4. Визначити, як знайти мінімальну довжину елемента впливання індуктивного датчика?

Заняття 6. Датчики переміщення та вимірювання відстані до об'єктів

Мета заняття: Назвати особливості використання датчиків переміщення та вимірювання відстані до об'єктів.
Визначити характеристики датчиків переміщення та вимірювання відстані до об'єктів згідно з метою використання.
Описати, як здійснити визначення параметрів датчиків переміщення та вимірювання відстані до об'єктів.

1. Теоретичні положення

Робота виконується з використанням ультразвукового датчика HC-SR04. Схема підключення ультразвукового датчика HC-SR04 до контролера Arduino Nano (а) та діаграми роботи (б) наведені на рис. 1.

Підключення датчика до Arduino:

- VCC на +5 V;
- TRIG на D7;
- ECHO на D8;
- GND на пін GND.

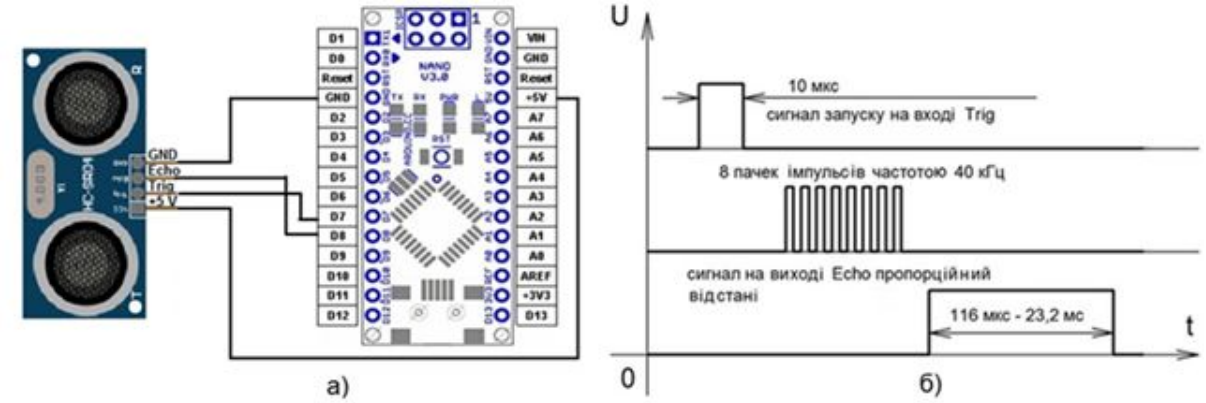


Рис. 1. Схема підключення ультразвукового датчика HC-SR04 до контролера Arduino Nano (а) та діаграми роботи (б)

Параметри датчика:

- кут огляду 15 градусів;
- відстань вимірювання від 0,03 до 5 м;
- для відстані від 0,03 до 0,6 м похибка складає 3 мм;
- для відстані від 0,6 до 5 м похибка збільшується.
- Датчик має 4 виводи:
- VCC: "+" живлення.
- TRIG (T): вивід вхідного сигналу.
- ECHO (R): вивід вихідного сигналу (Довжина сигналу залежить від відстані об'єкта до датчика).
- GND: "-" живлення (земля).

Принцип роботи датчика можна умовно розділити на 4 етапи

1. Подаємо імпульс тривалістю 10 мкс, на вивід Trig.

2. Усередині далекоміра вхідний імпульс перетворюється в 8 імпульсів частотою 40 КГц і надсилається вперед через випромінювач "Т бочечка"

3. Дійшовши до перешкоди, послані імпульси відбиваються і поступають на приймач "R бочечка", внаслідок чого отримуємо вихідний сигнал на виводі Echo, тривалість якого пропорційна відстані до об'єкту.

4. За допомогою програми контролера перетворюємо отриманий сигнал в відстань за формулою:

- тривалість імпульсу (мкс) / 58 = дистанція (см);

– тривалість імпульсу (мкс) / 148 = дистанція (дюйм).

Для вимірювання тривалості імпульсів, наприклад, у ультразвукових датчиків вимірювання відстані, використовується функція **pulseIn()**.

Ця функція зчитує тривалість позитивного (HIGH) чи негативного (LOW) імпульсу на визначеному вході (pin). Наприклад, якщо значення є високим (HIGH), pulseIn() чекає, поки стан pin зміниться на HIGH, починає підрахунок часу у мікросекундах, потім чекає, поки стан входу pin не зміниться на LOW, і зупиняє підрахунок.

pulseIn(pin, value)

pulseIn(pin, value, timeout)

де

pin - номер піна для зчитування імпульсу (int)

value - тип імпульсу для зчитування HIGH чи LOW (*int*)

(опціонально)

timeout - час очікування у мікросекундах, стандартне значення 1 секунда (*unsigned long*)

Повертає тривалість імпульсу (у мікросекундах) або 0, якщо не було повного імпульсу за час очікування (unsigned long).

