Raport Końcowy

Rodzaj zajęć i nazwa przedmiotu

Projekt z Systemów Mikroprocesorowych i Wbudowanych

Rok akademicki	Miasto	Tryb	Kierunek	Semestr	Prowadzący	Grupa	Sekcja
2023/24	G	S	INF	5	GB	5	7

Planowany termin wykonywania ćwiczenia		Faktyczny termin wykonania ćwiczenia	
Data	Godzina	Data	Godzina
2024.02.18			

Temat ćwiczenia

Robot jeżdżący “Aria”

Skład sekcji		
	Imię i nazwisko	Uwagi
1	Joanna Wiekiera	
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		

1. Temat projektu

Aria – robot jeżdżący

2. Założenia

Celem projektu było stworzenie jeżdżącego robota. Urządzenie jest w stanie omijać wykryte przez czujniki przeszkody (np. ściany, położone obiekty). Robot porusza się z wykorzystaniem mechanicznych kół. Wykorzystany został komponent *Arduino Uno* oraz czujniki optyczne.

3. Analiza zadania

Do skonstruowania robota jeżdżącego możliwe było użycie różnych komponentów elektronicznych. W trakcie analizy zadania zdecydowałam się na niżej wymienione elementy.

3.1. Mikrokontroler

Do projektu wykorzystałam moduł *Arduino Uno Rev3* z mikrokontrolerem *AVR ATmega328*. Mój wybór wynika z dużej bazy wiedzy dostępnej w Internecie, dostępnych bibliotek oraz faktu, że Arduino jest odpowiedni do projektów tego rozmiaru.

3.2. Zasilanie

Wybranym przeze mnie sposobem zasilania jest podłączanie szeregowo sześciu *baterii AA 1,5V* – co daje łącznie w przybliżeniu 9V. Alternatywne użycie jednej baterii 9V skutkuje krótszą pracą układu z powodu jej małej pojemności.

3.3. Czujnik ultradźwiękowy

Do implementacji funkcjonalności omijania przeszkód wykorzystane zostały *czujniki* ultradźwiękowe *HC-SR04*. Na korzyść wymienionego czujnika przemawiała jego atrakcyjna cena, dostępność i dobre opracowanie merytoryczne dostępne w zasobach Internetowych.

3.4. Napęd

Robot porusza się z wykorzystaniem dwóch silników prądu stałego z przekładniami napędzających koła. Komponenty wybrane zostały z powodu odpowiednich wymiarów. Robot wyposażony jest także w trzecie koło (umieszczone na przodzie urządzenia), jednak nie posiada napędu, dlatego nie zostało uwzględnione w analizie.

3.5. Sterownik silników

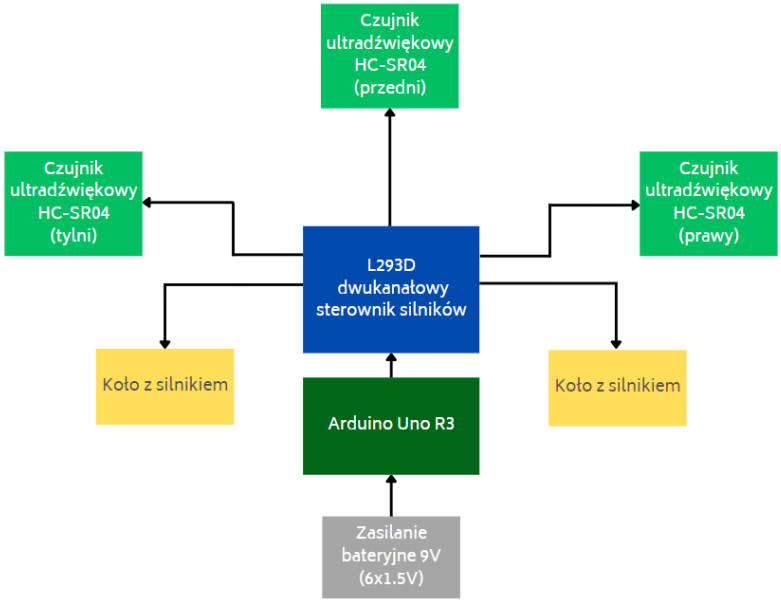
Do sterowania silnikami kół wykorzystałam dwukanałowy sterownik silników *L293D*. Układ pozwala na regulację kierunku i prędkości obrotowej silników o poborze prądu do 600mA i napięciu zasilania do 36V. Nie wybrałam proponowanego w Internecie *Forbot Robot Shield dla Arduino* z powodu bardzo dużej różnicy w cenie. Rozważałam także użycie dwukanałowego sterownika silników *L298N – moduł WB291111 – Iduino ST1112*, jednak nie zdecydowałam się na niego z powodu mniejszej dostępności materiałów dydaktycznych.

