Budowa i oprogramowanie robota typu Line Follower z wykorzystaniem Arduino

Streszczenie

Celem projektu jest budowa i oprogramowanie autonomicznego robota, którego głównym zadaniem jest poruszanie się po wyznaczonej trasie. Robot jest w stanie omijać napotkane na torze przeszkody oraz może być zdalnie sterowany za pomocą smartfona. W projekcie wykorzystano mikrokontroler ATmega328, który został zaprogramowany przy pomocy platformy Arduino. Urządzenie wyposażone jest w odbiciowe czujniki podczerwieni i ultradźwiękowy czujnik odległości. W części teoretycznej pracy opisane zostały parametry oraz sposób działania komponentów robota. W części praktycznej przedstawiono proces budowy konstrukcji mechanicznej oraz opracowanie oprogramowania. Następnie omówione zostały wyniki przeprowadzonych testów, które potwierdziły prawidłowe działanie robota.

Słowa kluczowe: Arduino, robot autonomiczny, line follower

Building and programming of a Line Follower robot using Arduino

Summary

The project aims at building and developing software for an autonomous Line Follower robot. The device can avoid obstacles and be remote-controlled using a smartphone. An ATmega328 microcontroller was programmed using Arduino. The robot is equipped with infrared and ultrasound sensors. The theoretical framework describes and explains how particular electronic components work together. The practical research elaborates on the design of the robot and its software. The presented results of the tests prove that the device works correctly.

Keywords: Arduino, autonomous robot, line follower

SPIS TREŚCI

1. Wprowadzenie

1.1 Definicja i zastosowanie robota typu Line Follower

Line Follower jest mobilnym, autonomicznym robotem, który porusza się po wyznaczonej trasie. Jest to przeważnie czarna linia na białym podłożu. Robot autonomiczny jest w stanie samodzielnie podejmować decyzje na podstawie danych zarejestrowanych przez czujniki. Rola człowieka powinna ograniczać się jedynie do zaprogramowania i uruchomienia robota [1].

Specjalistyczne urządzenia o podobnej konstrukcji wykorzystywane są do transportu wyposażenia na zautomatyzowanych halach produkcyjnych. Mniej złożone projekty są popularne wśród osób rozpoczynających naukę robotyki. Wielu studentów uczestniczy w konkursach polegających na zbudowaniu i oprogramowaniu robota, który najszybciej pokona wyznaczoną trasę. Podczas niektórych konkurencji konkursowych sprawdzane jest również, czy robot jest w stanie sprawnie omijać napotkane na trasie przeszkody.

1.2 Historia platformy Arduino

Platforma Arduino była początkowo skierowana do studentów – miała ułatwić im naukę programowania i budowania prototypów. Nauczyciel elektroniki Massimo Banzi uważał ówczesne platformy do nauki elektroniki za zbyt skomplikowane i kosztowne. Banzi postanowił samodzielnie zaprojektować sprzęt, który mógłby zostać wykorzystany w celach edukacyjnych. Z upływem czasu Arduino zyskało również zainteresowanie osób niezwiązanych ze szkolnictwem. Na przestrzeni lat wzrosła liczba dostępnych rodzajów płytek bazowych. Pojawiły się bardziej rozbudowane rozwiązania, które mogły znaleźć zastosowanie w urządzeniach IoT i robotach.

Obecnie platforma Arduino korzysta z open source, dzięki czemu każdy użytkownik może dostosować wybraną płytkę do swoich potrzeb. Środowisko do tworzenia aplikacji Arduino Integrated Development Environment (IDE) jest

stosunkowo łatwe w obsłudze i bardzo elastyczne – mogą się nim swobodnie posługiwać zarówno nowicjusze, jak i zaawansowani użytkownicy [2].

1.3 Motywacja

Główną motywacją do zrealizowania projektu była chęć poszerzenia wiedzy w zakresie robotyki – dziedziny nauki łączącej w sobie elementy informatyki, elektroniki i automatyki. Budowa robota wymagała przygotowania konstrukcji mechanicznej, instalacji odpowiednio dobranych komponentów elektronicznych oraz napisania oprogramowania umożliwiającego sterowanie. Nabycie wyżej wymienionych umiejętności może okazać się przydatne podczas konstruowania bardziej złożonych urządzeń.

Wszystkie komponenty zostały dobrane tak, aby możliwe było skonstruowanie w pełni sprawnego robota z wykorzystaniem niewielkich nakładów finansowych. Każdy z wykorzystanych elementów jest tani i szeroko dostępny na polskim rynku, dzięki czemu łatwo można go zastąpić w przypadku uszkodzenia. Zbudowany w ramach projektu robot porusza się po czarnej linii, jest w stanie omijać napotkane na trasie przeszkody oraz może być zdalnie sterowany przy pomocy aplikacji mobilnej przeznaczonej na smartfony z systemem operacyjnym Android.

Praca dyplomowa składa się z sześciu rozdziałów. W rozdziale drugim scharakteryzowano wykorzystane komponenty oraz wyjaśniono sposób ich działania. Rozdział trzeci obejmuje opracowane założenia projektowe. W czwartym rozdziale przedstawiono kolejne etapy budowy konstrukcji mechanicznej oraz omówiono najważniejsze funkcje odpowiedzialne za sterowanie robotem. Wyniki weryfikacji pracy urządzenia zostały zawarte w rozdziale piątym. W ostatnim rozdziale podsumowano rezultaty pracy oraz przedstawiono przykładowe ulepszenia, które należałoby wprowadzić, aby zwiększyć wydajność robota.

2. Charakterystyka robota typu Line Follower wraz z jego elementami

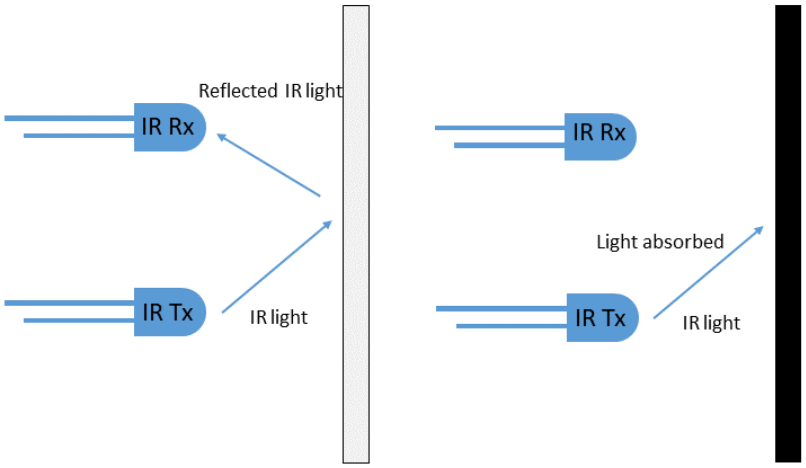
Robot typu Line Follower musi zostać wyposażony w czujniki, układ sterujący oraz układ wykonawczy. Dzięki czujnikom robot rejestruje informacje pochodzące z otoczenia. Zadaniem układu sterowania jest interpretowanie zarejestrowanych przez czujniki danych oraz wydanie odpowiednich poleceń do układu wykonawczego [3].

2.1 Czujniki

W projekcie wykorzystano dwa rodzaje czujników. Robot rejestruje obecność czarnej linii na białym podłożu dzięki trzem odbiciowym czujnikom podczerwieni zamontowanym na podwoziu. W celu wykrycia przeszkody na trasie urządzenie korzysta z ultradźwiękowego czujnika odległości.

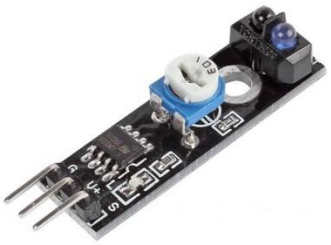
2.1.1 Odbiciowe czujniki podczerwieni

Dioda nadawcza emituje światło podczerwone, które zależnie od koloru napotkanej powierzchni jest odbijane (jasna powierzchnia) lub pochłaniane (ciemna powierzchnia). Fototranzystor rejestruje odbite od podłoża światło, co powoduje zmianę przepływu prądu między emiterem a kolektorem. Przepływ prądu przez obwód detektora jest zależny od sposobu, w jaki powierzchnia odbija światło [4]. Można zatem stwierdzić, że im jaśniejsza powierzchnia, tym większa wartość otrzymanego sygnału. Dane z odbiornika przesyłane są przez pin cyfrowy. W przypadku, gdy do czujnika będzie docierać zbyt mała ilość światła podczerwonego, stan na wyjściu ustawiony zostanie jako wysoki. Natomiast kiedy poziom wartości progowej potencjometru zostanie przekroczony, wtedy stan na wyjściu będzie niski.



