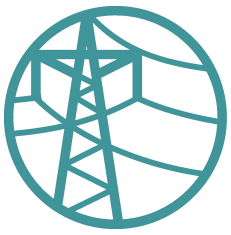


WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ

Damian Bielecki

Implementacja sterowania hierarchicznego mobilnym robotem kołowym z wykorzystaniem języka Python

Praca dyplomowa inżynierska

Opiekun pracy:
dr. Paweł Penar

Rzeszów, 2023

Spis treści

1. Wprowadzenie

Od lat można zauważyć zwiększająca 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 ścieżke 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 autonomie 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.

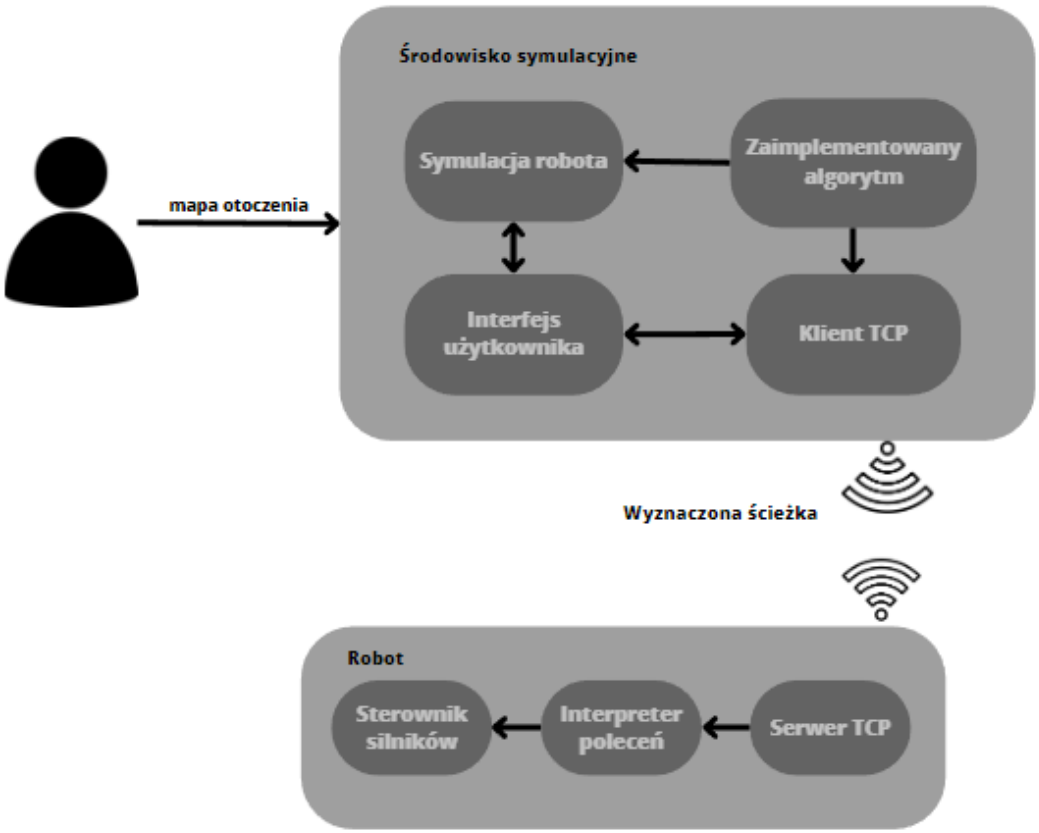
Rozdział piąty - przedstawia przeprowadzone testy wykonane w środowisku symulacyjnym oraz rzeczywistym.

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 trójkołowym 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 implementacje. Tak wyznaczoną ścieżkę można wysłać do robota operującego w środowisku rzeczywistym.