4. Zmiany dokonane podczas realizacji projektu

Podczas tworzenia projektu zdecydowałam się na odejście od pomysłu wykorzystania czujników światła. Robot początkowo miał być w stanie pokonywać labirynt narysowany na planszy, jednak z uwagi na plany rozbudowy funkcjonalności robota o umiejętność parkowania równoległego, zdecydowałam się na wykorzystanie wyłącznie czujników ultradźwiękowych.

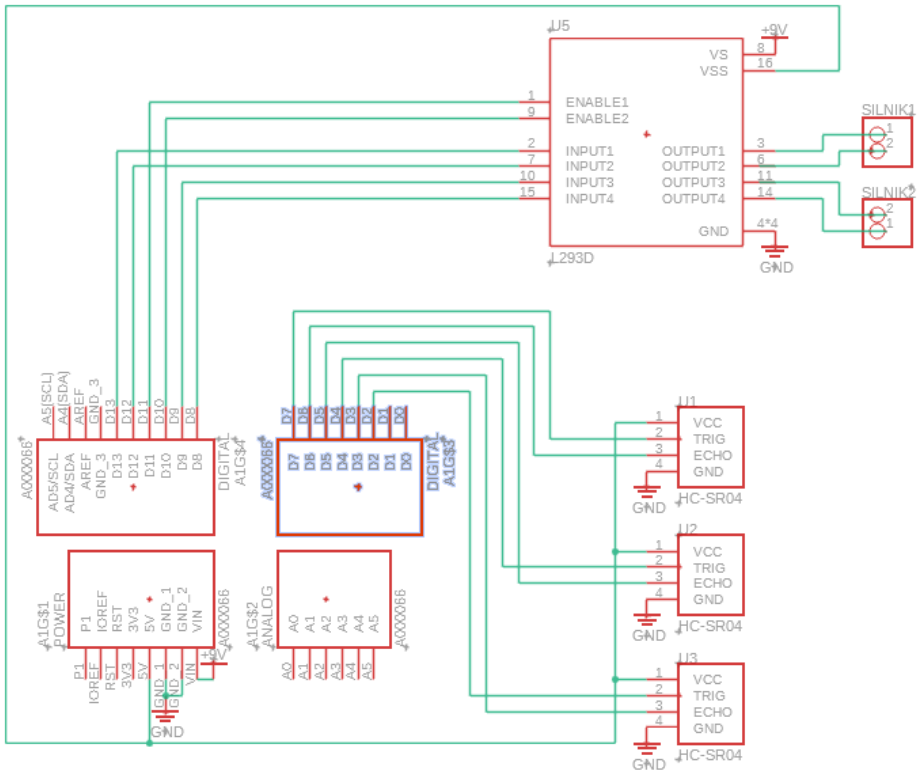