2. Приклад розв’язання задач з теми заняття

У наведеному нижче скетчі дистанція відсилається в порт комп'ютера, а при дистанції менше 15 сантиметрів включається світлодіод, що підключений до піну 13.

ultrasonic-sensor

```
#define Trig 7
#define Echo 8
#define ledPin 13
void setup()  {
pinMode(Trig, OUTPUT); // вихід
pinMode(Echo, INPUT); // вхід
pinMode(ledPin, OUTPUT);
Serial.begin(9600); //задаємо швидкість послідовного каналу
}
unsigned long impulseTime=0;
unsigned long distance_mm=0;
void loop() {
digitalWrite(Trig, HIGH); // Подаємо імпульс на вхід trig
delayMicroseconds(10); // що дорівнює 10 мікросекунд
digitalWrite(Trig, LOW); // відключаємо
impulseTime=pulseIn(Echo, HIGH); // визначаємо тривалість імпульсу
distance_mm=impulseTime*10/58; // Перераховуємо в міліметри
Serial.println(distance_mm); // Виводимо на послідовний канал
if (distance_mm<150) // Якщо відстань менше ніж 15 сантиметрів
{
digitalWrite(ledPin, HIGH); // включаємо світлодіод
}
else {
digitalWrite(ledPin, LOW); // у протилежному випадку виключаємо
}
delay(100);
/* чекаємо 0,1 секунди, Наступний імпульс може бути відправленний тільки після зникнення відбитого попереднього імпульсу. Рекомендований період між імпульсами повинен бути не менше 50 мс.*/
}
```

Завдання 1

Змінити наведену програму вимірювання відстані до об'єкту згідно таких значень контактів для підключення сигналів `Trig` та `Echo`, та відстані, що визначається шляхом включення світлодіода.

Варіант	Контакт підключення сигналу `Trig`	Контакт підключення сигналу `Echo`	Відстань, см
1	1	2	50
2	4	6	70
3	7	8	20
4	9	10	90

Завдання 2

Перевірити роботу програми керування маніпулятором в ручному режимі за допомогою емулятора UnoArduSim. Для імітації роботи ультразвукового датчика HC-SR04 використовувати інструмент Одна дія ('1SHOT'), наведений у Додатку.

Контрольні запитання за темою заняття.

1. Описати, які засоби можна використовувати для визначення відстані?
2. Назвати, що обмежує діапазони визначення відстані ультразвукових датчиків?
3. Описати, як здійснюється визначення відстані ультразвукових датчиків?
4. Описати функцію, що здійснює вимірювання тривалості імпульсів?
5. Описати, як здійснюється перетворення тривалості отриманого сигналу в відстань?

Заняття 7. Мікропроцесорні пристрої обробки вимірювальної інформації

Мета заняття: Назвати особливості використання мікропроцесорних пристроїв для обробки вимірювальної інформації.
Визначити алгоритми обробки вимірювальної інформації згідно з метою використання.
Описати, як створити програми обробки вимірювальної інформації.

1. Теоретичні положення

На рис. 1 наведено підключення ультразвукового датчика та двох двигунів постійного струму мобільного робота з диференційним приводом, що переміщується з об'їздом перешкод, які визначає ультразвуковий датчик. При визначенні перешкоди на відстані менш як 30 см робот зупиняється та починає виконувати розворот на місці, поки не зникне перешкода. Після зникнення перешкоди рух вперед продовжується.

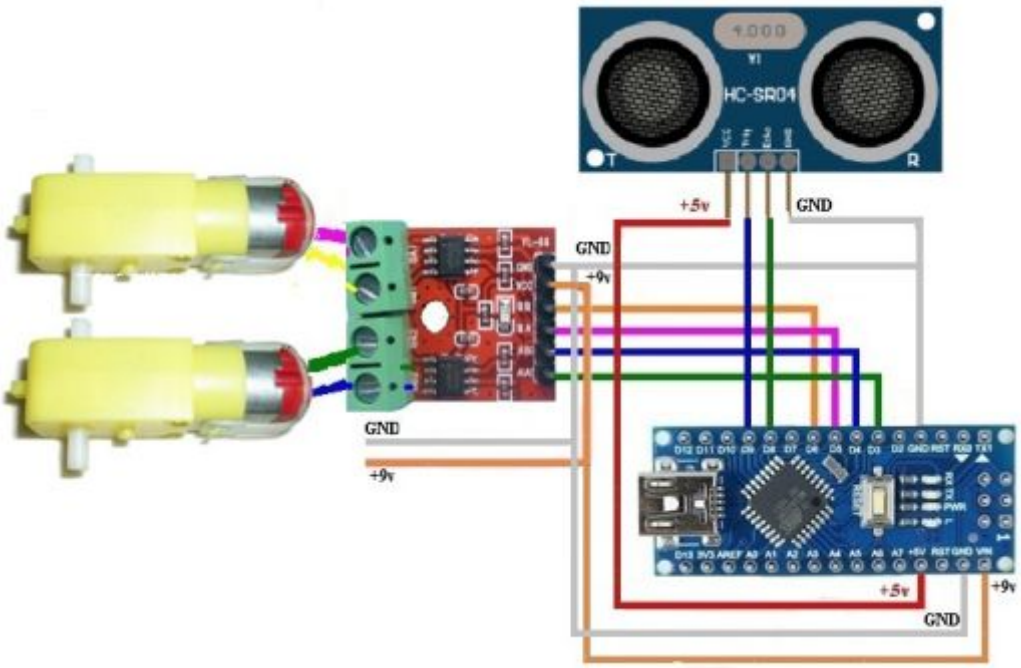


Рис. 7.1. Підключення ультразвукового датчика та двох двигунів постійного струму мобільного робота

