



**WYDZIAŁ  
ELEKTROTECHNIKI  
I INFORMATYKI**  
POLITECHNIKI RZESZOWSKIEJ

**Damian Bielecki**

Implementacja sterowania hierarchicznego mobilnym robotem  
kołowym z wykorzystaniem języka Python

**Praca dyplomowa inżynierska**

Opiekun pracy:  
dr. Paweł Penar

Rzeszów, 2023



# Spis treści

<b>1. Wprowadzenie</b>	<b>5</b>
<b>2. Przegląd literatury</b>	<b>6</b>
2.1. Przegląd istniejących rozwiązań	6
2.2. Opis przyjętego rozwiązania	6
<b>3. Algorytm A*</b>	<b>8</b>
<b>4. Implementacja algorytmu i środowiska testowego</b>	<b>9</b>
<b>5. Projekt robota</b>	<b>10</b>
5.1. Założenia projektowe	10
5.2. Projektowanie zarysu robota w środowisku CAD	10
5.3. Projekt, wykonanie oraz podłączenie elektroniki robota	12
5.4. Oprogramowanie robota	15
5.4.1. Sterowanie silnikami	15
5.4.2. Serwer TCP	16
<b>6. Przeprowadzone testy</b>	<b>17</b>
<b>Literatura</b>	<b>18</b>



# 1. Wprowadzenie

Od lat można zauważyć zwiększającą się popularność różnych robotów, które bazując na wprowadzonej do systemu mapie autonomicznie poruszają się po terenie tak aby osiągnąć określony cel. Celem projektu inżynierskiego będzie zaimplementowanie algorytmu wyszukującym najkrótszą ścieżkę. Do przetestowania algorytmu zostanie napisany prosty program symulujący robota poruszającego się po wyznaczonej ścieżce. Kolejnym etap projektu to zbudowanie robota mającego zweryfikować zaproponowaną przez algorytm ścieżkę.

Głównym powodem podjęcia się implementacji takiego algorytmu jest chęć zapoznania się z algorytmem planującym ścieżkę i próba zaimplementowania takiego rozwiązania na fizycznym robocie.

## **Nazakres pracy składa się:**

- Przegląd literatury
- Implementacja algorytmu A\* w języku Python
- Symulacja działania pracy algorytmu
- Weryfikacja na obiekcie rzeczywistym

## **Omówienie rozdziałów**

Rozdział pierwszy - zawiera przegląd literatury oraz istniejących rozwiązań. Na ich podstawie określona zostanie ogólna struktura własnego rozwiązania.

Rozdział drugi - skupia się na genezie, charakterystyce i opisie działania wybranego algorytmu wyszukiwania najkrótszej ścieżki. Dodatkowo zostaną przedstawione inne algorytmy zwiększającą autonomię robota mobilnego.

Rozdział trzeci - w tym rozdziale zostanie przedstawione napisany algorytm oraz środowisko symulacyjne w języku Python.

Rozdział czwarty - omawia budowę robota mobilnego oraz jego program sterujący. W szczególności skupiono się na segment komunikacji oraz regulatorze PID sterującym silnikami.