5. Specyfikacja wewnętrzna

a) schemat blokowy urządzenia



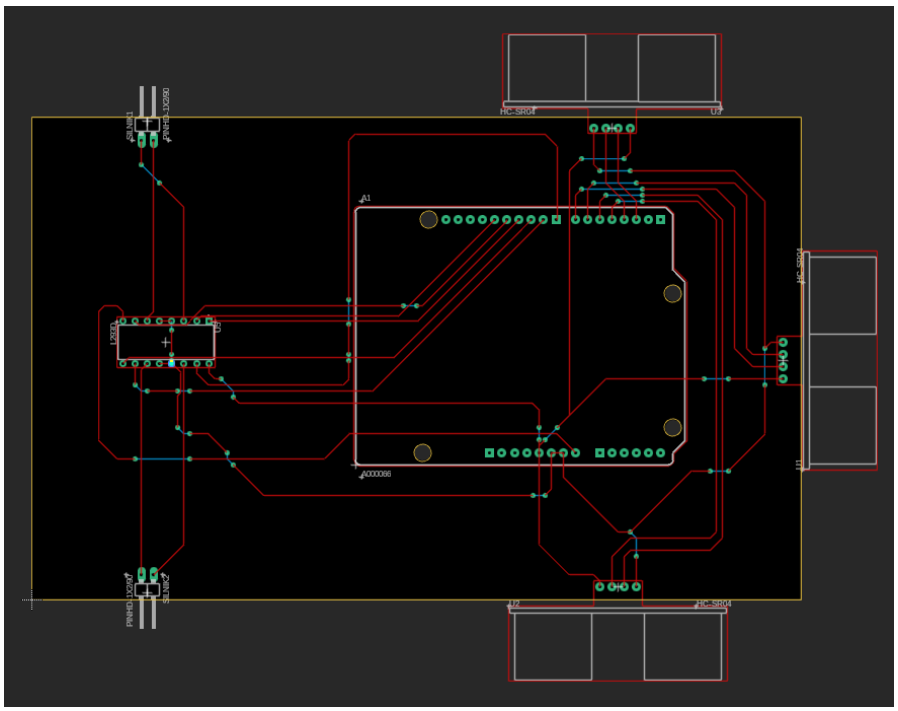
rys. 5.1. Schemat blokowy urządzenia

b) schemat ideowy urządzenia

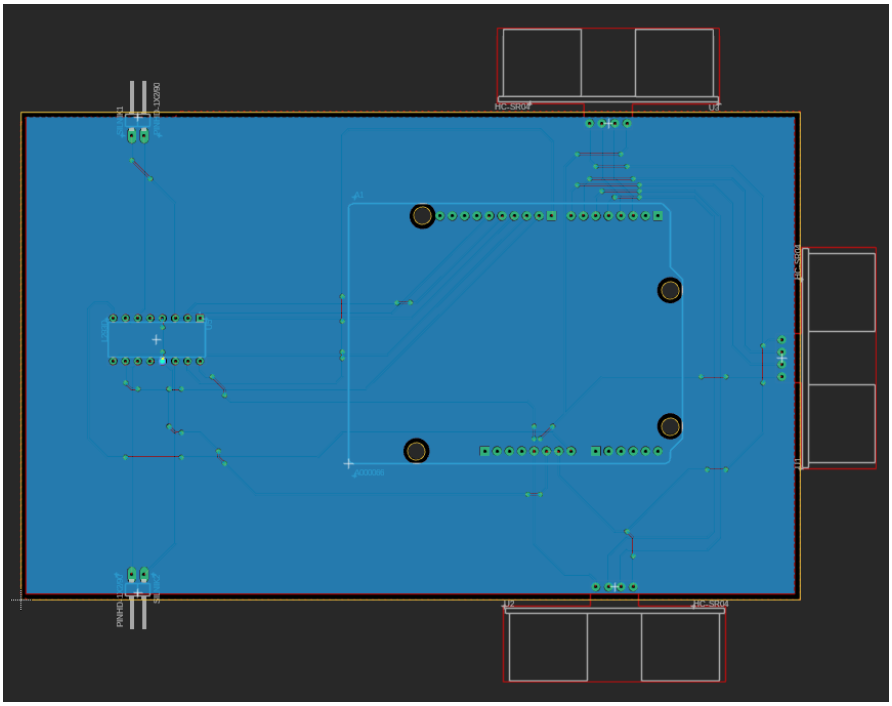


rys. 5.2. Schemat ideowy urządzenia

c) schemat montażowy



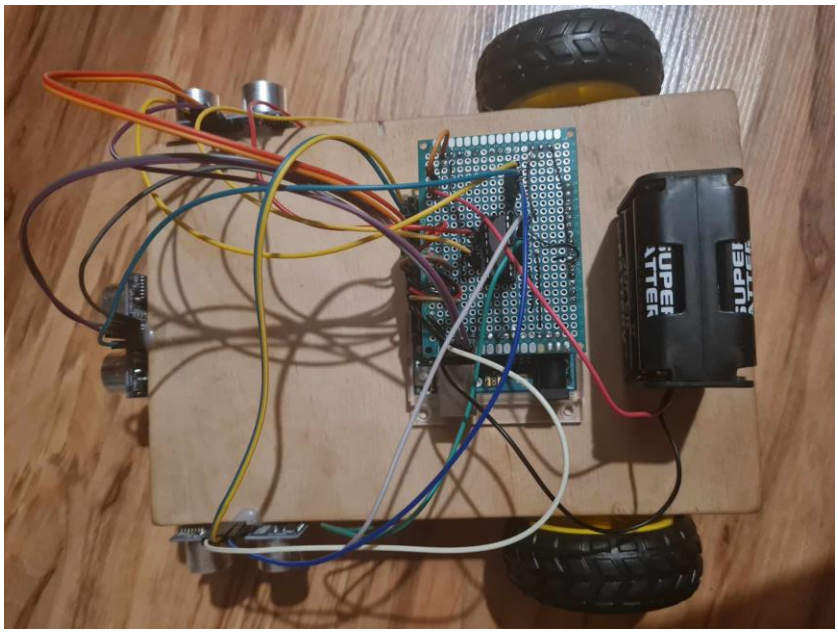
rys. 5.3. Schemat montażowy urządzenia



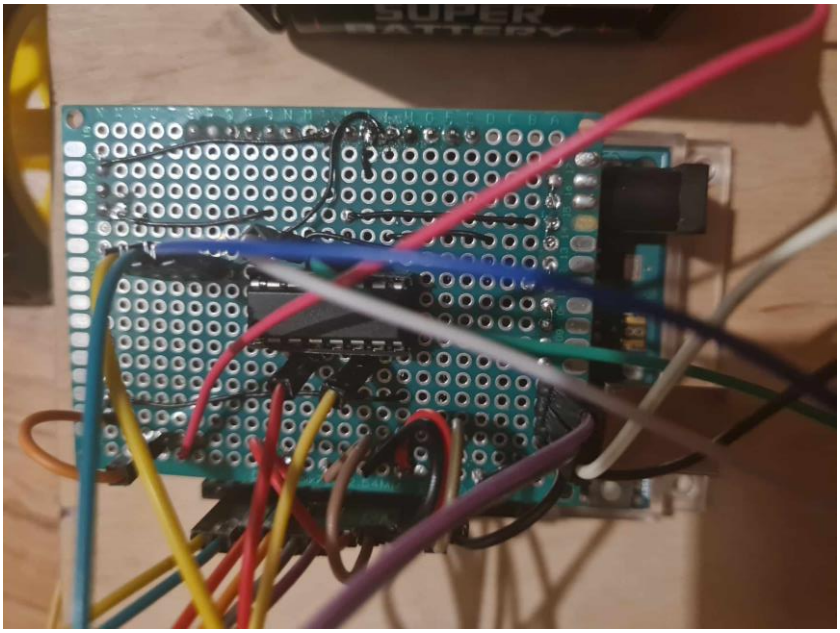
rys. 5.4. Schemat montażowy urządzenia z wylanymi warstwami

d) montaż układu

Do montażu układu wykorzystano płytkę uniwersalną, na której umieszczono sterownik silników *L293D.* Płytka umieszczona została na module *Arduino Uno* z wykorzystaniem przylutowanych goldpinów. Wszystkie elementy elektroniczne i mechaniczne umieszczone zostały na drewnianej platformie, a czujniki z modułem *Arduino Uno* połączone zostały z wykorzystaniem przewodów żeńsko-męskich.