Rysunek 2.1. Działanie odbiciowych czujników podczerwieni w zależności od koloru napotkanej powierzchni

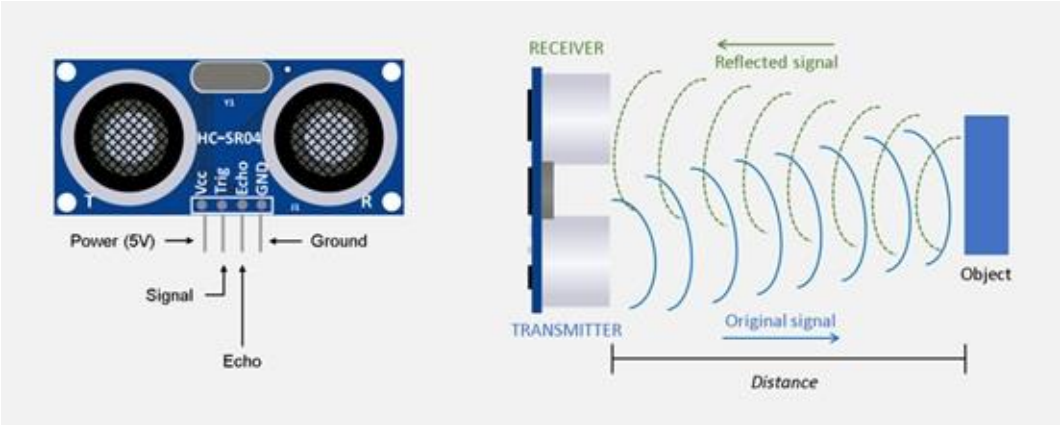
W projekcie wykorzystane zostały trzy moduły TCRT5000 z czujnikiem odbiciowym i komparatorem LM393. Zakres działania czujnika wynosi od 0 do 8 mm. Jego czułość może być regulowana przy pomocy wbudowanego potencjometru.



Rysunek 2.2. Moduł TCRT5000

2.1.2 Ultradźwiękowy czujnik odległości

Czujnik emituje fale dźwiękowe o wysokiej częstotliwości, a następne oczekuje, aż zostaną odbite od powierzchni przedmiotu znajdującego się w pobliżu. Odległość między czujnikiem a przeszkodą zostaje obliczona na podstawie zmierzonego czasu, który upłynął od momentu wyemitowania sygnału, aż do zarejestrowania jego odbicia.



Rysunek 2.3. Działanie ultradźwiękowego czujnika odległości

W projekcie wykorzystany został czujnik ultradźwiękowy HC-SR04. Posiada on cztery piny sygnałowe. Piny VCC i GND zasilają układ a TRIGGER wraz z ECHO odpowiadają za dokonywanie pomiarów. W celu rozpoczęcia pomiaru należy podać na wejście wyzwalające TRIGGER impuls napięciowy w stanie wysokim 5V przez minimum 10µs. Następnie na pin ECHO wystawiony zostaje sygnał, którego długość trwania zależna jest od odległości urządzenia od przeszkody [5].

Poniżej przedstawiono wzór matematyczny stosowany podczas obliczania odległości.

$$s = \frac{v * t}{2}$$

Gdzie:

s - odległość od przeszkody (cm)

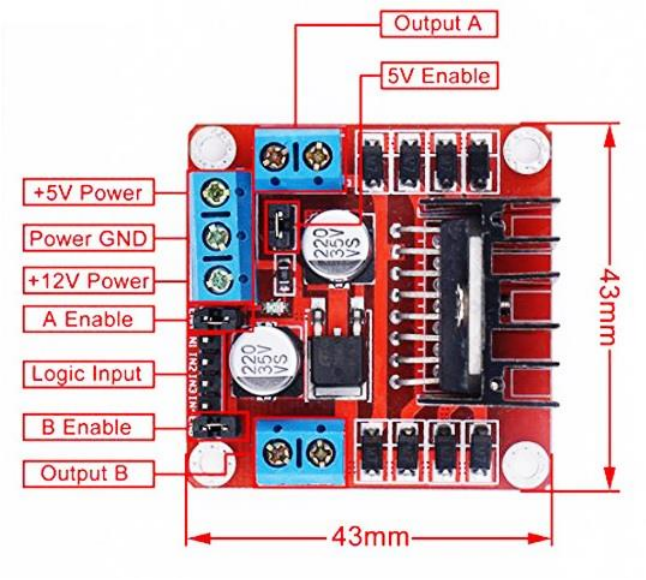
t - czas trwania sygnału ECHO (μs)

v - prędkość dźwięku w powietrzu ($\frac{\mathrm{m}}{\mathrm{s}}$)

Czas trwania sygnału ECHO musi zostać podzielony na pół, ponieważ w obliczeniach uwzględniona została długość czasu, jaka upłynęła od odbicia się sygnału do zarejestrowania jego odbicia.

2.2 Sterownik silników

Kierunek i liczba obrotów silników prądu stałego kontrolowane są przez dwukanałowy sterownik L298N. Sygnały sterujące są wyprowadzone na złącza szpilkowe typu goldpin, dzięki czemu możliwe jest łączenie modułu z płytką Arduino za pomocą przewodów. Wykorzystany moduł jest lekki i zajmuje niewielką powierzchnię – cechy te są istotne podczas budowania robota mobilnego.



Rysunek 2.4. Dwukanałowy sterownik silników L298N

Tabela 2.1. Wyprowadzenia modułu ze sterownikiem L298N [6]

Pin	Opis
+12V	Napięcie zasilania silników
+5V	Zasilanie części logicznej
GND	Masa układu
OUT1, OUT2	Wyjścia kanału silnika A
OUT3, OUT4	Wyjścia kanału silnika B
ENA	Sygnał PWM do sterownia prędkością obrotową silnika A
IN1, IN2	Sterowanie kierunkiem kanału A
ENB	Sygnał PWM do sterownia prędkością obrotową silnika B
IN3, IN4	Sterowanie kierunkiem kanału B

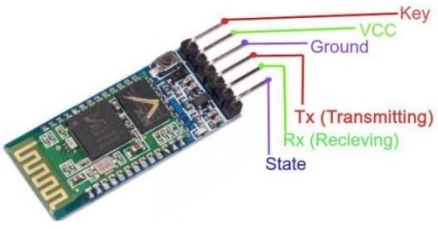
Tabela 2.2. Tabela prawdy sterownika L298N [6]

IN1/IN3	IN2/IN4	Opis
-	-	Silnik jest wyłączony
0	0	Silnik hamuje
0	1	Silnik jest włączony i obraca się w kierunku przeciwnym do ruchu wskazówek zegara
1	0	Silnik jest włączony i obraca się w kierunku zgodnym z ruchem wskazówek zegara
1	1	Silnik hamuje

2.3 Moduł Bluetooth

Moduł Bluetooth HC-05 jest powszechnie używany do komunikowania się z urządzeniami wykorzystującymi mikrokontroler Arduino. Umożliwia on połączenie urządzenia ze smartfonami z systemem Android. HC-05 komunikuje się poprzez komendy AT. Są to ciągi znaków ASCII, które są wysyłane do modułu poprzez interfejs UART. W tym celu korzysta się z dwóch linii: RxD (tor danych odbieranych) oraz TxD (tor danych nadawanych). UART jest układem scalonym, którego zadaniem jest asynchroniczne przekazywanie oraz odbieranie informacji za pośrednictwem portu szeregowego. W tego typu układzie wykorzystywane są dwa konwertery: równoległo - szeregowy oraz szeregowo - równoległy. Pierwszy z nich służy do konwersji danych pochodzących z komputera a drugi do konwersji danych przychodzących do komputera [7].

W projekcie wykorzystano technologię Bluetooth ze względu na jej niskie koszty, niski pobór energii oraz możliwość komunikacji ze smartfonami.



Rysunek 2.5. Moduł Bluetooth HC-05

2.4 Płytka uruchomieniowa

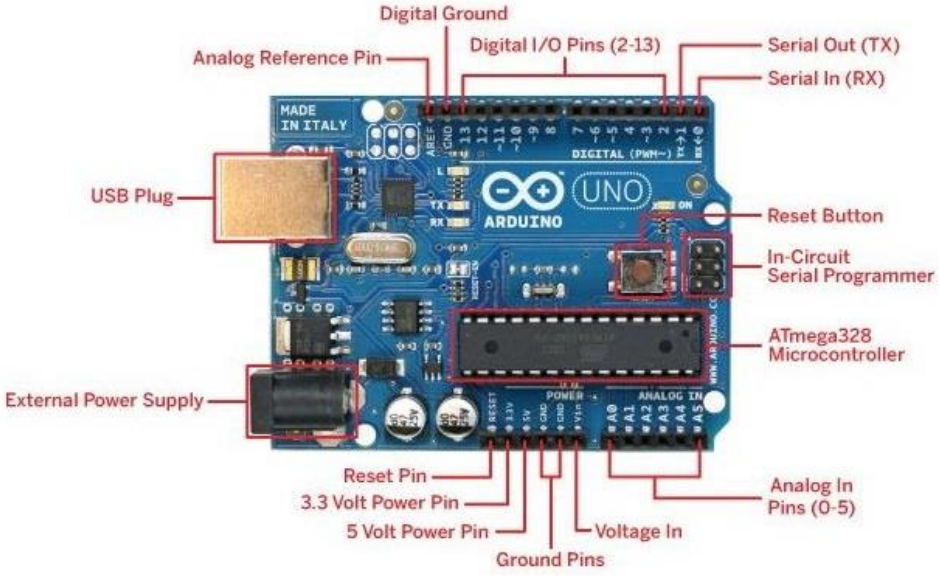
W projekcie wykorzystano Arduino Uno Rev3 – jedną z najpopularniejszych podstawowych płytek uruchomieniowych wyposażonych w mikrokontroler Atmel ATmega328. Płytka nie wymaga zewnętrznego programatora oraz umożliwia łatwą wymianę uszkodzonego kontrolera [8].