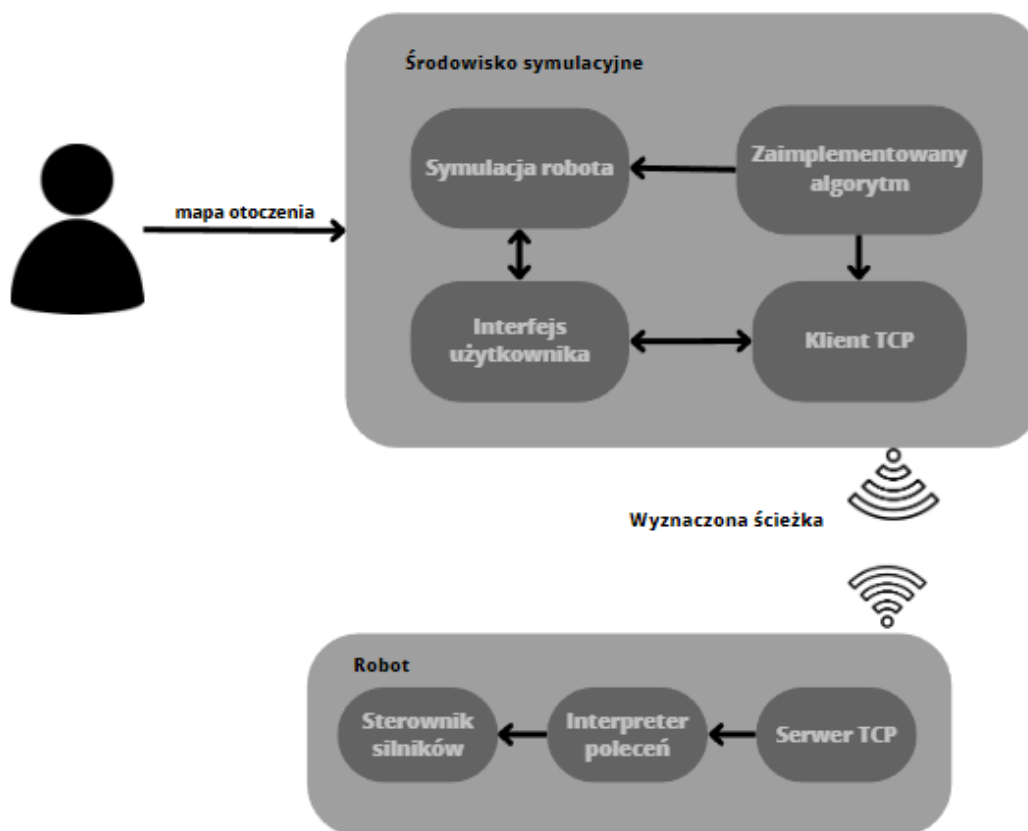
Rozdział piąty - przedstawia przeprowadzone testy wykonane w środowisku symulacyjnym oraz rzeczywistym.

## 2. Przegląd literatury

### 2.1. Przegląd istniejących rozwiązań

### 2.2. Opis przyjętego rozwiązania

Algorytm wyznaczania najkrótszej ścieżki wraz z środowiskiem testowym zostało napisane w języku Python. Implementacja algorytmu nie wymaga żadnych dodatkowych bibliotek. Do zbudowania środowiska symulacyjnego zostały użyte biblioteki graficzne do rysowania kształtów geometrycznych oraz podstawowych elementów graficznego interfejsu użytkownika. Weryfikacja działania algorytmu została przeprowadzona na zbudowanym trójkątowym robocie. Oprogramowanie robota zostało napisane w języku C++ i dodatkowych bibliotekach udostępnionych przez producentów użytych sterowników.



Rysunek 2.1: Ogólny schemat projektu

Użytkownik po uruchomieniu środowiska symulacyjnego będzie miał możliwość

edycji mapy opartej na siatce, określenia punktu początkowego i końcowego. Kolejnym krokiem jest wyznaczenie ścieżki oraz jej graficzną implementację. Tak wyznaczoną ścieżkę można wysłać do robota operującego w środowisku rzeczywistym.