3. Algorytm A*

3.1. Geneza powstania

Algorytm A* powstał w ramach projektu Shakey, zapoczątkowanego w 1966 roku przez Charles Rosen'a. Celem projektu było zbudowanie robota, który potrafiłby planować własne działania. Zbudowany robot wyróżniał się na tle innych tym że intergował kilka różnych modeli sztuczenj inteligencji pracujących jako jeden system.

Robot był zbudowany z: - kamery telewizyjnej i dalmierza optycznego - system wizyjny do obserwacji środowiska - łącze radiowe - służącego do komunikacji z bazą, odbierania i wysyłania komend - detektor uderzeń - pozwalający na zatrzymanie robota w przypadku kolizji

Komunikacja odbywała się poprzez wysyłane radiowo tekstowe polecenia mające określoną strukturę np.: GOTO D4 - co oznaczało automatyczne przemieszczenie się robota do wskazanej pozycji

Rysunek 3.2: Robot Shakey i Charles Rosen, inicjator projektu [2]

3.2. Opis działania algorytmu

A* to heurestyczny algorytm wyznaczający najkrótszą możliwą ścieżke w grafie. Jest to algorytm zupełny i optymalny a więc zawsze zostanie wyznaczona optymalne rozwiązanie. Ze względu na przeszukiwanie oparte na grafie algorytm najlepiej działa

na strukturze drzewiastej. Zadaniem algorytmu jest minimalizacja funkcji:

$$f(x) = h(x) + g(x) \tag{3.1}$$

gdzie: $f(x)$ - minimalizowana funkcja, $g(x)$ - to rzeczywisty koszt dojścia do punktu x.

Funckja $h(x)$ to funkcja heurestyczna oszacowująca ona koszt dotarcia od punktu x do wierzchołka docelowego

Zalety:

- jest kompletny i optymalny
- może przeszukować skomplikowane mapy
- jest najwydajnieszym takim algorytmem

Wady:

- jego wydajność w znacznej mierze zależy od funkcji heurestycznej
- Każda akcja ma stały koszt wykonania
- nie nadaje się do często zmieniającego się otoczenia robota, wymaga ponownego przeliczenia

Przykładowe funkcje heurestyczne:

- Funkcja euklidesowa

$$h(x) = 10 * \sqrt[2]{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{3.2}$$

- Geometria Manhattanu (innaczej metryka miejska)

$$h(x) = |x_2 - x_1| + |y_2 - y_1| \tag{3.3}$$

Gdzie: x_1 i y_1 to współrzędne wyznaczanego punktu, x_2 i y_2 to koordynaty celu

4. Implementacja

4.1. Implementacja algorytmu

Mechanizm wyszukiwania najkrótszej ścieżki został zamknięty w jednym module o nazwie aStar. Na moduł składa się klasa AStar ze wszystkimi potrzebnymi metodami oraz klasa Node reprezentująca pojedynczy punkt przeszukiwanego grafu.

Wyznaczanie najkrótszej ścieżki rozpoczyna się od wyznaczenie kosztu punktu startowego, utworzenie zbioru z nieprzeszukanymi wierzchołkami, do którego dopisujemy punkt początkowy.

```
1     openList = []
2     startNode.h = self.heurestic(startNode.getCords(), endNode.
    getCords())
3     startNode.g = 0
4     heappush(openList, startNode)
```

Listing 1: Przygotowanie danych

Wewnętrzna funkcja heurestic przyjmująca pozycje dwóch punktów odpowiada za wyliczenie optymistycznego kosztu przejścia od punktu x do wierzchołka docelowego. Takie podejście pozwala na szybką podmianę funkcji bez znaczących zmian w programie. Wykorzystywana funkcja heurestyczna to równanie (3.2)

W kolejnym kroku uruchamiana jest pętla, która wykonywana jest dopóki w zbiorze otwartym znajdują się nie odwiedzone elementy. Z listy pobierany jest element o najmniejszej liczbie punktów co oznacza że dany wierzchołek drzewa jest bliżej od pozostałych. Jeżeli pobrany element nie jest celem to pobierani są wszyscy jego sąsiedzi.