2. Приклад розв’язання задач з теми заняття

Програма визначення перешкод та зупинкою при наявності перешкоди з використанням двигунів постійного струму DC MOTOR емулятора UnoArduSim.

```
#define Trig 7
#define Echo 8
#define Pwm1 3
#define Dir1 2
#define Pwm2 9
#define Dir2 5
#define ledPin 13
void setup() {
pinMode(Trig, OUTPUT); // вихід
pinMode(Echo, INPUT); // вхід
pinMode(ledPin, OUTPUT);
Serial.begin(9600); //задаємо швидкість послідовного каналу
```

```
pinMode(Dir1, OUTPUT);
pinMode(Dir2, OUTPUT);
pinMode(Pwm1, OUTPUT);
pinMode(Pwm2, OUTPUT);
digitalWrite(Dir1, LOW);
digitalWrite(Dir2, LOW);
}
unsigned long impulseTime=0;
unsigned long distance_mm=0;
void loop() {
digitalWrite(Trig, HIGH); // Подаємо імпульс на вхід trig
delayMicroseconds(10); // що дорівнює 10 мікросекунд
digitalWrite(Trig, LOW); // відключаємо
impulseTime=pulseIn(Echo, HIGH); // визначаємо тривалість імпульсу
distance_mm=impulseTime*10/58; // Перераховуємо в міліметри
Serial.println(distance_mm); // Виводимо на послідовний канал
if (distance_mm<150) // Якщо відстань менше ніж 15 сантиметрів
{
digitalWrite(ledPin, HIGH); // включаємо світлодіод
digitalWrite(Pwm1, LOW);
digitalWrite(Pwm2, LOW);
}
else
{
digitalWrite(ledPin, LOW); // у протилежному випадку виключаємо
analogWrite(Pwm1, 200);
analogWrite(Pwm2, 200);
}
delay(100);
/* чекаємо 0,1 секунди, Наступний імпульс може бути відправленний тільки після зникнення відбитого попереднього імпульсу. Рекомендований період між імпульсами повинен бути не менше 50 мс.*/
}
```

Завдання 1

Скласти програму для переміщення мобільного робота з об'їздом перешкод для контролера Arduino.

Завдання 2

Перевірити роботу програми керування маніпулятором в ручному режимі за допомогою емулятора UnoArduSim.

Контрольні запитання за темою заняття.

1. Описати, як визначити структуру та склад мікропроцесорних пристроїв у залежності від задачі обробки інформації?
2. Описати, які засоби програмування мікропроцесорних пристроїв використовують при проектуванні систем обробки інформації?
3. Описати, як для масштабування результатів вимірювання?
4. Описати, як здійснити пересування мобільного робота з диференційним приводом вперед по прямій?
5. Описати, як здійснити поворот мобільного робота з диференційним приводом на місці?

Контрольні завдання

Контрольне завдання 1. Розрахунок механічних компонент мехатронних систем.

Контрольне завдання 2. Розрахунок основних параметрів електроприводів.

Контрольне завдання 3. Розрахунок основних параметрів датчиків внутрішньої інформації.

Контрольне завдання 4. Розробка алгоритмів систем керування мехатронних пристроїв.

ЛІТЕРАТУРА

1. Мехатроніка. Конспект лекцій для студентів бакалаврів, спеціальність: 131 - Прикладна механіка, спеціалізація: Мехатроніка та промислові роботи, спеціалізація: Інженерія логістичних систем, спеціальність: 133 – Галузеве машинобудування, спеціалізація: Підйомно-транспортні, дорожні, меліоративні машини і обладнання / Укладачі: Семенюк В.Ф., Михайлов Є. П. Одеса: ОНПУ, 2017. 74 с. Рег. ном. КЛ07854 14.02.17 №4257-РС-2017 Електронна Інтернет-версія.

2. Блум Джереми. Изучаем Arduino: инструметы и методы технического волшебства: Пер. с англ. - СПб.: БХВ-Петербург, 2015. - 336 с.: ил. (електронна версія)

3. Методичні вказівки до лабораторних робіт з дисципліни "Маніпулятори та промислові роботи" розділ "Інформаційні системи промислових роботів. Датчики внутрішньої інформації" для студентів спеціальностей 6.0505 – “Інженерна механіка” денної форми навчання / Укл.: *Є.П.Михайлов, О.Б.Кнюх*. - Одеса: ОНПУ, 2010. – 27с. МВ03477 15.02.10, №0096-РС-2010 Електронна Інтернет-версія.

ІНФОРМАЦІЙНІ РЕСУРСИ

4. Download the Arduino IDE URL: https://www.arduino.cc/en/Main/Software

5. UnoArduSim: https://www.sites.google.com/site/unoardusim/home

Додаток

Створення та перевірка програм для контролера Arduino за допомогою емулятора UnoArduSim

Додаток UnoArduSim є безкоштовним емулятором (симулятором) контролера Arduino, який дає можливість здійснити виконання програми в реальному часі без наявності самої плати Arduino.

При цьому є можливість перегляду ходу виконання програми.

UnoArduSim призначений для полегшення налагодження Arduino програм і містить набір віртуальних пристроїв вводу / виводу ('I / O' Пристроїв), які можна налаштовувати і підключати до віртуального Arduino.

UnoArduSim не потребує установки. Його треба завантажити за посиланням https://www.sites.google.com/site/unoardusim/home, після чого запустити файл UnoArduSim.exe і працювати з ним.

Симулятор працює в операційній системі Windows починаючи з Windows XP.

Після завантаження UnoArduSim для спрощення запуску можна створити ярлик на робочому столі.