### 3. Algorytm A\*



## 4. Implementacja algorytmu i środowiska testowego

fdfdd

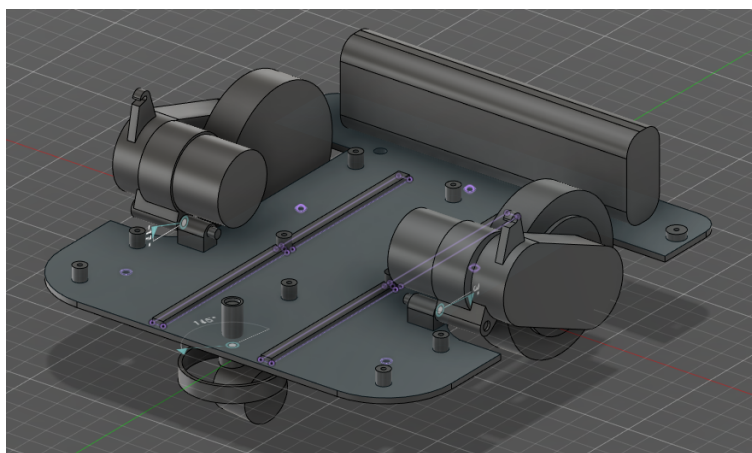
## 5. Projekt robota

### 5.1. Założenia projektowe

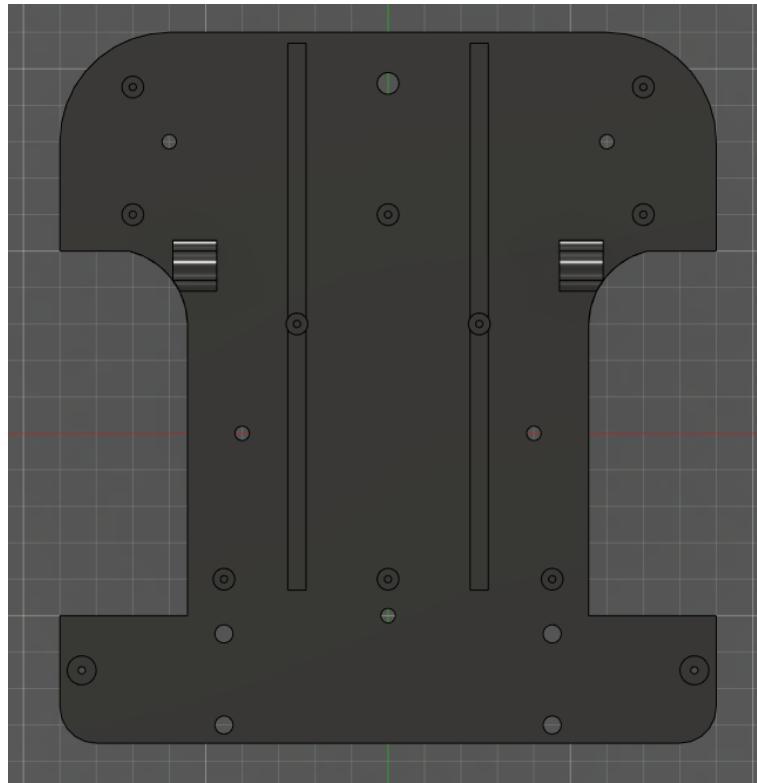
Wykonany robot powinien być jak najmniejszy i najprostszy w wykonaniu oraz sterowaniu tak aby móc przetestować przy jego pomocy działanie algorytmu A\*. Robot będzie zbudowany z platformy, do której zostaną przyczepiony napęd, elektronika sterująca oraz bateria. Do platformy zostaną przymocowane dwa gotowe moduły napędowe składające się z silnika, przekładni oraz dużego koła. Aby pojazd stał stabilnie, doczepione zostanie trzecie koło obracające się swobodnie w każdym kierunku. Całość będzie sterowana przy pomocy mikrokontrolera ESP32 oraz dwukanałowym sterownikiem silników DC opartym na układzie L298n. Za zasilanie będzie odpowiadała litowo-jonowy akumulator 4S.

### 5.2. Projektowanie zarysu robota w środowisku CAD

Zbudowany ma być prosty w budowie i wykonaniu a więc założyłem że podstawa utrzymująca pozostałe komponenty zostanie wydrukowana na drukarce 3D. Model platformy i pozostałych posiadanych elementów został wykonany w programie Fusion 360. Modele modułów napędowych, przedniego kółka oraz baterii pozwoliły na optymalne wyznaczenie pod względem wielkości lokalizacji wszystkich elementów.