4.2. Implementacja środowiska testowego

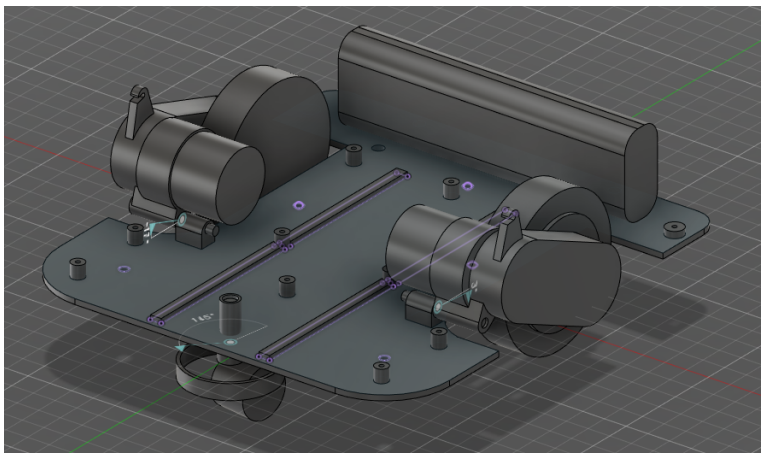
5. Projekt robota

5.1. Założenia projektowe

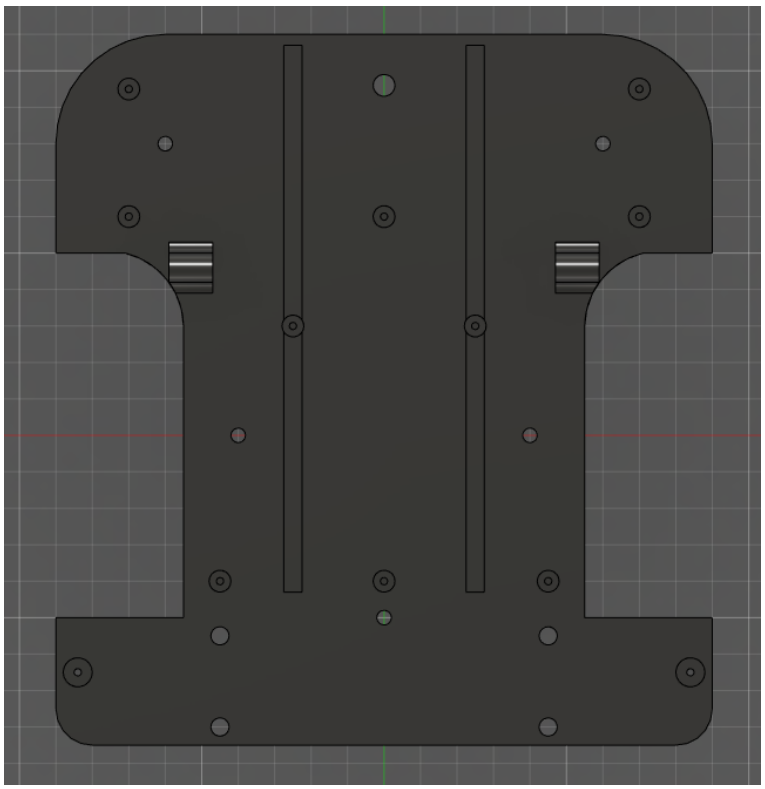
Wykonany robot powinien być jak najmniejszy i najprostszy w wykonaniu oraz sterowaniu tak aby móc przetesotwać przy jego pomocy działanie algorytmu A*. Robot będzie zbudowany z platformy, do której zostaną przyczepiony napęd, elektronika sterująca oraz bateria. Do platformy zostaną przymocowane dwa gotowe 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 utrzymujące 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.