Після запуску програми з'являється вікно (рис. 1) з вихідним прикладом програми:

```
/* This is a default program--
Use File->Load Prog to load a different program
*/
int count;
void setup()
{
count=0;
}
void loop()
{
```

```
count=count+1;
delay(100);
}
```

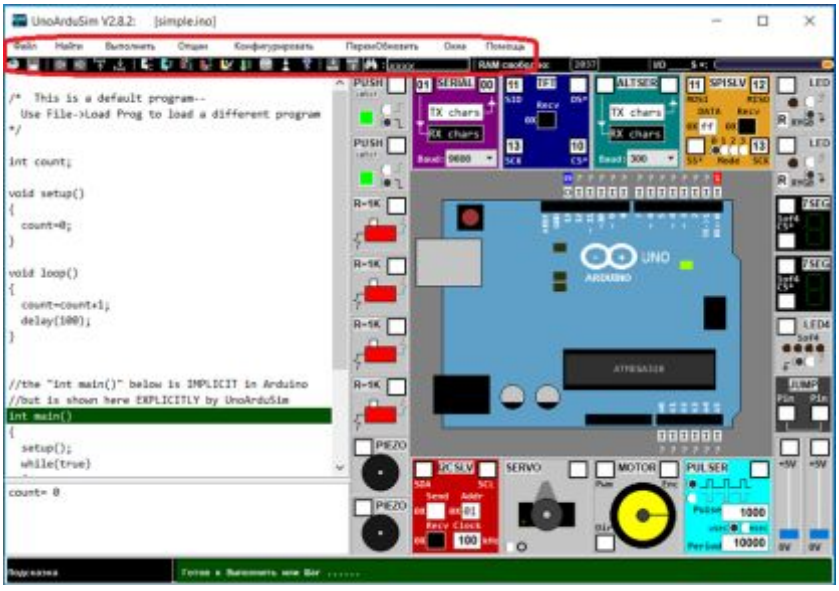


Рис. 1. Вихідне вікно

Подвійним кліком зони, де наведена програма, запускається вікно редагування Arduino програм (Edit / View Program), яке наведено на рис.2.

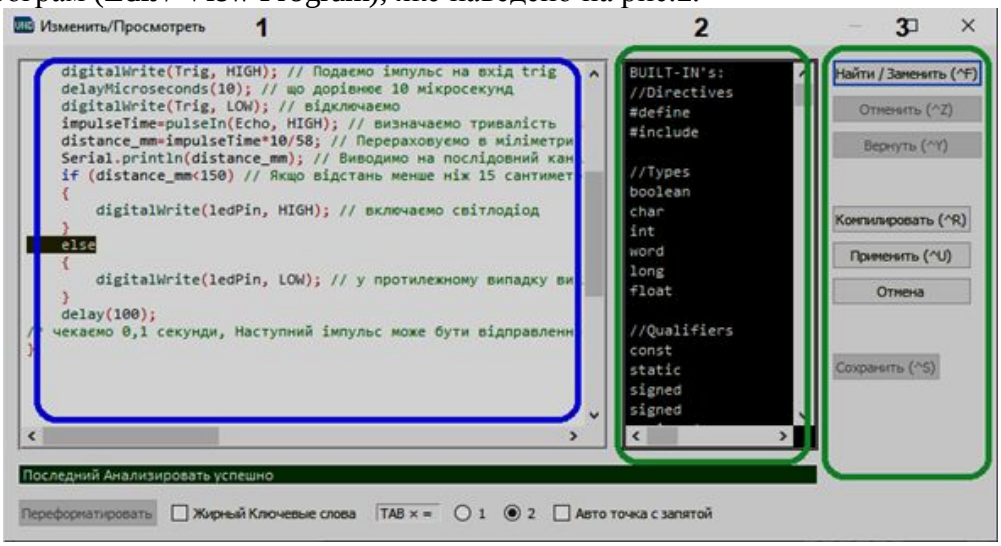


Рис. 2. Вікно редагування програми

Зона 1 призначена для внесення та редагування Arduino програм (скетчів). Вона містить невеликий код для емуляції скетчу, цей код можна видалити, потім він додається автоматично. Тут можна вставити свій код або скопійований з програми Arduino IDE.

Права кнопка мишки тут не працює, так що копіювати і вставляти доводиться через команди Control C і Control V.

Зона 2 містить каталог мови Ардуіно, в якому наведені команди та функції, який працює наступним чином – треба встановити курсор в потрібному місці в зоні 1 і двічі клікнути на потрібну функцію в зоні 2.

Зона 3 включає кнопки:

Найти / Заменить (^ H) - для виклику пошуку у програмі,

Отменить (^ Z) **Вернуть** (^ Y) - крок назад чи крок вперед,

Компилировать (^ R) - скомпілювати (перевірити на наявність помилок),

Применить (^ U) - прийняти,

Отмена - скасування,

Сохранить (^ S) - зберегти.

На рис. 3 наведене вікно змінних.

Тут під час емуляції можна спостерігати стан і значення всіх змінних, які містяться в скетчі.