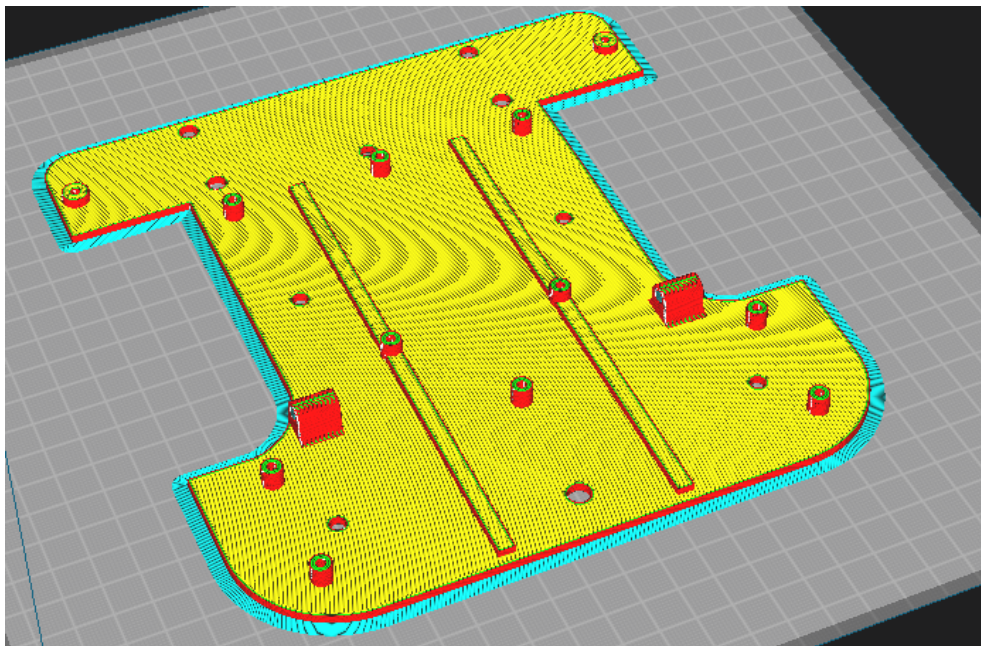


Rysunek 5.2: Zaprojektowane podwozie robota w programie Fusion360



Rysunek 5.3: Widok z góry na podwozie robota

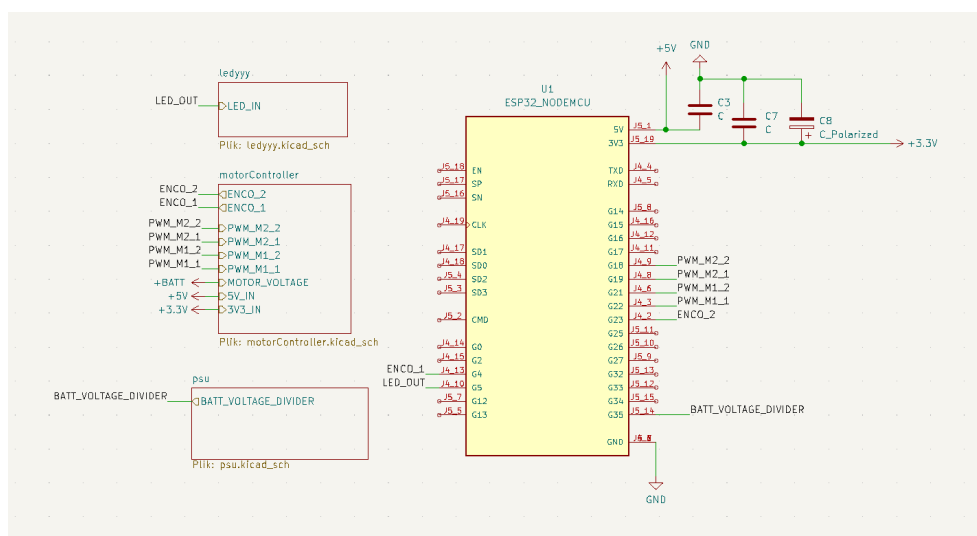
Model został przygotowany do druku w programie Ultimaker Cura. Parametry druku zostały dobrane eksperymentalnie na podstawie własnych doświadczeń. Podstawę wydrukowano z PLA w temperaturze 220°C i wysokości warstwy 0,2mm. Żeby zwiększyć wytrzymałość temperatura głowicy została lekko zawyżona względem wymagań producenta filamentu