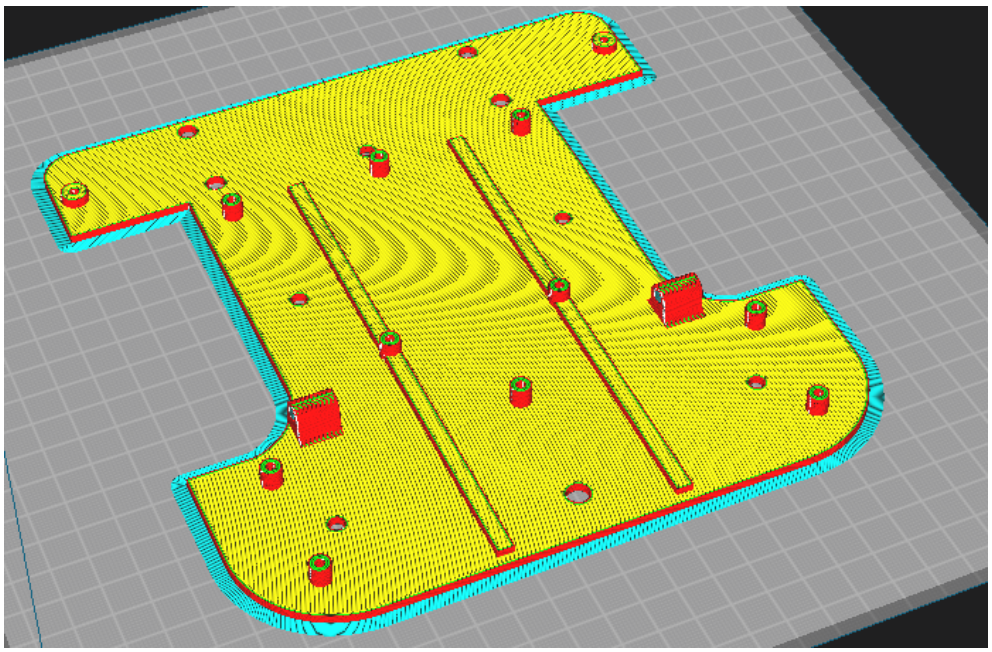


Rysunek 5.3: Zaprojektowane podwozie robota w programie Fusion360



Rysunek 5.4: Widok z góry na podwozie robota

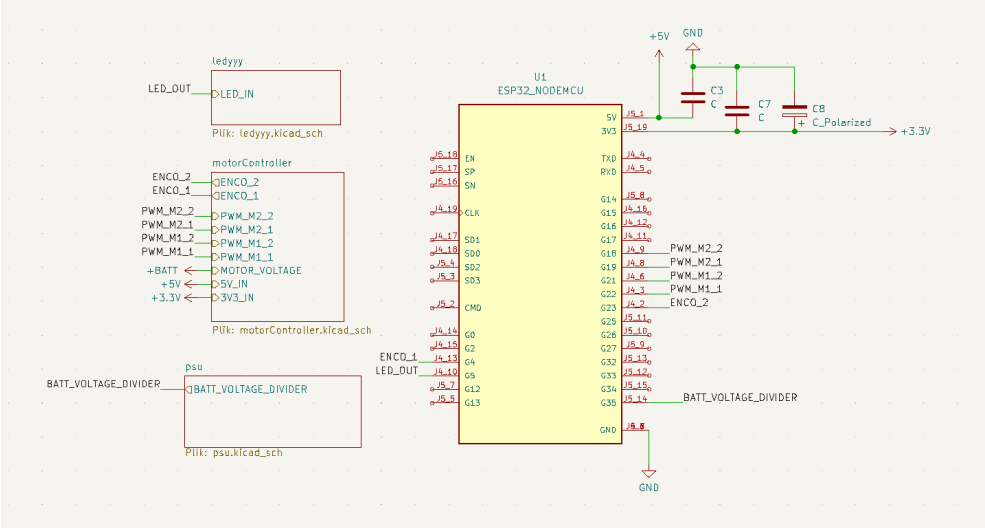
Model został przygotowany do druku w programie Ultimaker Cura. Parametry druku zostały dobrane eksperymentalnie na podstawie własnych doświadczeń. Podstawę wydrukowano z PLA w temperaturze 220*C i wysokości warstwy 0,2mm. Żeby zwiększyć wytrzymałość temperatura głowicy została lekko zawyżona względem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