Specyfikacja dla mikrokontrolera ATmega328:

- Pamięć Flash: 32 kB
- Pamięć SRAM: 2 kB
- Pamięć EEPROM: 1 kB

Specyfikacja dla płytki:

- 14 cyfrowych linii I/O (6 może pełnić funkcję wyjścia PWM)
- 6 wejść analogowych
- Maksymalna częstotliwość zegara: 16MHz
- Zalecane napięcie zasilania 7-12 V



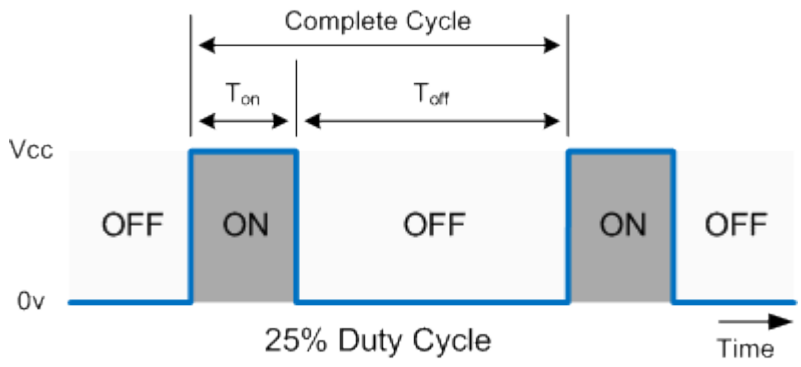
Rysunek 2.6. Płytka uruchomieniowa Arduino Uno Rev3

2.5 Układ wykonawczy

Układ wykonawczy stanowią cztery silniki prądu stałego oraz mikroserwo kontrolowane za pomocą modulacji PWM.

2.5.1 Sterowanie sygnałem PWM

Najpopularniejszą metodą sterowania silnikami prądu stałego jest PWM (*ang. Pulse-Width Modulation*). Polega ona na regulowaniu sygnału napięciowego poprzez zmianę szerokości impulsu o stałej amplitudzie. Prędkość obrotowa silnika zależna jest od tego, jak długo silnik pozostaje zasilany w danym przedziale czasu. Stosunek czasu zasilania silnika oraz czasu, w trakcie którego napięcie wynosi 0V nazywany jest cyklem pracy. Cykl pracy odnosi się do czasu, przez który urządzenie jest włączone. Jeśli wynosi on 75%, to impuls jest włączony przez 75% czasu, a wyłączony przez 25%. PWM jest cyfrowy, co oznacza, że posiada dwa stany: wyłączony i włączony (które odpowiadają 0 i 1 w zapisie binarnym) [9].



Rysunek 2.7. Schemat ilustrujący działanie PWM na przykładzie cyklu pracy wynoszącego 25%

2.5.2 Silniki

Koła robota wprawiane są w ruch przy użyciu czterech szczotkowych mikrosilników prądu stałego z przekładnią. Robot typu Line Follower powinien być w stanie szybko reagować na nagłe zmiany kształtu linii. Kluczowym czynnikiem decydującym o wyborze silnika jest liczba obrotów na minutę, jaką jest w stanie wykonać silnik. Zbudowany w ramach projektu robot nie jest przeznaczony do udziału w konkursie, dlatego też silniki zostały wybrane ze względu na korzystny stosunek mocy do ceny.

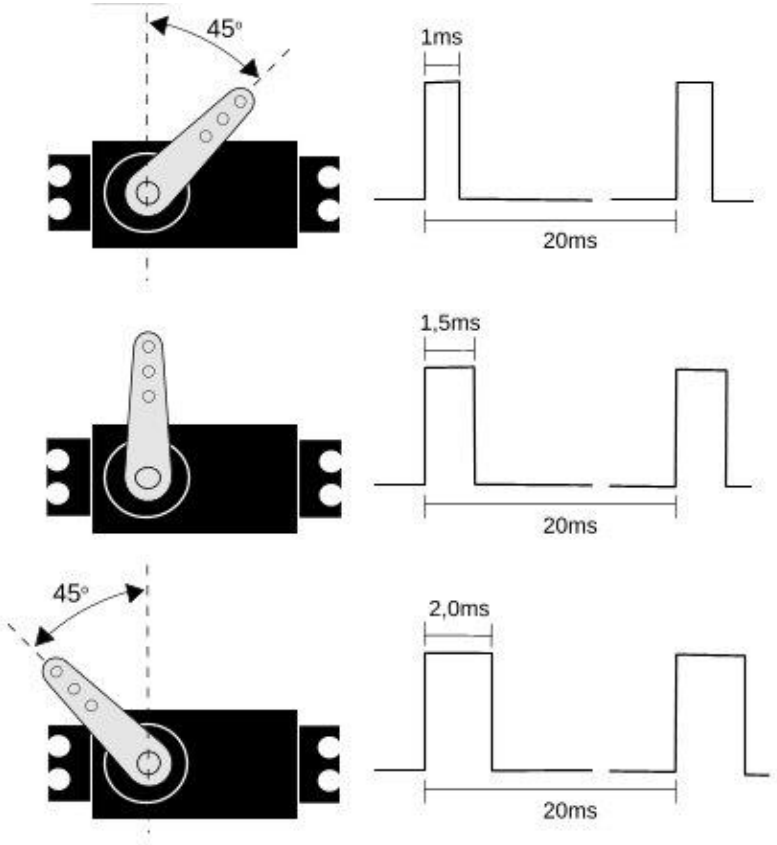
Specyfikacja silników:

- Znamionowe napięcie zasilania silników: 6 V
- Prędkość obrotowa osi wyjściowej motoreduktora: 100 RPM
- Pobór prądu przy zasilaniu napięciem 6 V: do 350 mA
- Siła ciągu na przekładni (6V): 5,5 KG/cm

Rysunek 2.8. Mikrosilnik prądu stałego

2.5.3 Serwomechanizm

Serwomechanizm składa się z silnika prądu stałego, przekładni, potencjometru i sterownika. Serwo jest w stanie wykonać obrót o zadany kąt w zakresie od 0 do 180°. Sterownik ustawia serwo w odpowiedniej pozycji na podstawie otrzymanego z kontrolera sygnału PWM [10].



Rysunek 2.9. Schemat ilustrujący sterowanie serwomechanizmem

W projekcie wykorzystane zostało mikroserwo TowerPro SG-90. Szerokość impulsu dla takiego modelu powinna wynosić między 1ms a 2s a czas trwania sygnału sterującego 20ms.

3. Założenia projektowe

Założenia projektowe zostały podzielone ze względu na działanie poszczególnych części projektu.

Założenia związane z działaniem robota:

- Robot sprawnie porusza się po czarnej linii o szerokości 2cm znajdującej się na jasnej powierzchni oraz pokonuje zakręty o zróżnicowanym promieniu krzywizny.
- Robot samodzielnie rozpoczyna podążanie za linią oraz zatrzymuje się w miejscu oznaczonym, jako koniec trasy (czarna poprzeczna linia).
- Podczas przejazdu robot nie powinien wypadać z zakrętów ani zbaczać z trasy.
- Robot jest w stanie omijać napotkane na trasie przeszkody.
- Robot może być zdalnie sterowany przez użytkownika przy pomocy aplikacji mobilnej przeznaczonej na smartfony z systemem Android.
- Robot pozytywnie przechodzi testy polegające na sprawdzeniu szybkości i precyzji, z jaką pokonuje wyznaczoną trasę.