Rysunek 5.4: Widok przygotowanego do druku modelu

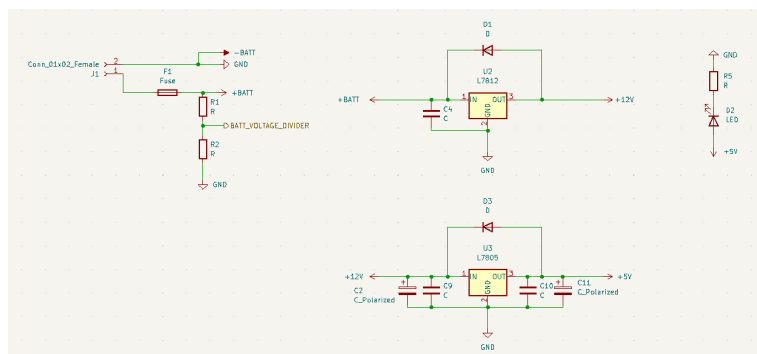
### 5.3. Projekt, wykonanie oraz podłączenie elektroniki robota

Ze względu na chęć bezprzewodowego sterowania robotem zostanie wykorzystany moduł ESP32, dla którego zostanie przygotowana odpowiednia płytki z wyprowadzeniami do enkoderów silnika oraz ich sterownika. Schemat elektroniczny i projekt pcb został wykonany w programie KiCad.



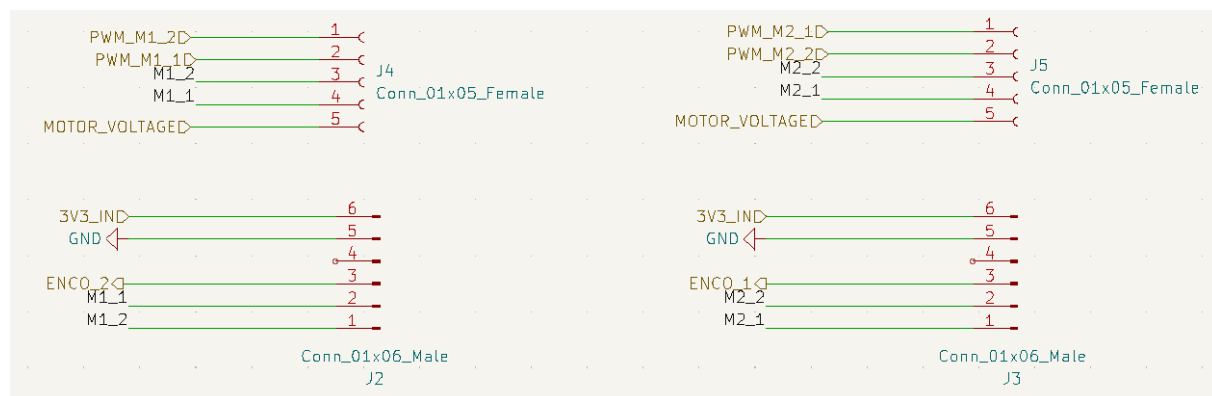
Rysunek 5.5: Ogólny schemat połączeń

Wszystkie potrzebne elementy podsystemy mikrokontrolera zostały już wlutowane w module deweloperskim a więc mój schemat zawiera jedynie odpowiednie połączenia z modułami roboczymi.