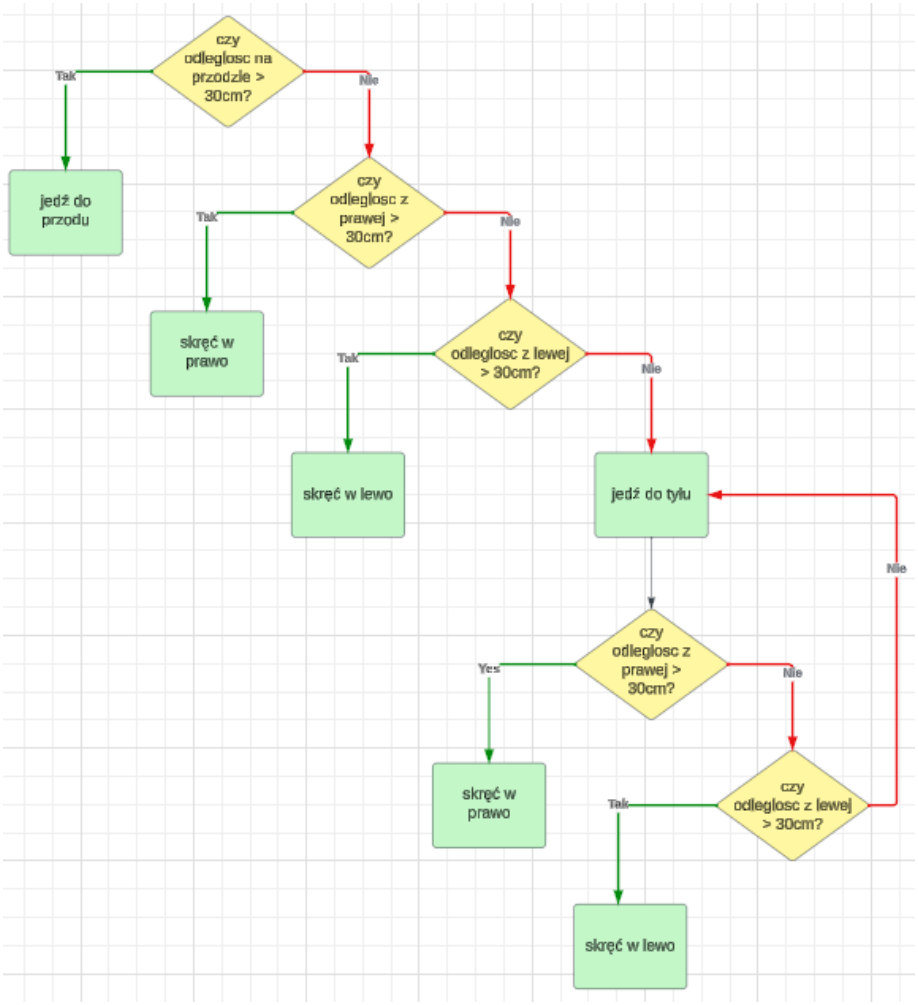
rys. 5.5. Montaż układu – widok na całego robota z góry



rys. 5.5. Montaż układu – widok na płytkę z góry

6. Algorytm wykorzystany do omijania przeszkód

Algorytm omijania przeszkód przedstawiony został na poniższym diagramie. W przypadku napotkania przeszkody, sprawdzane jest czy kolejno z prawej i lewej strony nie ma przeszkody. Jeśli robot może skręcić w jedną ze stron, robi to.