Założenia związane z działaniem aplikacji mobilnej:

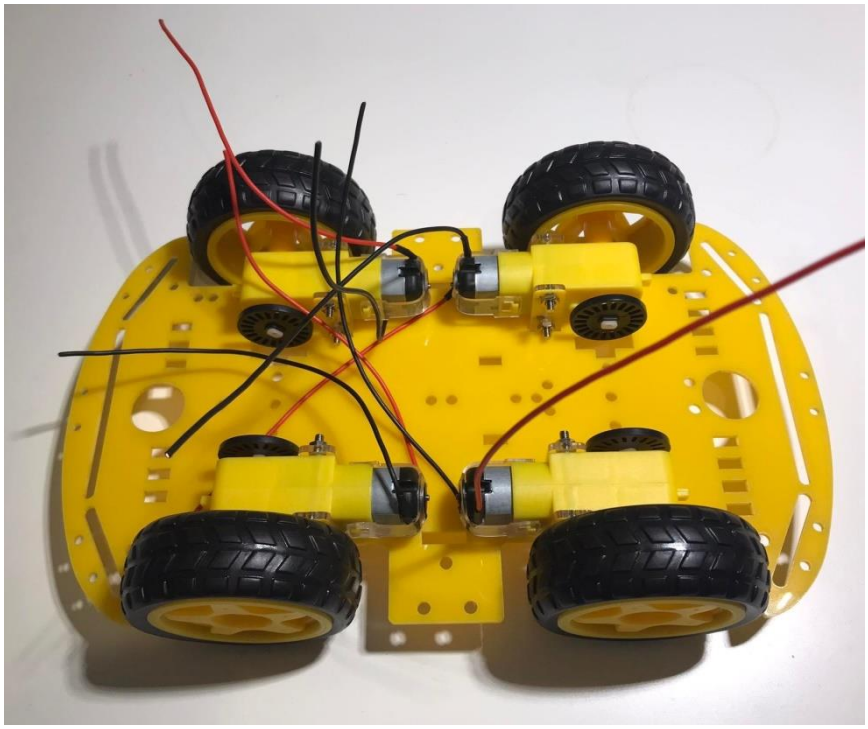
- Aplikacja przeznaczona jest na smartfony z systemem Android.
- Użytkownik może łączyć i rozłączać swojego smartfona z wybranym z listy urządzeniem posiadającym moduł Bluetooth.
- Użytkownik może wybrać tryb sterowania robotem, gdzie:
  - Tryb automatyczny oznacza, że robot porusza się samodzielnie po wyznaczonej trasie.
  - Tryb manualny oznacza, że użytkownik steruje robotem za pomocą aplikacji mobilnej.
- Użytkownik może wybierać kierunek, w jakim ma poruszać się robot klikając na jedną z czterech strzałek, gdzie:
  - ↑ - oznacza, że robot będzie poruszać się do przodu,
  - ↓ - oznacza, że robot będzie poruszać się do tyłu,
  - → - oznacza, że robot będzie poruszać się w prawo,
  - ← - oznacza, że robot będzie poruszać się w lewo.

- Użytkownik może wybierać kierunek, w jakim ma poruszać się robot wydając polecenia głosowe w języku angielskim, gdzie:
  - *Go forward* – oznacza, że robot będzie poruszać się do przodu,
  - *Go back* – oznacza, że robot będzie poruszać się do tyłu,
  - *Turn right* – oznacza, że robot będzie poruszać się w prawo,
  - *Turn left* – oznacza, że robot będzie poruszać się w lewo.

4. Praktyczna realizacja projektu: konfiguracja mikrokontrolera, instalacja oprogramowania, czujników oraz innych elementów robota

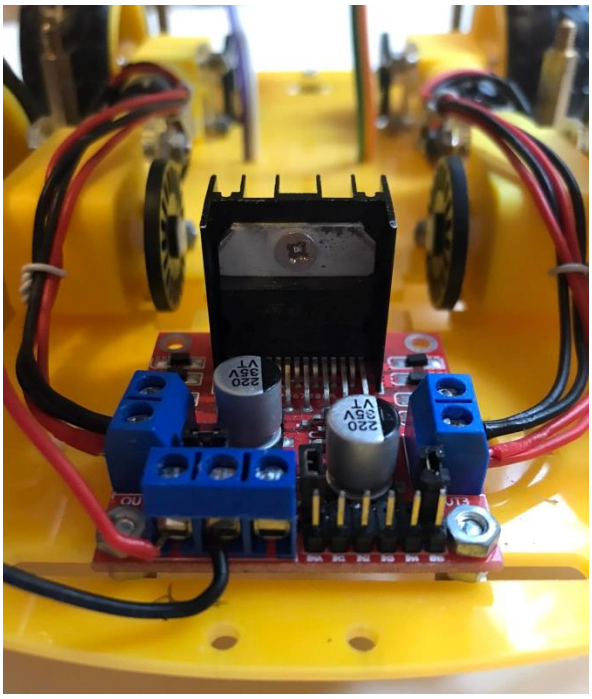
4.1 Konstrukcja mechaniczna

Podstawę konstrukcji mechanicznej robota stanowią dwie płyty pleksi połączone ze sobą za pomocą dystansów mocujących. Do silników przylutowane zostały przewody, które zabezpieczono z wykorzystaniem koszulek termokurczliwych. Uchwyt każdego z silników został przymocowany za pomocą dwóch śrub imbusowych. Koła zostały przytwierdzone do osi silników, a następnie całość zamocowano do dolnej podstawy robota.



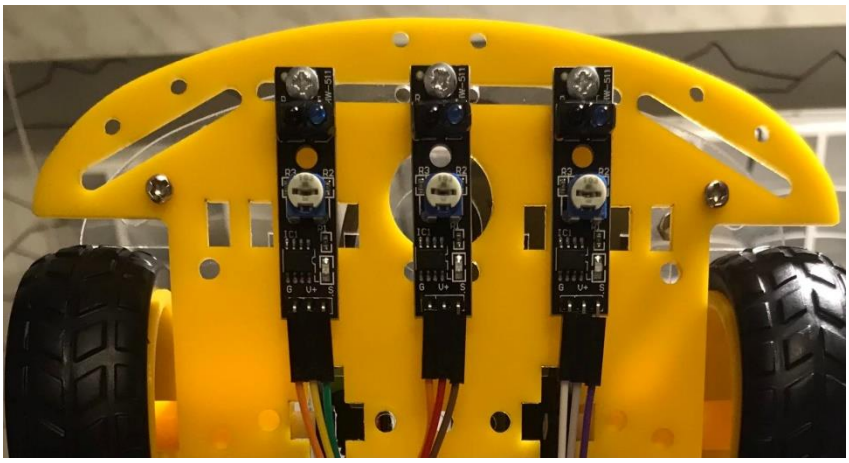
Rysunek 4.1. Podwozie robota wraz z zamontowanymi silnikami i kołami

Do tylnej części podwozia przykręcona została płytka sterująca silnikami L298N. Końce przewodów silników przylutowano do odpowiednich wtyczek znajdujących się w module sterującym, gdzie *OUTPUT1* dotyczy silników po lewej stronie a *OUTPUT2* silników po prawej stronie. Koła nie są skrętne, dlatego pojazd będzie pokonywać zakręty poprzez zmianę liczby obrotów silników po wybranej stronie robota.



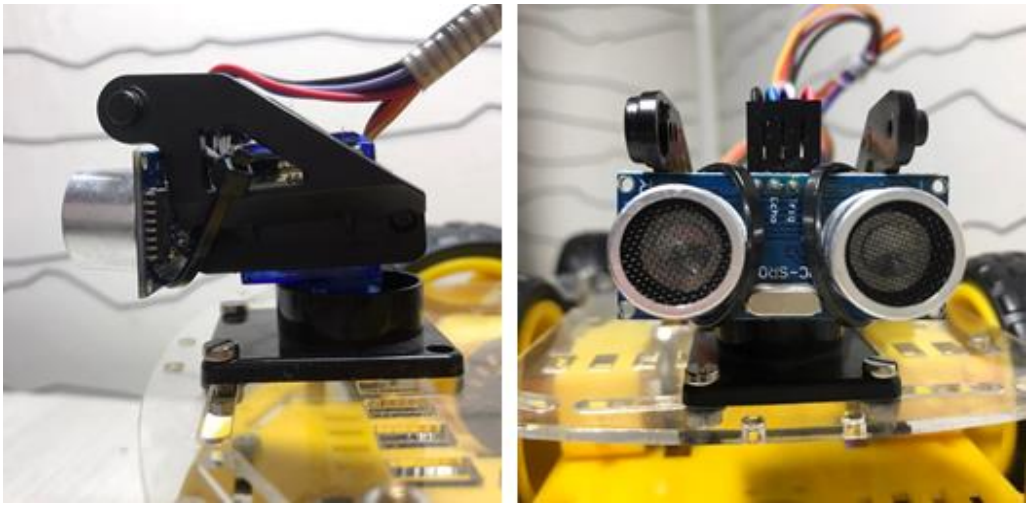
Rysunek 4.2. Moduł sterowania wraz z przyłączonymi silnikami

Z przodu podstawy robota przykręcone zostały trzy czujniki odbiciowe. Odległość między każdym czujnikiem musi być zbliżona do szerokości czarnej linii. W projekcie wartość ta wynosi 2cm. Dzięki takiemu ustawieniu sensory będą w stanie trafnie odczytywać informacje o zmieniającej się krzywiźnie trasy.