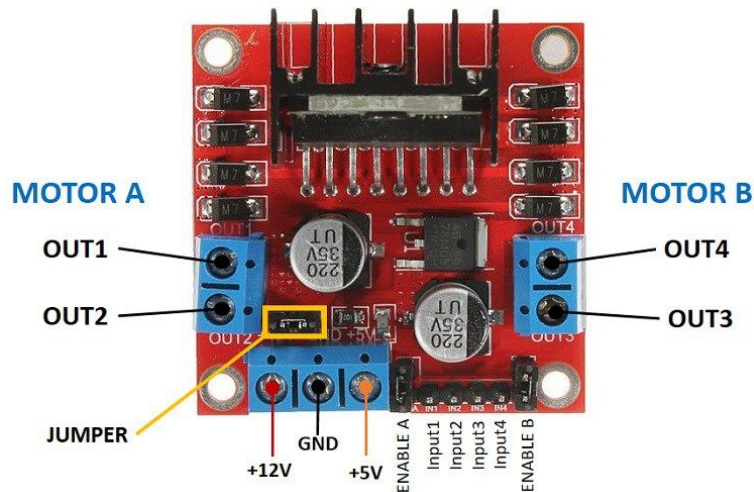
Rysunek 5.6: Ogólny schemat połączeń

Użyta bateria posiada napięcie maksymalne 16,8V a więc musi zostać odpowiednio zmniejszone przed podaniem go na piny zasilania. Ze względu na wykorzystywaną łączność bezprzewodową a więc zwiększone zużycie prądu zmniejszanie napięcia zostało podzielone na dwie sekcje. Najpierw zmniejszane jest poprzez stabilizator LM7812 z 16.8V do 12V a następnie poprzez LM7805 z 12V napięcie redukowane jest do 5V. ESP32 zasilane jest napięciem 3.3V tworzonym poprzez stabilizator AMS1117 w module deweloperskim. Aby zredukować spadki napięć powstałe przy nagłym poborze prądu dodałem dodatkowe kondensatory filtrujące. Dodatkowo na schemacie widoczny jest dzielnik napięcia pozwalający na poziom naładowania baterii przez sterownik jednak ostatecznie nie został on wykorzystany w projekcie.

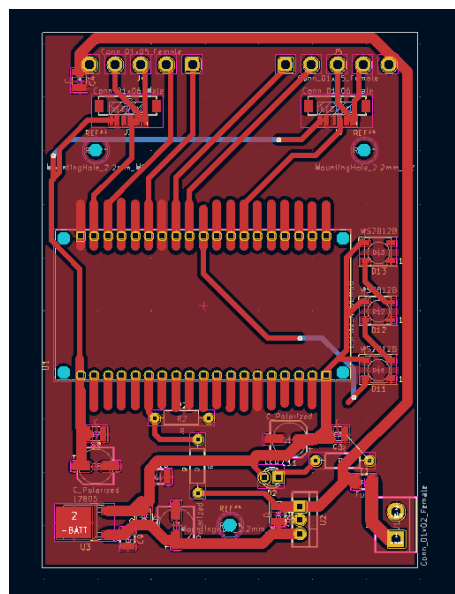


Rysunek 5.7: Ogólny schemat połączeń silników

Użyte moduły napędowe posiadają enkoder inkrementalny zbudowany z czujnika halla i tarczy magnetycznej zamocowanej na wale silnika. Czujnik halla jest zasilany napięciem 3.3V i na wyjściu otrzymujemy sygnał ze szpilkami"proporcjonalny do aktualnych obrotów silnika. Silniki zasilane są poprzez moduł z układem L298n a prędkość ustalana jest poprzez odpowiednio generowany sygnał PWM.