rys. 6.1. Schemat blokowy algorytmu wykonywanego w przypadku napotkania przeszkody

7. Kod programu

Poniżej zamieszczony został kod programu zaimplementowanego w robocie.

```
#define inputRight1 12 //pierwszy pin determinujacy kierunek prawego koła (HIGH - do przodu)
#define inputRight2 13 //pierwszy pin determinujacy kierunek prawego koła (HIGH - do tyłu)
#define enablePinRight 11 // PWM prawego koła (nadawanie prędkości obrotowej prawego koła)

#define inputLeft1 8 //pierwszy pin determinujacy kierunek lewego koła (HIGH - do przodu)
```

```
#define inputLeft2 9 //pierwszy pin determinujacy kierunek lewego koła (HIGH - do tyłu)
#define enablePinLeft 10 // PWM lewego koła (nadawanie prędkości obrotowej lewego koła)

#define trigPinFront 7 //wyjście odpowiadające wejściu zezwalającemu przedmiego czujnika
#define echoPinFront 6 //wejście odpowiadające wyjściu przedniego czujnika zwracającego pomiar

#define trigPinRight 4 //wyjście odpowiadające wejściu zezwalającemu prawego czujnika
#define echoPinRight 5 //wejście odpowiadające wyjściu prawego czujnika zwracającego pomiar

#define trigPinLeft 2 //wyjście odpowiadające wejściu zezwalającemu lewego czujnika
#define echoPinLeft 3 //wejście odpowiadające wyjściu lewego czujnika zwracającego pomiar

#define barier 30.0 //odległość wyrażona w centymetrach, w przypadku której robot ma zatrzymać się przed dalszą jazdą

#define degree90time 1200 //czas w mikrosekundach, przez jaki ma wykonywać się operacja skrętu, by robot skręcił o 90 stopni (na testowym podłożu)

#define motorSpeed 100 //częstotliwość decydująca o prędkości obrotowej silników

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

  pinMode(inputRight1, OUTPUT); //ustawienie pierwszego pinu prawego koła na wyjście
  pinMode(inputRight2, OUTPUT); //ustawienie drugiego pinu prawego koła na wyjście
  pinMode(enablePinRight, OUTPUT); //ustawienie PWM prawego koła na wyjście (zakres 0-225)

  pinMode(inputLeft1, OUTPUT); //ustawienie pierwszego pinu lewego koła na wyjście
  pinMode(inputLeft2, OUTPUT); //ustawienie drugiego pinu lewego koła na wyjście
  pinMode(enablePinLeft, OUTPUT); //ustawienie PWM lewego koła na wyjście (zakres 0-225)
```

```
  pinMode(trigPinFront, OUTPUT); // ustawienie pinów podłączonych do trig
czujnika jako wyjście
  pinMode(echoPinFront, INPUT); // ustawienie pinów podłączonych do echo czujnika
jako wejście

  pinMode(trigPinRight, OUTPUT);
  pinMode(echoPinRight, INPUT);

  pinMode(trigPinLeft, OUTPUT);
  pinMode(echoPinLeft, INPUT);

  analogWrite(enablePinRight, motorSpeed); //nadanie prędkości obrotowej silników
  analogWrite(enablePinLeft, motorSpeed);
}

void loop() {
  // algorytm omijania napotkanej przeszkody
  if(measureDistance('f') > barier){
    goForward();
    delay(100);
  }
  else{
    stop(1000);
    if(measureDistance('r') > barier){
      turnRight(degree90time);
    } else if(measureDistance('l') > barier) {
      turnLeft(degree90time);
    } else{
        goBack(1000);
        if(measureDistance('r') > barier){
          turnRight(degree90time);
        } else if(measureDistance('l') > barier) {
          turnLeft(degree90time);
        }
    }
  }
}

// pomiar odległości na wybranym czujniku
// parametr: direction - wybór kierunku pomiaru:
//f - front, przedmi czujnik
//r - right, prawy czujnik
//l - left, lewy czujnik
float measureDistance(char direction) {
  float czas, dystans;
```

```
  if(direction == 'f'){
    digitalWrite(trigPinFront, HIGH); // nadanie sygnału czujnika, rozpoczęcie
pomiaru odkległości
    delayMicroseconds(10); // odczekanie 10 mikrosekund
    digitalWrite(trigPinFront, LOW);

    czas = pulseIn(echoPinFront, HIGH); // odczyt czasu pomiaru
  }
  else if(direction == 'r'){
    digitalWrite(trigPinRight, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPinRight, LOW);

    czas = pulseIn(echoPinRight, HIGH);

  }
  else if(direction == 'l'){
    digitalWrite(trigPinLeft, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPinLeft, LOW);

    czas = pulseIn(echoPinLeft, HIGH);
  }

  dystans = czas / 58.00; // obliczenie odległości od przeszkody (w centymetrach)
  // testowe wypisanie odkległości
  Serial.print(direction);
  Serial.print(": ");
  Serial.println(dystans);

  return dystans;
}

// skręt w prawo
// parametr: time - czas skrętu
void turnRight(long time){
  rightRun(1);
  leftRun(0);
  delay(time);
}

// skręt w lewo
// parametr: time - czas skrętu
void turnLeft(long time){
  rightRun(0);
```

```
  leftRun(1);
  delay(time);
}

// jazda do przodu
// parametr: time - czas jazdy
void goForward(){
  rightRun(1);
  leftRun(1);
}

// jazda do tyłu
// parametr: time - czas jazdy
void goBack(long time){
  rightRun(-1);
  leftRun(-1);
  delay(time);
}

// zatrzymanie się robota
// parametr: time - czas zatrzymania
void stop(int time){
  rightRun(0);
  leftRun(0);
  delay(time);
}

// nadanie kierunku prawego koła
// parametr: i - kierunek jazdy:
// -1 - koło jedzie do tyłu
//  0 - koło się zatrzymuje
//  1 - koło jedzie do przodu
void rightRun(int i){
  if(i == -1) { //back
    digitalWrite(inputRight1, LOW);
    digitalWrite(inputRight2, HIGH);
  }
  else if(i == 0){ //stop
    digitalWrite(inputRight1, LOW);
    digitalWrite(inputRight2, LOW);
  }
  else if(i == 1){ //forward
    digitalWrite(inputRight1, HIGH);
    digitalWrite(inputRight2, LOW);
  }
```

```
}

// nadanie kierunku lewego koła
// parametr: i - kierunek jazdy:
// -1 - koło jedzie do tyłu
//  0 - koło się zatrzymuje
//  1 - koło jedzie do przodu
void leftRun(int i){
  if(i == -1) { //back
    digitalWrite(inputLeft1, LOW);
    digitalWrite(inputLeft2, HIGH);
  }
  else if(i == 0){ //stop
    digitalWrite(inputLeft1, LOW);
    digitalWrite(inputLeft2, LOW);
  }
  else if(i == 1){ //forward
    digitalWrite(inputLeft1, HIGH);
    digitalWrite(inputLeft2, LOW);
  }
}
```

8. **Wnioski**

Robot zgodnie z oczekiwaniami był w stanie omijać przeszkody. W idealnych lub zbliżonych do idealnych warunkach, robot omijał przeszkody i nie dochodziło do kolizji ze ścianami.

Słabą stroną rozwiązania było zasilanie – baterie wyładowywały się bardzo szybko, zmuszając do ich wymiany lub użycia dodatkowego źródła zasilania (np. powerbanka). Robot nie zawsze skręcał dokładnie o 90 stopni – przy innym typie podłoża niż testowe, zmieniało się tarcie, a przez to też stopień o jaki obróci się robot w założonym czasie. Problem ten prowadził do komplikacji – czujniki nie zawsze były w stanie wykryć przeszkodę, która nie była położona do nich prostopadle.