Rysunek 4.3. Czujniki odbiciowe zamontowane na podwoziu

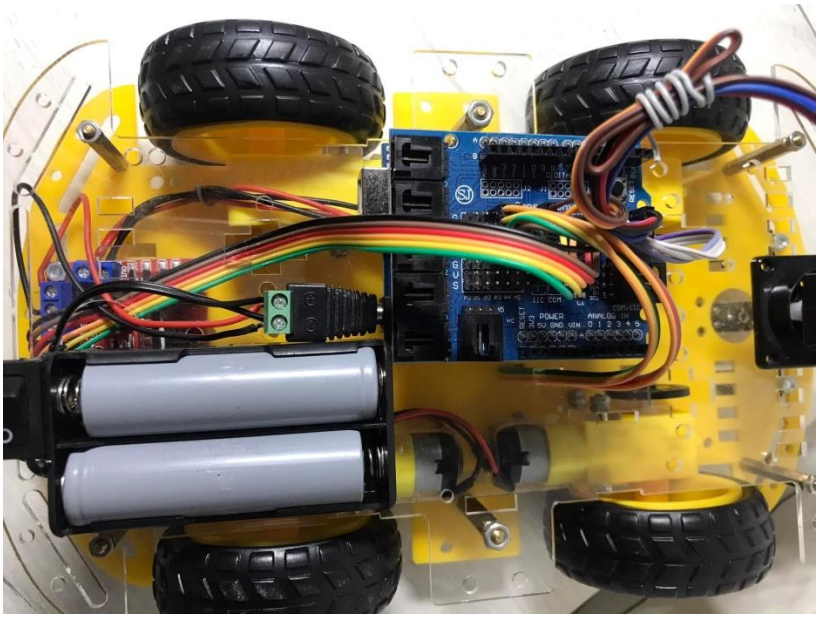
Z przodu górnej podstawy robota znajduje się ultradźwiękowy czujnik odległości. Jest on przymocowany do serwomechanizmu przy pomocy plastikowego uchwytu i opaski zaciskowej. Ramię mikroserwo jest przytwierdzone do podstawki uchwytu. Czujnik odległości zmienia swoją pozycję wraz z każdym wychyleniem serwomechanizmu. Dzięki takiemu połączeniu cała konstrukcja zyskuje możliwość obrotu, co zwiększa skuteczność wykrywania przeszkody przez robota.



Rysunek 4.4. Ultradźwiękowy czujnik odległości przytwierdzony do serwomechanizmu przy pomocy uchwytu

Na środku górnej podstawy robota zamontowano płytkę uruchomieniową Arduino Uno Rev3 wraz z kompatybilnym do niej modułem Sensor Shield V5.0. Ze względu na obecność pinów cyfrowych i analogowych wyprowadzonych bezpośrednio z płytki Arduino nakładka jest szczególnie pomocna w projektach, które wymagają użycia dużej liczby przewodów. Dzięki niej podłączanie komponentów do Arduino jest znacznie ułatwione i zajmuje mniej czasu. Obok płytki uruchomieniowej znajduje się przymocowany za pomocą opaski zaciskowej moduł Bluetooth.

Z tyłu podstawy za pomocą dwustronnej taśmy montażowej przytwierdzony został koszyk na akumulatory. Robot jest zasilany dwoma szeregowo połączonymi ogniwami litowo - jonowymi o pojemności 2Ah.



Rysunek 4.5. Górna podstawa konstrukcji mechanicznej robota

4.2 Oprogramowanie

Robot został w całości zaprogramowany z wykorzystaniem środowiska Arduino IDE, którego język jest zbliżony do C/C++. Pracę nad oprogramowaniem można podzielić na kilka podstawowych etapów:

- Zaprogramowanie silników tak, aby robot mógł poruszać się z daną prędkością w wybranym kierunku.
- Dodanie instrukcji warunkowych, dzięki którym robot będzie wykrywać i odpowiednio reagować na zmianę krzywizny toru.
- Dodanie instrukcji warunkowych, dzięki którym robot będzie wykrywać i omijać napotkaną na drodze przeszkodę.
- Wprowadzenie możliwości odbierania i interpretacji poleceń wydawanych przez użytkownika korzystającego z aplikacji mobilnej.

4.2.1 Sterowanie silnikami

Wykorzystany w projekcie moduł sterujący jest dwukanałowy, dlatego też kierunek i liczba obrotów pary silników zamontowanych po tej samej stronie podwozia są kontrolowane w tym samym momencie.

Prędkość silników zależna jest od dobranej wartości współczynnika wypełnienia impulsu. Parametr ten można ustawić w środowisku Arduino IDE przy użyciu wbudowanej funkcji *analogWrite()* odpowiadającej za generowanie sygnałów PWM [11]. Funkcja ta, jako pierwszy argument przyjmuje wartość cyklu pracy wyrażonego w postaci liczby z przedziału od 0 do 255. Wartość 0 oznacza, że silnik jest wyłączony a wartość 255, że silnik pracuje przez cały czas.

Kierunek obrotu silników kontrolowany jest dzięki funkcji *digitalWrite()*. Robot zacznie poruszać się w wybranym kierunku, kiedy na pinie przypisanemu do odpowiedniego silnika ustawiony zostanie stan wysoki. W tym celu utworzono cztery funkcje – każda z nich odpowiada za poruszanie się robota w innym kierunku. Wartości stanu dla poszczególnych pinów cyfrowych dobrano w oparciu o tabelę prawdy płytki sterującej L298N, która została zamieszczona w podrozdziale 2.2.1.

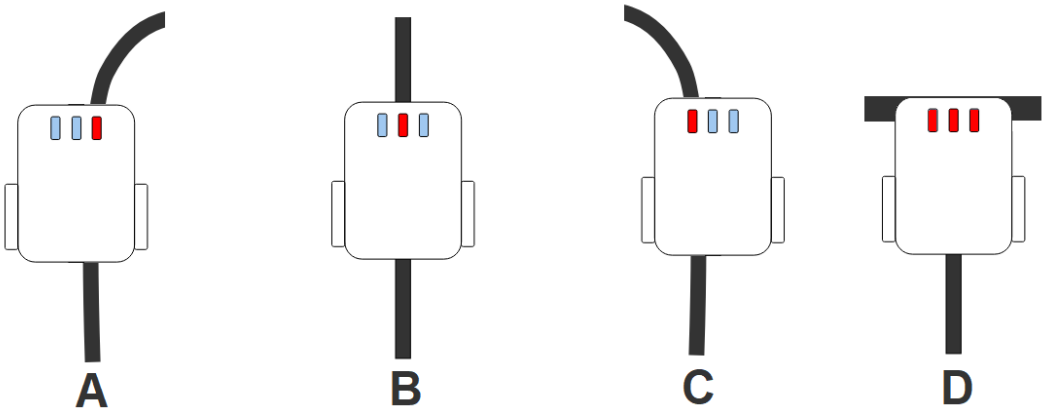
```
#define ENA 3 //Włączenie prawych silników
#define ENB 11 //Włączenie lewych silników
#define IN1 6 //Sterowanie kierunkiem obrotu prawych silników
#define IN2 7 //Sterowanie kierunkiem obrotu prawych silników
#define IN3 8 //Sterowanie kierunkiem obrotu lewych silników
#define IN4 9 //Sterowanie kierunkiem obrotu lewych silników


//Jazda na wprost
void go_forward(int value){
  analogWrite(ENA, value);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  analogWrite(ENB, value);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}
```

```
18 //Jazda do tyłu
19 void go_back(int value){
20   analogWrite(ENA, value);
21   digitalWrite(IN1, HIGH);
22   digitalWrite(IN2, LOW);
23   analogWrite(ENB, value);
24   digitalWrite(IN3, HIGH);
25   digitalWrite(IN4, LOW);
26 }
27 //Skręcanie w lewo
28 void turn_left(int value){
29   analogWrite(ENA, value);
30   digitalWrite(IN1, HIGH);
31   digitalWrite(IN2, LOW);
32   analogWrite(ENB, value);
33   digitalWrite(IN3, LOW);
34   digitalWrite(IN4, HIGH);
35 }
36 //Skręcanie w prawo
37 void turn_right(int value{
38   analogWrite(ENA, value);
39   digitalWrite(IN1, LOW);
40   digitalWrite(IN2, HIGH);
41   analogWrite(ENB, value);
42   digitalWrite(IN3, HIGH);
43   digitalWrite(IN4, LOW);
44 }
45 //Zatrzymanie się
46 void Stop(){
47   digitalWrite(IN1, LOW);
48   digitalWrite(IN2, LOW);
49   digitalWrite(IN3, LOW);
50   digitalWrite(IN4, LOW);
51 }
```

Rysunek 4.6. Fragment kodu odpowiedzialny za sterowanie silnikami

4.2.1 Wykrywanie linii i podejmowanie decyzji o kierunku jazdy

Robot wykrywa czarną linię dzięki obecności trzech czujników odbiciowych zamontowanych na podwoziu. Czujniki są oddalone od siebie o około 2cm. Dzięki takiemu ustawieniu możliwe jest prawidłowe rejestrowanie przez robota zmiany kształtu toru.