Rysunek 5.5: Widok przygotowanego do druku modelu

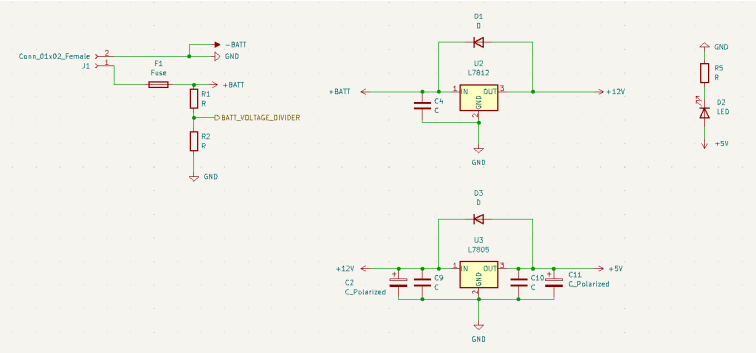
5.3. Projekt, wykonanie oraz podłączenie elektroniki robota

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



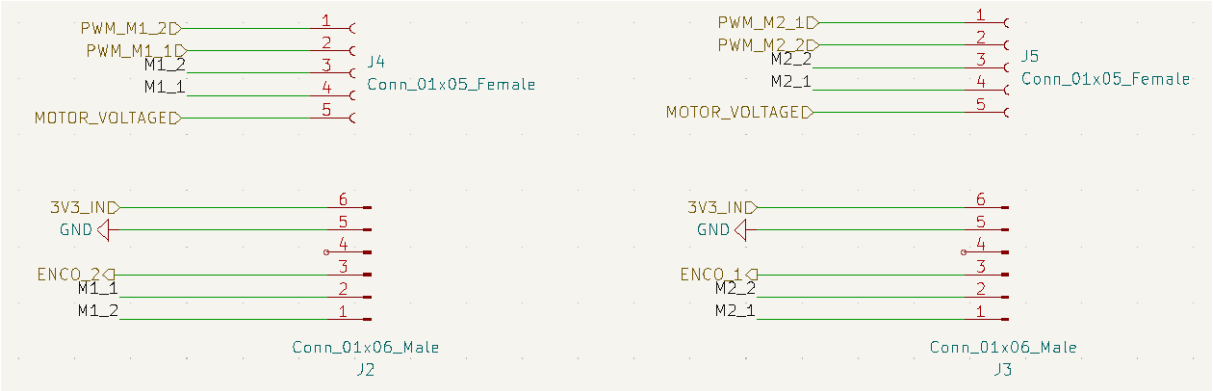
Rysunek 5.6: Ogólny schemat połączeń

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



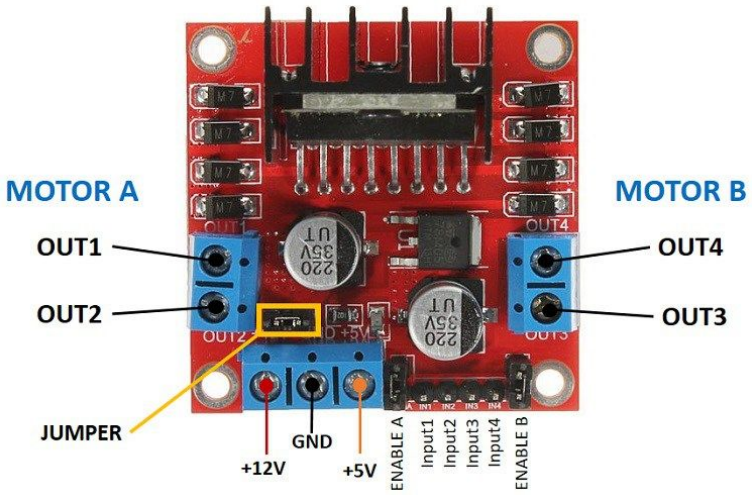
Rysunek 5.7: Ogólny schemat połączeń

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.