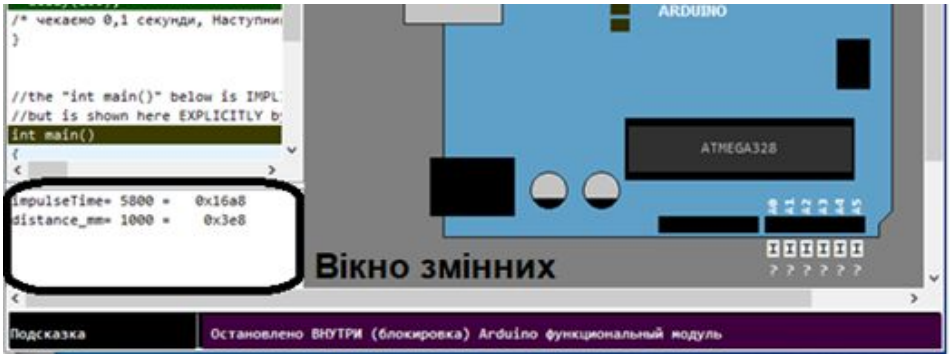


Рис. 3. Вікно змінних

Скорочену та повну інформацію про UnoArduSim можна знайти у закладці «Помощь» - « Быстрый помощь», «Полный помощь».

На рис. 4 наведений приклад виконання програми, що здійснює миготіння контакту 13 та підрахунок кількості миготінь.

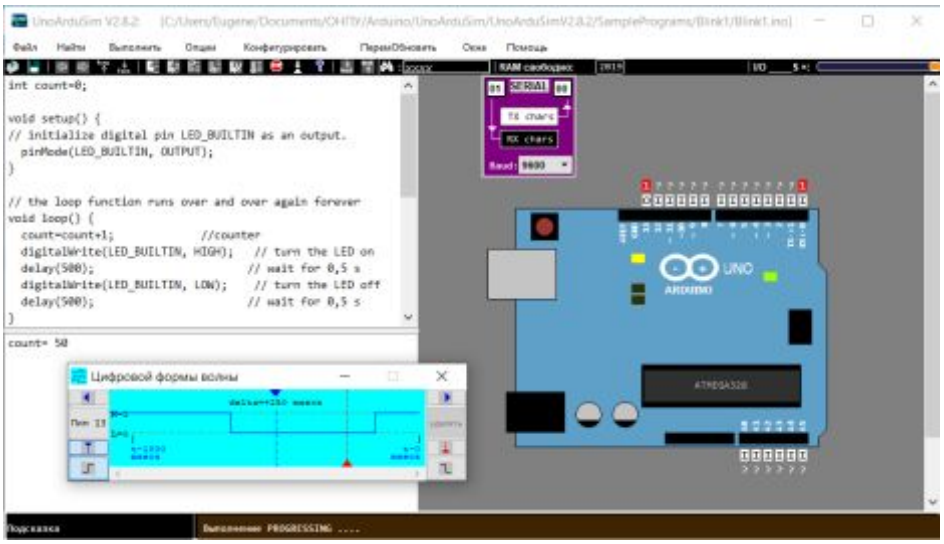


Рис. 4. Відображення виконання програми на лабораторній панелі

На лабораторній панелі можна спостерігати стан контактів віртуального Arduini UNO.

Знак питання означає що контакт не використовується, нуль на синьому фоні це сигнал низького рівня LOW, одиниця на червоному фоні це сигнал високого рівня HIGH, стрілка вгору на рожевому фоні, це сигнал ШІМ. На платі є функціонуючі світлодіоди: ON, RX, TX і pin 13.

Також, якщо клікнути лівою кнопкою по активному контакту, то запуститься вікно цифрового осцилографа (видає значення 0 та 1).

Права кнопка запускає аналоговий осцилограф, що дозволяє проглянути сигнали на аналогових входах.

Синя і червона мітки параметра delta виставляються шляхом перетягування їх мишкою, а параметр t (час) налаштовується коліщатком мишки.

Змінна count у вікні змінних показує кількість миготінь.

Додаток UnoArduSim зберігається у папці UnoArduSimV2.8.2 (остання версія).

У папці translations знаходяться переклади файлів допомоги UnoArduSim_FullHelp та UnoArduSim_QuickHelp на різні мови. (російською мовою UnoArduSim_QuickHelp_ru.pdf та UnoArduSim_FullHelp_ru.pdf).

У папці SamplePrograms знаходяться приклади програм. Цю папку можна використовувати для зберігання своїх програм у папках з відповідними назвами.

Для перевірки отриманих даних можна використовувати Монитор ‘SERIAL’, який дає можливість введення та відображення даних (рис. 5).

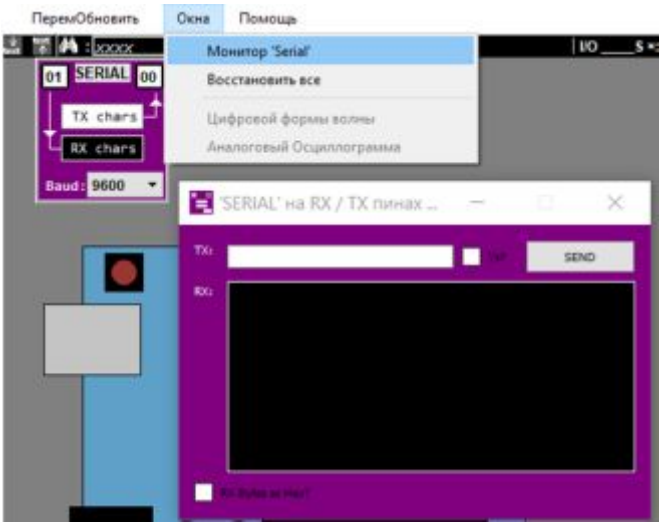


Рис. 6. Монитор ‘SERIAL’,

Монитор ‘SERIAL‘ (вікно ‘SERIAL‘ на RX / TX пинах…) можна визвати подвійним кліком на інструмент ‘SERIAL’, або через закладку «Окна» - «Монитор ‘SERIAL’».