Rysunek 4.7. Zachowanie czujników odbiciowych w zależności od promienia krzywizny trasy

Na schematycznym, uproszczonym rysunku przedstawiony został sposób, w jaki czujniki odbiciowe reagują na zmianę kształtu czarnej linii. W sytuacji A linia znajduje się jedynie w zasięgu prawego czujnika. Robot powinien podjąć decyzję o wykonaniu skrętu w prawo. Sytuacja C jest analogiczna do sytuacji A – wyłącznie lewy czujnik jest w stanie zarejestrować obecność czarnej linii. W takim przypadku robot powinien rozpocząć skręcanie w lewo. W przykładzie B linia rejestrowana jest przez środkowy czujnik. Robot powinien w takiej sytuacji kontynuować jazdę na wprost. Zakończenie trasy zostało przedstawione w punkcie D. Metę stanowi poprzeczna, czarna linia, której obecność zostaje odnotowana przez wszystkie trzy czujniki odbiciowe. Kiedy robot znajdzie się w takim miejscu, powinien natychmiast się zatrzymać. Aby robot mógł poprawnie rozpoznawać każdą z wymienionych sytuacji, oprogramowanie musi zawierać odpowiednie instrukcje warunkowe.

```
1 void loop(){
2 //Środkowy czujnik rejestruje czarną linię;
3 //robot jedzie na wprost
4  if ((digitalRead(LEFT_IR_SENSOR) == 0)&&
5      (digitalRead(MIDDLE_IR_SENSOR) == 1)&&
6      (digitalRead(RIGHT_IR_SENSOR) == 0)){
7      go_forward(150);
8 }
```

```
 9 //Lewy i środkowy czujnik rejestruje czarną linię;
10 //na trasie pojawia się łagodny zakręt w lewo
11  if ((digitalRead(LEFT_IR_SENSOR) == 1)&&
12      (digitalRead(MIDDLE_IR_SENSOR) == 1)&&
13      (digitalRead(RIGHT_IR_SENSOR) == 0)){
14      turn_left(150);
15 }
16 //Lewy czujnik rejestruje czarną linię;
17 //na trasie pojawia się ostry zakręt w lewo
18  else if ((digitalRead(LEFT_IR_SENSOR) == 1)&&
19      (digitalRead(MIDDLE_IR_SENSOR) == 0)&&
20      (digitalRead(RIGHT_IR_SENSOR) == 0)){
21      turn_left(150);
22 }
23 //Prawy i środkowy czujnik rejestruje czarną linię;
24 //na trasie pojawia się łagodny zakręt w prawo
25  else if ((digitalRead(LEFT_IR_SENSOR) == 0)&&
26      (digitalRead(MIDDLE_IR_SENSOR) == 1)&&
27      (digitalRead(RIGHT_IR_SENSOR) == 1)){
28      turn_right(150);
29 }
30 //Prawy czujnik rejestruje czarną linię;
31 //na trasie pojawia się ostry zakręt w prawo
32  else if ((digitalRead(LEFT_IR_SENSOR) == 0)&&
33      (digitalRead(MIDDLE_IR_SENSOR) == 0)&&
34      (digitalRead(RIGHT_IR_SENSOR) == 1)){
35      turn_right(150);
36 }
37 //Wszystkie czujniki rejestrują czarną linię;
38 //robot znalazł się na końcu trasy
39  else if ((digitalRead(LEFT_IR_SENSOR) == 1)&&
40      (digitalRead(MIDDLE_IR_SENSOR) == 1)&&
41      (digitalRead(RIGHT_IR_SENSOR) == 1)){
42      Stop();
43 }
44 }
```

Rysunek 4.8. Fragment kodu odpowiedzialny za wykrywanie czarnej linii

4.2.2 Wykrywanie i omijanie napotkanych na trasie przeszkód

Robot wykrywa obecność przeszkody dzięki ultradźwiękowemu czujnikowi odległości. W celu umożliwienia robotowi rejestrowania napotkanych na trasie obiektów utworzono funkcję *get_distance()*, która zwraca odległość od przeszkody wyrażoną w centymetrach.

```
int get_distance() {
  digitalWrite(TRIGGER, LOW);
  delayMicroseconds(2);
  //Aby rozpocząć pomiar należy podać na wejście TRIGGER
  //impuls napięciowy w stanie wysokim przez min. 10 mikrosekund
  digitalWrite(TRIGGER, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGGER, LOW);
  //Mierzenie czasu trwania impulsu ECHO
  duration = pulseIn(ECHO, HIGH);
  //Obliczanie odległości korzystając ze wzoru
  // przedstawionego w podrozdziale 2.1.2
  distance = duration/58;
  return distance;
}
```

Rysunek 4.9. Fragment kodu odpowiedzialny za odczytywanie odległości od napotkanej przeszkody

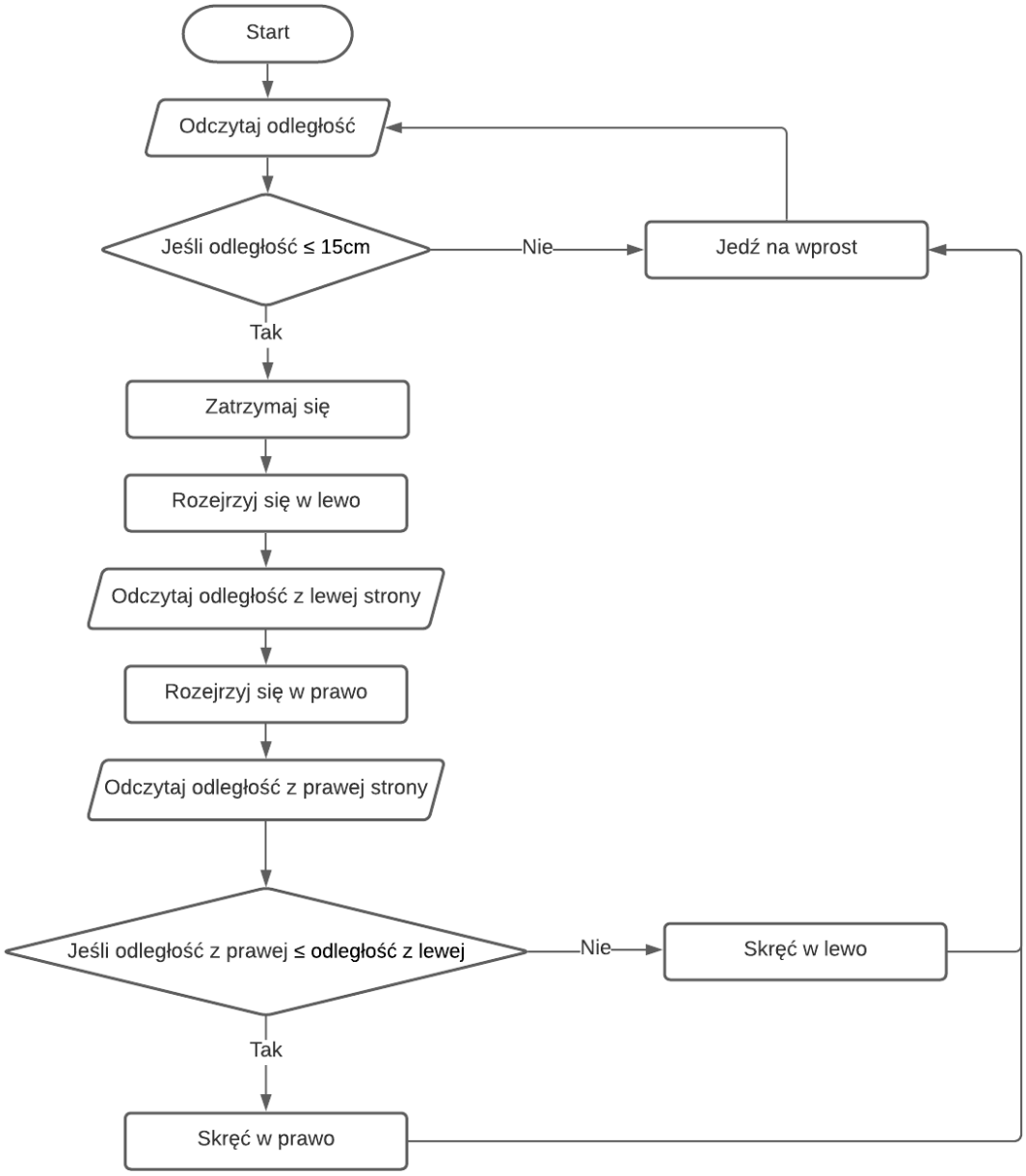
Kiedy robot zarejestruje, że w pobliżu znajduje się przeszkoda, powinien zatrzymać się i podjąć decyzję, w jaki sposób należy ją ominąć. Czujnik odległości został przymocowany do serwomechanizmu, dzięki czemu cała konstrukcja zyskuje możliwość obrotu. Kontrolując kąt wychylenia serwomechanizmu możliwe jest sprawdzenie, czy wokół napotkanej przeszkody nie ma innych obiektów, które utrudniłyby jej ominięcie. W tym celu utworzono dwie funkcje zwracające odległość po wybranej stronie przeszkody: *look_left()* oraz *look_right()*. Pierwsza z nich ustawia serwo pod kątem 30° a druga pod kątem 150°. Obie z nich działają analogicznie do omówionej wcześniej funkcji *get_distance()*.