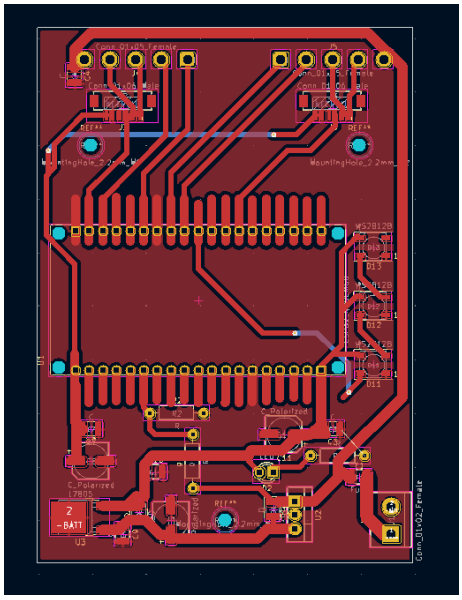


Rysunek 5.8: Ogólny schemat połączeń silników

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 śzpilkami"proporcjonalny do aktualnych obrotów silnika. Silniki zasilane są poprzez moduł z układem L298n a prędkość ustalana jest poprzez odpowiednio generowany sygnał PWM.



Rysunek 5.9: Wykorzystany moduł do sterowania silnikami, L298n



Rysunek 5.10: Projekt PCB

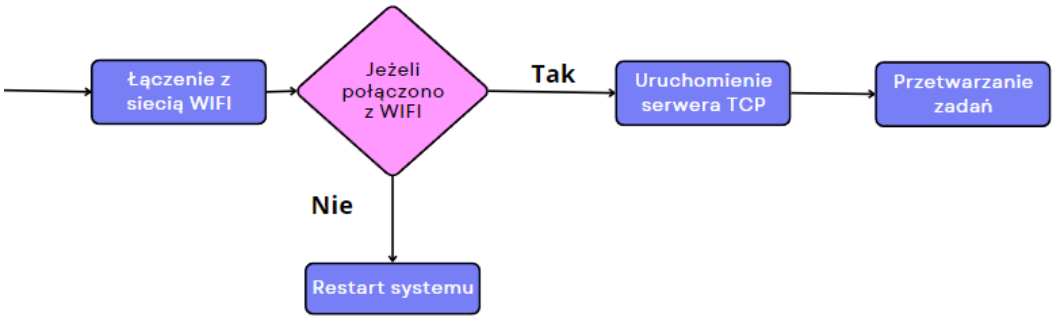
Na widocznym powyżej zdjęciu widać ostateczną wersje projektu pcb. Można zauważyć że widoczne są na niej trzy adresowalne diody świecące, które nie zostały

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 5.11: Schemat programu cz.1



Rysunek 5.12: 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 utowrzonego przez robota wysyłają komendy tekstowe wraz z odpowiednimi arguemntami, 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 eksperymentalinie, 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.

[2] https://medium.com/dish/75-years-of-innovation-shakey-the-robot-385af2311ec8 Dostęp 04.01.2023.

POLITECHNIKA RZESZOWSKA im. I. Łukasiewicza Rzeszów, 2023
Wydział Elektrotechniki i Informatyki

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

RZESZOW UNIVERSITY OF TECHNOLOGY Rzeszow, 2023
Faculty of Electrical and Computer Engineering

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