Rysunek 5.8: Wykorzystany moduł do sterowania silnikami, L298n



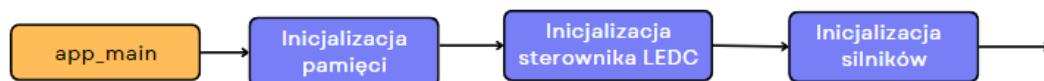
Rysunek 5.9: Projekt PCB

Na widocznym powyżej zdjęciu widać ostateczną wersję projektu pcb. Można zauważyć że widoczne są na niej trzy adresowalne diody świecące, które nie zostały

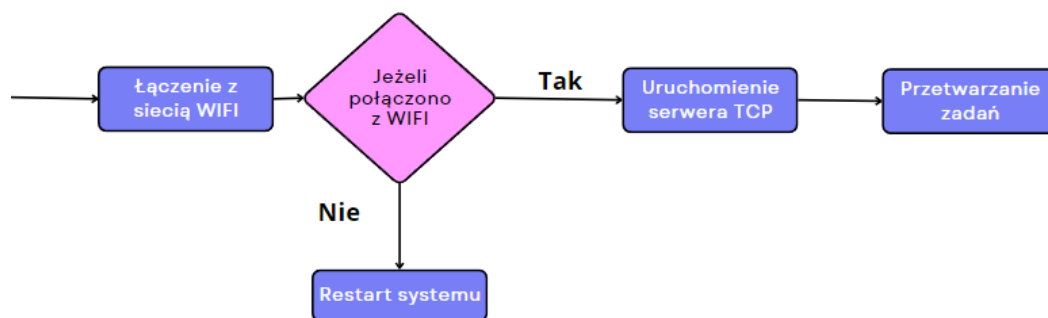
wykorzystywane przez napisane oprogramowanie.

## 5.4. Oprogramowanie robota

Program sterujący robotem został napisany w C++. Szkielet aplikacji bazuje na projekcie utworzonym przez framework IDF w wersji 4.4 udostępnionym przez producenta użytego procesora. Po za tym użyta została biblioteka implementująca system czasu rzeczywistego FreeRTOS i ASIO do obsługi połączenia TCP.



Rysunek 5.10: Schemat programu cz.1



Rysunek 5.11: Schemat programu cz.2

Działanie programu rozpoczyna się od wywołania funkcji `app_main`, inicjalizacji funkcji systemowych i sterowników silników. W dalszej kolejności nawiązywana jest łączność z siecią WiFi. W przypadku braku połączenia system resetuje się. Po poprawnym połączeniu uruchamiany jest kontekst biblioteki ASIO a następnie uruchomienie serwera TCP. Klienci połączeniu do serwera utworzonego przez robota wysyłają komendy tekstowe wraz z odpowiednimi argumentami, które robot odpowiednio przetwarza.

### 5.4.1. Sterowanie silnikami

Sterowanie silnikami odbywa się poprzez klasę `MotorController`, która dziedziczy po klasie `PIDController` implementującej regulator PID. Obiekt automatycznie

tworzy timer uruchamiający co 100ms metodę aktualizującą wyjścia sterujące silnikiem. Aktualna prędkość wyznaczana jest na podstawie przerwania wyzwalanego przez enkoder silnika. Przerwanie inkrementuje licznik, czyszczony przez wcześniej opisany timer. Nastawy regulatora PID zostały dobrane eksperymentalnie, człon proporcjonalny wynosi 8 a całkujący i różniczkujący 0,1.

#### **5.4.2. Serwer TCP**



## 6. Przeprowadzone testy

## Literatura

- [1] <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/index.html>. Dostęp 04.01.2023.

**STRESZCZENIE PRACY DYPLOMOWEJ INŻYNIERSKIEJ**  
**IMPLEMENTACJA STEROWANIA HIERARCHICZNEGO**  
**MOBILNYM ROBOTEM KOŁOWYM Z WYKORZYSTANIEM**  
**JĘZYKA PYTHON**

Autor: Damian Bielecki, nr albumu: ME-163461

Opiekun: dr. Paweł Penar

Słowa kluczowe: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po polsku

**BSC THESIS ABSTRACT**  
**TEMAT PRACY PO ANGIELSKU**

Author: Damian Bielecki, nr albumu: ME-163461

Supervisor: (academic degree) Imię i nazwisko opiekuna

Key words: (max. 5 słów kluczowych w 2 wierszach, oddzielanych przecinkami)

Treść streszczenia po angielsku