Po sprawdzeniu, czy ominięcie przeszkody po wybranej stronie jest możliwe, robot powinien przejść do wykonywania manewru. Podczas omijania, prędkość silników musi zostać zredukowana. Zbyt wysoka liczba obrotów sprawi, że wykonywany manewr się nie powiedzie lub zostanie wykonany mało precyzyjnie. Funkcja *turn()* zawiera instrukcje warunkowe do sterowania silnikami, dzięki którym robot poprawnie ominie napotkany na trasie obiekt oraz wróci na tor jazdy.

Wszystkie wymienione powyżej funkcje zostały wykorzystane w głównej funkcji służącej do omijania przeszkód, *avoid obstacle()*.

```
1 void avoid_obstacle(){
2   /* Mierzenie odległości od znajdującej się
3    na tarsie przeszkody */
4   distance = get_distance();
5
6   /* Jeżeli przeszkoda znajduje się w zasięgu 15cm
7    robot zatrzymuje się */
8   if(distance <= 15){
9     Stop();
10
11  /* Sprawdzenie, czy po lewej stronie przeszkody
12   znajdują się inne obiekty */
13    look_left();
14
15  /* Sprawdzenie, czy po prawej stronie przeszkody
16   znajdują się inne obiekty */
17    look_right();
18
19    delay(100);
20
21  /* Jeżeli po prawej stronie przeszkody
22     zostaną wykyte inne obiekty
23     wartość zmiennej bool obstacle
24     zostanie ustawiona jako true;
25     funkcja turn() wyda polecenie
26     wykonania manewru omijania przeszkody
27     z lewej strony */
28    if(right_distance <= left_distance){
29      obstacle = true;
30      turn();
31    }
32    else{
33      obstacle = false;
34      turn();
35    }
36    delay(100);
37  }
38 }
```

Rysunek 4.10. Fragment kodu odpowiedzialny za podejmowanie decyzji w przypadku wykrycia przeszkody



Rysunek 4.11. Schemat algorytmu omijania napotkanych na trasie przeszkód

4.2.3 Zdalne sterowanie robotem

Dzięki bezprzewodowej komunikacji Bluetooth, użytkownik ma możliwość zdalnego sterowania silnikami, które zostały wcześniej podpięte do płytki Arduino. Moduł Blutooth komunikuje się za pomocą magistrali UART, która wykorzystuje dwie linie transmisji - Rx dla odbieranych danych oraz Tx dla transmitowanych danych. Aby rozpocząć transmisję szeregową należy użyć wbudowanej funkcji *Serial.begin()*, która przyjmuje jako parametr prędkość wyrażoną w bitach na sekundę [12]. Najczęściej przyjmowaną wartością w projektach jest 9600 bitów na sekundę. Funkcja *Serial.available()* zwraca liczbę odebranych bajtów. W sytuacji, kiedy dane będą dostępne, program powinien zapisać je do określonej zmiennej.

```
1 int bluetooth_data; /* Zmienna, do której zapisywana jest
2                        dana odebrana z modułu Bluetooth */
3
4 void setup(){
5   Serial.begin(9600); /* Uruchomienie szeregowej
6                          transmisji danych */
7 }
8
9 void loop(){
10   if(Serial.available() > 0){ //Sprawdź, czy odebrano dane
11     bluetooth_data = Serial.read(); /* Jeśli tak, zapisz je
12                                        do zmiennej */
13     Serial.println(bluetooth_data);
14   }
15 }
```

Rysunek 4.12. Fragment kodu odpowiedzialny za obsługę połączenia Bluetooth

Aby program był w stanie poprawnie interpretować otrzymane dane, należy napisać odpowiednie instrukcje warunkowe. Zgodnie z założeniami projektowymi, użytkownik może wybrać jeden z dwóch trybów pracy robota – automatyczny oraz manualny. W trybie manualnym, po wciśnięciu wybranego przycisku, do płytki przesyłany jest określony znak, który następnie zostaje interpretowany przez oprogramowanie.

Tabela 4.12. Instrukcje, które wykonuje robot po odebraniu przez port szeregowy określonego znaku

Znak	Instrukcja
1	Jedź na wprost
2	Jedź do tyłu
3	Skręć w lewo
4	Skręć w prawo
5	Zatrzymaj się
6	Włącz tryb automatyczny
7	Włącz tryb manualny

4.3 Aplikacja mobilna

Aplikacja mobilna została opracowana z wykorzystaniem platformy MIT App Inventor, której język jest zbliżony do języka Scratch. Środowisko to umożliwia szybkie tworzenie aplikacji obsługujących urządzenia IoT w przeglądarce internetowej.

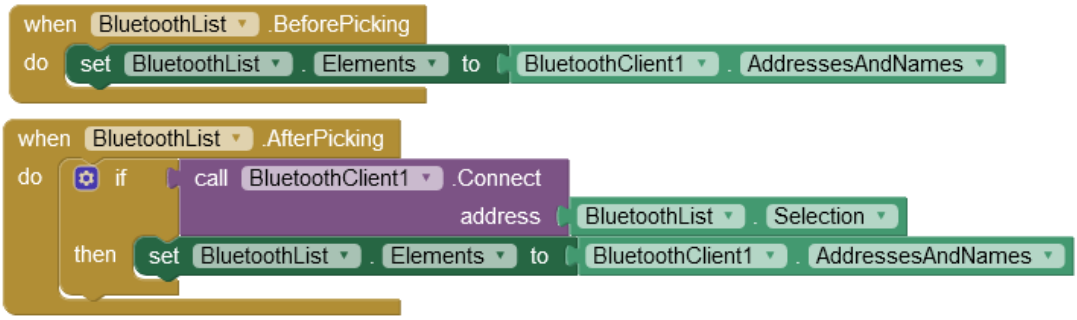
Panel platformy posiada dwa tryby pracy:

- Designer – odpowiada za projektowanie interfejsu graficznego aplikacji, które będą widoczne na ekranie smartfona (przyciski, suwaki, menu itp.).
- Blocks – odpowiada za oprogramowanie elementów utworzonych w trybie Designer.

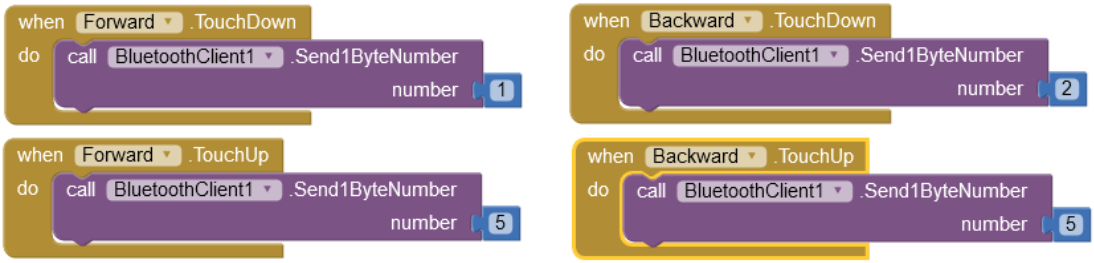
Opracowana na potrzeby projektu aplikacja wykorzystuje następujące elementy składowe:

- ListPicker – odpowiada za wybór urządzenia Bluetooth, z którym ma zostać sparowany smartfon.
- Button – przyciski, po których wciśnięciu do robota przesyłane zostają odpowiednie znaki, które następnie są interpretowane przez oprogramowanie.
- BluetoothClient – zapewnia łączność Bluetooth.
- SpeechRecognizer – umożliwia wydawanie poleceń głosowych.

Poniżej przedstawiono kilka ważniejszych, przykładowych bloków, które zostały utworzone w MIT App Inventor w ramach aplikacji:



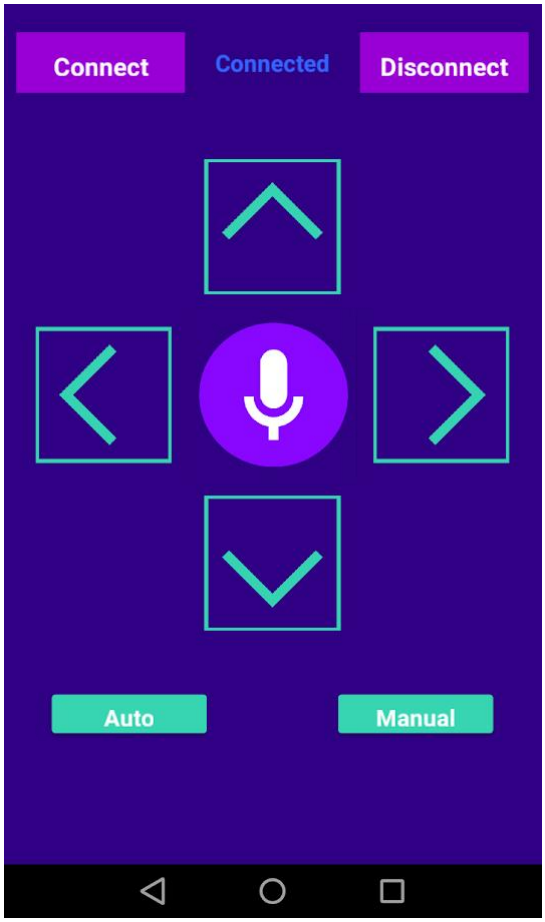
Rysunek 4.13. Fragment oprogramowania odpowiedzialny za wybór urządzenia, z którym ma zostać sparowany smartfon