Справа на додатку UnoArduSim в запущеному стані розташована лабораторна панель з різними інструментами розташованими по периметру, які можна налаштовувати і підключати до віртуального Arduino UNO.

Всі ці інструменти можна додавати, вказавши потрібну кількість у вікні I/O Устройства, яке знаходиться у вкладці **Конфигурировать – 'I/O' Устройства** (рис. 6).

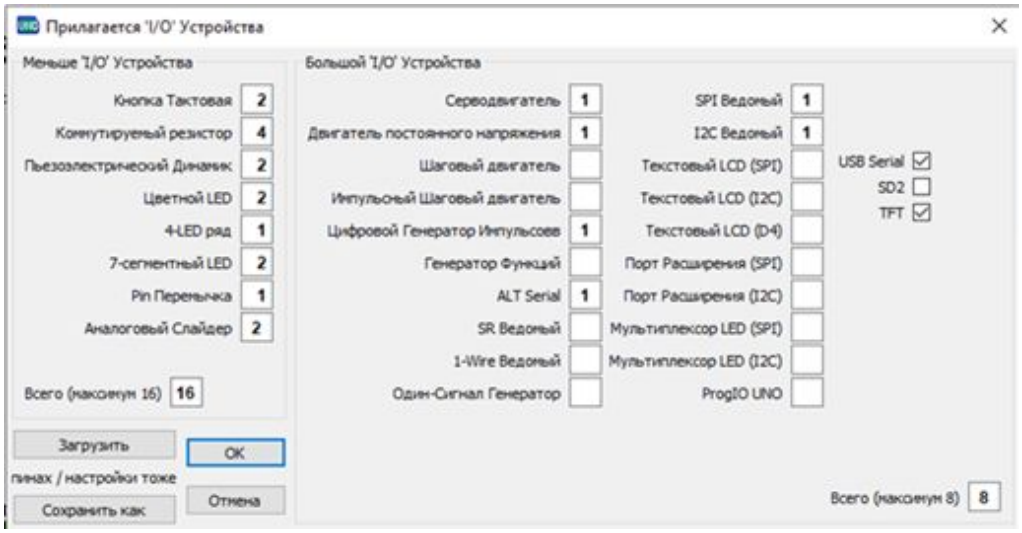


Рис. 6. Вкладка **Конфигурировать – 'I/O' Устройства**

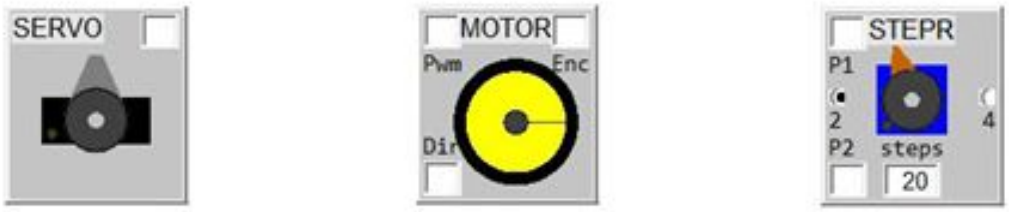
Видалити інструменти можна видаливши ці значення у вікні кількості.

Детальну інформацію про інструменти можна знайти у вказаних закладках «Быстрый помощь», «Полный помощь».

Для збереження налагодження треба натиснути кнопку «Сохранить как» та вказати папку, де знаходиться відповідна програма.

Розглянемо деякі інструменти.

У наявності є три модулі моторів (рис.7), а саме, серводвигуни, двигуни постійного струму та крокові двигуни.



а) б) в)

Рис. 7. Серводвигун (а), двигун постійного струму (б), кроковий двигун (в)

Серводвигун SERVO, що наведений на рис. 7, а, управляється ШІМ сигналом, в залежності від шпаруватості ШІМ сигналу вал приймає певний кут від 0 до 180 градусів. Використовується, коли треба здійснити поворот на певний кут.

Двигун постійного струму DC MOTOR наведений на рис. 7. б.

Цей інструмент вводу-виводу здійснює емуляцію електродвигуна постійного струму.

Швидкість обертання встановлюється за допомогою сигналу з широтно-імпульсною модуляцією, що подається на вхід Pwm, напрямок обертання встановлюється на вході Dir.

На вихід Enc поступають сигнали з умонтованого енкодера (імпульсного датчика переміщення), що видає 8 імпульсів на одне обертання.

Кроковий двигун STEPR (Stepper) наведений на рис. 7, в. У вікні steps виставляється кількість кроків на один оборот ротора. Є два варіанти: двохфазний і чотирьохфазний які перемикаються шляхом установки галочки над двійкою або над четвіркою. В даний момент активний двофазний варіант з підключенням до контактів P1 і P2, якщо вибрати чотирьохфазний то відповідно додадуться ще два контакти P3 і P4. Обертається за рахунок зміни полярності, при кожній зміні полярності відбувається переміщення ротора на один крок.

На рис. 8 наведені інструменти для введення цифрових та аналогових даних.