Rysunek 4.14. Fragment oprogramowania odpowiedzialny za sterowanie robotem

Komponent *TouchDown* odnosi się do czasu, kiedy przycisk dopiero co został wciśnięty. Jeżeli użytkownik wybierze przycisk oznaczający jazdę na wprost, wtedy do urządzenia zostanie wysłany znak o wartości 1. Po tym zdarzeniu, aktywowany zostaje komponent *TouchUp*, który uruchamiany jest w później fazie wciskania przycisku. W tym momencie do urządzenia wysłany zostaje znak o wartości 5. Otrzymane w ten sposób znaki są później interpretowane przez płytkę Arduino i na ich podstawie wydawane są odpowiednie polecenia związane z pracą silników. Dzięki zastosowaniu powyższych komponentów, robot przez kilka sekund jedzie w wybranym przez użytkownika kierunku a następnie zatrzymuje się, czekając na podjęcie kolejnej decyzji. Analogicznie zbudowane bloki kodu zostały opracowane dla pozostałych przycisków oraz komend głosowych.

Rysunek 4.15. Część aplikacji odpowiedzialna za parowanie smartfona z wybranym urządzeniem posiadającym moduł Bluetooth



Rysunek 4.16. GUI aplikacji mobilnej

5. Weryfikacja pracy robota

5.1 Testowanie komponentów

Przed rozpoczęciem pracy nad przygotowaniem konstrukcji mechanicznej sprawdzono, czy poszczególne komponenty funkcjonują prawidłowo. Zbadano:

- pracę silników w zależności od dobranej wartości współczynnika wypełnienia PWM,
- pracę ultradźwiękowego czujnika odległości poprzez zbliżanie do czujnika obiektów i odczytywanie wykonanych przez niego pomiarów,
- pracę czujników odbiciowych w zależności od rodzaju powierzchni,
- kąt wychylenia serwomechanizmu w zależności od dobranej wartości współczynnika wypełnienia PWM,
- komunikację między modułem Bluetooth a płytką Arduino.

Wszystkie wymienione wyżej komponenty funkcjonowały prawidłowo, co umożliwiło rozpoczęcie dalszych prac nad projektem.

5.2 Testowanie komponentów

Tor, na którym mają zostać przeprowadzane jazdy testowe powinien umożliwiać sprawdzenie pracy robota na zróżnicowanej trasie. Utworzony na potrzeby projektu tor został wyklejony na białej podłodze czarną taśmą izolacyjną o szerokości 2cm. Tor składa się zarówno z łagodnych łuków jak i bardzo stromych zakrętów, które muszą zostać pokonane w krótkim czasie. Dzięki tak rozbudowanemu torowi jazdy, możliwe jest sprawdzenie, z jaką precyzją robot jest w stanie pokonywać trasę.

Rysunek 5.1. Tor testowy

Aby sprawdzić, jak dobrze robot radzi sobie z omijaniem przeszkód, ustawiono na mniej złożonych fragmentach toru obiekty o wymiarach około 20cm x 20cm. Robot powinien podjąć decyzję, z której strony należy ominąć przeszkodę a następnie powrócić na trasę i kontynuować jazdę.

Testy potwierdziły, że robot jest w stanie bezbłędnie pokonywać przedstawioną na Rys.5.1 trasę oraz omijać przeszkody napotkane na mniej złożonych fragmentach toru. Czas trwania przejazdu robota po trasie pozbawionej przeszkód wynosi przeciętnie 24 sekundy. Otrzymany wynik jest satysfakcjonujący dla robota, którego zbudowano z wykorzystaniem niewielkich nakładów finansowych.

5.3 Testowanie aplikacji mobilnej

Opracowaną na potrzeby projektu aplikację zainstalowano na kilku smartfonach z systemem operacyjnym Android. Podczas testowania aplikacji mobilnej sprawdzono:

- parowanie smartfona z modułem Bluetooth,
- wydawanie robotowi poleceń za pomocą przycisków,
- wydawanie robotowi poleceń głosowych.

Aplikacja mobilna została przetestowana przez kilku użytkowników. Podczas testów nie zarejestrowano nieprawidłowości związanych z funkcjonowaniem aplikacji. Robot bezbłędnie wykonywał polecenia wydawane przez użytkowników. Proces komunikacji pomiędzy smartfonem a przymocowanym do robota modułem Bluetooth przebiegał bez zakłóceń. Po zakończeniu testów poproszono użytkowników o ocenę poszczególnych elementów aplikacji w skali od 1 do 5. Interfejs graficzny aplikacji oceniono jako intuicyjny i przejrzysty.

Tabela 5.1 Ocena aplikacji mobilnej w opinii użytkowników

	Sterowanie	**Przejrzystość GUI**	**Uwagi**
Użytkownik A	5	5	-
Użytkownik B	5	4	Brak możliwości wydawania poleceń głosowych w języku polskim
Użytkownik C	5	4	-
Użytkownik D	5	5	-

6. Wnioski i uwagi końcowe

W ramach projektu zbudowano w pełni funkcjonalnego robota z wykorzystaniem niewielkich nakładów finansowych. Przybliżony koszt budowy urządzenia wyniósł około stu trzydziestu złotych. Średni czas pokonania trasy (Rys. 5.1.) przez robota to 24 sekundy. Trasa mierzy w przybliżeniu siedem metrów i składa się z osiemnastu ostrych i trzech łagodnych zakrętów. Uzyskany wynik jest satysfakcjonujący dla urządzenia wykonanego z tanich, ogólnodostępnych komponentów. Robot jest w stanie omijać przeszkody znajdujące się na mniej złożonych odcinkach trasy. Pojazd może być również zdalnie sterowany przy pomocy aplikacji mobilnej. Wszystkie założenia projektowe zostały spełnione.

Konstrukcja robota umożliwia wprowadzanie ulepszeń i dodatkowych funkcji w przyszłości. Wykorzystane w projekcie komponenty mogą zostać zamienione na bardziej zaawansowane technologicznie odpowiedniki. W celu uzyskania lepszego rezultatu wykrywania czarnej linii, należy zwiększyć liczbę odbiciowych czujników podczerwieni lub zastąpić je kamerą. Po wprowadzeniu powyższych rozwiązań, możliwe będzie zastosowanie złożonych algorytmów decyzyjnych. Aby skrócić średni czas pokonywania trasy, silniki powinny zostać zamienione na odpowiedniki o większej liczbie obrotów na minutę. Konstrukcja mechaniczna robota musi być lekka, aby zmniejszyć obciążenie silników. Wprowadzenie powyższych ulepszeń jest kosztowne, ale niezbędne w przypadku chęci uczestnictwa w konkursie. Praktyczne umiejętności nabyte w trakcie realizacji projektu będą przydatne podczas konstruowania bardziej złożonych urządzeń.

Bibliografia

[1] A. Morecki, J.Knapczyk: *Podstawy robotyki*. Wydawnictwo Naukowo-Techniczne, Warszawa 1999.

[2] Banzi M., Chaniewska M., *Wprowadzenie do Arduino*, APN Promise, Warszawa, 2014.

[3] Cook, D., *Intermediate robot building*, [Berkeley, Calif.] 2010.

[4] Infrared Sensor - How it Works, Types, Applications, Advantage & Disadvantage, https://electricalfundablog.com/infrared-sensor/ [18.11.2021]

[5] Gajek A., Juda Z., *Czujniki*, Wydawnictwa Komunikacji i Łączności, Warszawa, 2011.

[6] L298N Motor Driver Module, https://components101.com/modules/l293n-motor-driver-module [30.11.2021]

[7] Amariei C., *Arduino development cookbook*, Packt Publ, Birmingham, 2015.

[8] Arduino Uno Rev3, https://docs.arduino.cc/hardware/uno-rev3/ [05.12.2021]

[9] Holmes D., Lipo T., *Pulse width modulation for power converters*, IEEE, Pisacataway, N.J., 2003.

[10] Firoozian R., *Servo motors and industrial control theory*, Springer, Cham [etc.], 2014.

[11] Baichtal J., Matuk K., *Fascynujący świat robotów*, Helion, Gliwice, 2015.

[12] Bloom, J., *Exploring Arduino: Tools and Techniques for Engineering Wizardry*, Wiley, 2018.

Załączniki

1. Program mikrokontrolera – projekt środowiska Arduino IDE
   - Line_Follower.ino
2. Program aplikacji mobilnej – projekt środowiska MIT App Inventor
   - Line_Follower_Controller.aia

Spis rysunków

Spis tabel