а) б) в)

Рис. 8. Кнопка (а), повзунковий потенціометр (б), перемикач (в)

Кнопка PUSH (рис. 8, а). На рисунку кнопка має на виході одиницю, а при натисканні на неї на виході буде нуль. Якщо переставити галочку вище то на виході буде нуль, а при натисканні буде одиниця, тобто навпаки.

Повзунковий потенціометр (рис. 8, б). Управляється шляхом перетягування повзунка мишкою (вгору-вниз). На виході формується аналоговий сигнал від 0 до 5 вольт, в програмному вигляді це значення від 0 до 1023.

Перемикач R = 1K (рис. 8, в) представлє собою подтягіваюшій резистор з номіналом 1 Ком, який за замовчуванням підключений до мінуса. Якщо клікнути на нього то він підключиться до плюса.

Інструмент Одна дія ('1SHOT'), наведений на рис. 8.

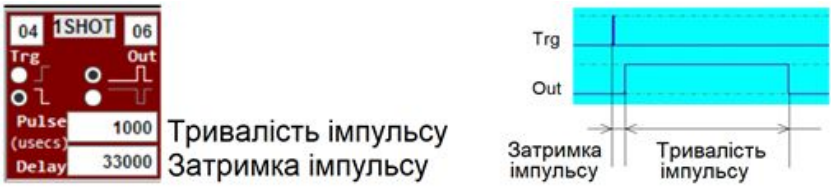


Рис. 8. Інструмент Одна дія ('1SHOT'

Цей інструмент вводу-виводу здійснює емуляцію цифрової одноразової дію, яка може генерувати імпульс обраної полярності та тривалості Pulse на контакті "Out".

Імпульс виникає після заданої затримки Delay від фронту запуску, який надходить на вхідний контакт Trg (тригер).

Параметри Pulse та Delay надаються у мікросекундах.

Приклади мехатронних пристроїв

На рис. 9, рис. 10 та рис. 11 наведений приклад програми керування серводвигуном, двигуном постійного струму та кроковим двигуном за допомогою повзункового потенціометра, що встановлює кут повороту серводвигуна та швидкість обертання двигуна постійного струму. Перемикач встаонлює напрямок обертання двигуна постійного струму.

Вигляд основного вікна та конфігурація наведені, відповідно на рис 9 та рис. 10.

На рис. 11 наведена програма керування двигунами.

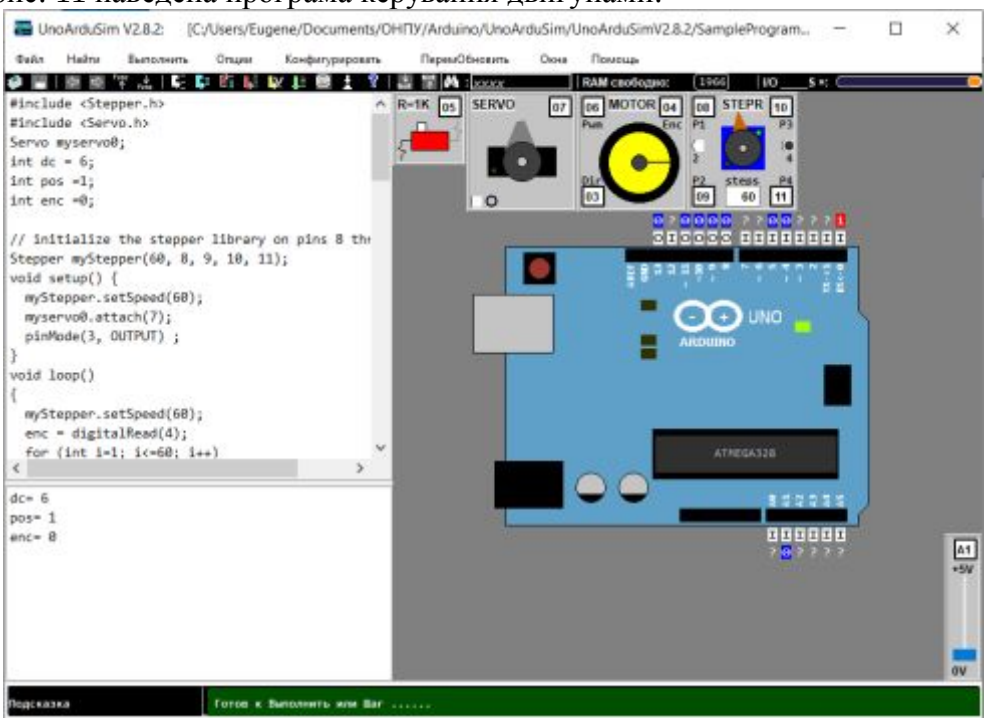


Рис. 9. Вигляд основного вікна

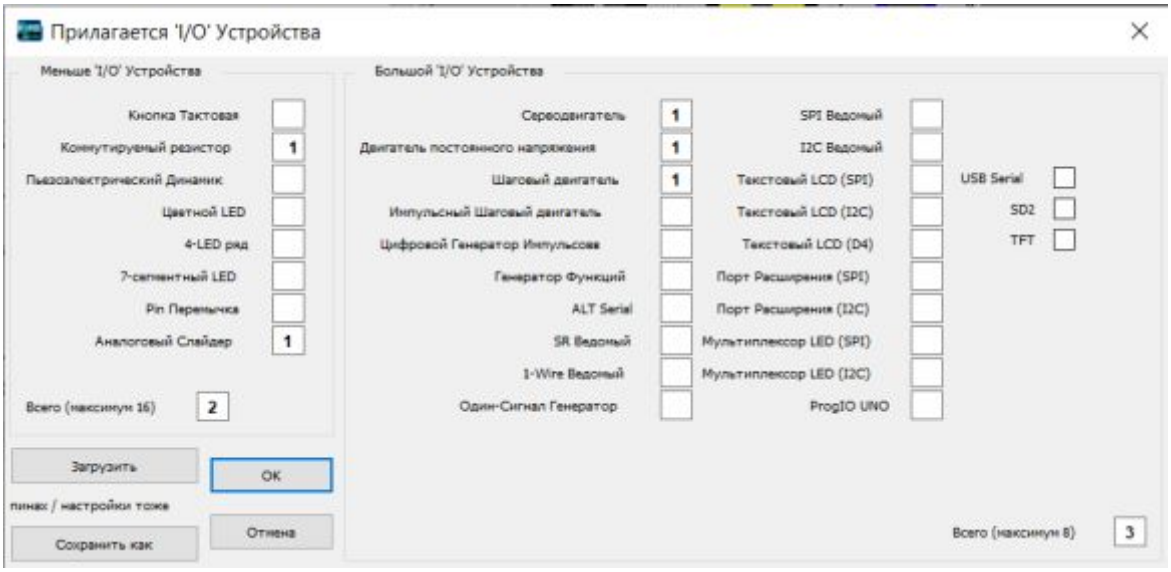


Рис. 10. Конфігурація присрою

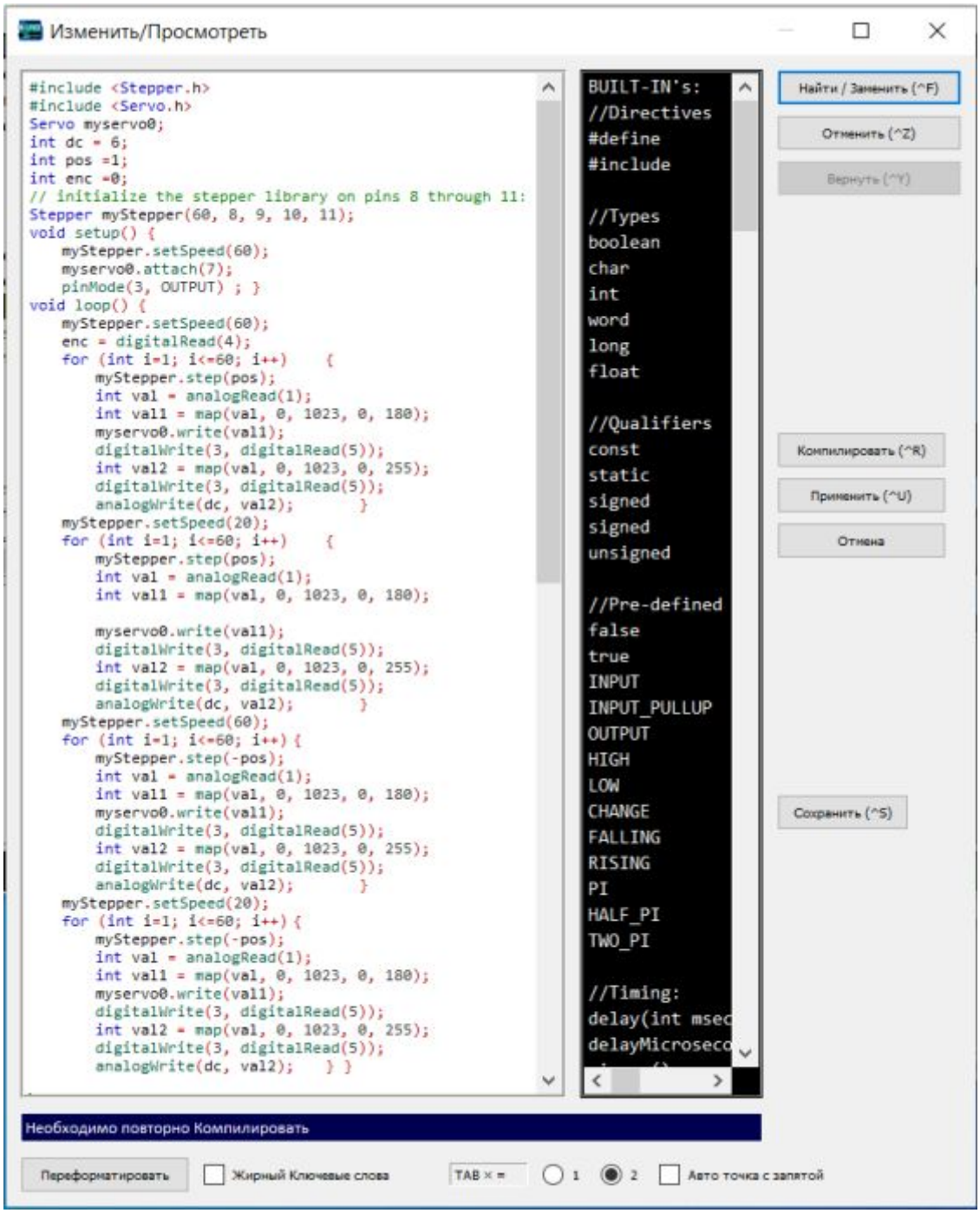


Рис. 11. Програма керування двигунами