POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

KIERUNEK AUTOMATYKA I ROBOTYKA

Projekt inżynierski

System sterowania robotem mobilnym

Autor: Jakub Kaniowski

Kierujący pracą: dr inż. Krzysztof Jaskot

Gliwice, styczeń 2022

Spis treści

Streszczenie

Realizowana praca dyplomowa dotyczy dziedziny robotyki. Zadaniem autora pracy jest skonstruowanie i wykonanie rzeczywistego, zdalnie sterowanego pojazdu, wykorzystującego koła szwedzkie, które umożliwią ruch robota mobilnego w wielu kierunkach.

Prezentowana praca inżynierska ma charakter interdyscyplinarny. Autor w sposób chronologiczny prezentuje poszczególne etapy realizacji pracy (od planowania, poprzez projektowanie, wykonywanie i programowanie urządzenia). Końcowym wynikiem pracy ma być układ sterowania robotem czterokołowym (koła typu szwedzkiego), z wykorzystaniem ogólnodostępnych narzędzi programistycznych i urządzeń pomiarowych.

Słowa kluczowe: robot mobilny, mecanum, koła szwedzkie, koła mecanum, system sterowania robotem

Rozdział 1

Wstęp

System sterowania robotem mobilnym, to projekt inżynierski realizowany w ramach samorozwoju i pogłębiania własnych zainteresowań z dziedziny robotyki, informatyki i elektroniki. Jest to projekt wymagający dużego zaangażowania, z uwagi na złożoność zagadnień z tychże dziedzin.

Robotyka jest bardzo dynamicznie rozwijającą się, interdyscyplinarną dziedziną nauki, zawierającą tematy z obszaru elektroniki, mechaniki i informatyki. Dzięki prężnemu rozwojowi przemysłu, zwłaszcza w XIX i XX wieku i co za tym idzie - rozwojowi technologii - stworzono możliwość tworzenia urządzeń elektro-mechanicznych, służących do automatyzacji czynności manualnych często zastępujących człowieka oraz wykonywania innych zadań powierzonych przez operatora urządzenia.

Początek robotyki, można datować na połowę lat pięćdziesiątych ubiegłego wieku - konkretniej na rok 1954, w którym to złożono pierwszy wniosek patentowy na manipulator stworzony przez Georga C. Devola, który został wykorzystany w automatyzacji produkcji dla koncernu motoryzacyjnego General Motors [1].

Z uwagi na szerokie spektrum zastosowań robotyki w otaczającym nas świecie, może zajść potrzeba stworzenia urządzeń o zupełnie odmiennej konstrukcji i stopniu skomplikowania – który zależy od złożoności zadania jakie urządzenia mają wykonać. W robotyce istnieje pewien wewnętrzny podział, który klasyfikuje działalność naukową i konstrukcyjną. Można wyróżnić następujące części:

- robotyka medyczna - działalność związana z tworzeniem robotów chirurgicznych i rehabilitacyjnych; szeroko pojęta integracja człowieka z maszyną – poczynając od zaawansowanych protez, kończąc na egzoszkieletach wspomaganych

- robotyka przemysłowa - działalność związana z automatyzacją przemysłu, podnoszeniu jakości i powtarzalności danego procesu technologicznego, sprowadzającego się do tworzenia robotów lub manipulatorów przemysłowych
- robotyka mobilna - szeroko pojęta działalność związana z budową urządzeń mogących zmieniać swoje położenie w przestrzeni, w sposób autonomiczny lub sterowany przez człowieka – bez jego bezpośredniego udziału w np.: przeniesieniu maszyny (np. roboty kołowe mobilne, drony).

Należy wspomnieć, iż wymienione powyżej dziedziny robotyki, mają pewne części wspólne, chociażby kwestie teorii robotyki – kinematyki, opisu matematycznego itp., zatem nie można traktować tej klasyfikacji w sposób bardzo rygorystyczny.

Roboty kołowe mobilne, z biegiem czasu znacznie się spopularyzowały. Wynika to głównie z powszechnego dostępu do wiedzy i spadku cen wykorzystywanej w nich elektroniki. W obecnym czasie, popularyzacja tych robotów umożliwiła wprowadzenie ich do gospodarstw domowych w postaci odkurzaczy, kosiarek autonomicznych lub zabawek edukacyjnych. Należy dodać, że większość powszechnie znanych robotów, bazuje na napędach gąsienicowych lub tradycyjnych kołach (np. opartych na mechanizmie różnicowym i/lub przynajmniej jednej osi skrętnej), z kolei temat robotów wyposażonych w koła szwedzkie, jest znacznie mniej spopularyzowany i powszechny dla większego grona odbiorców.

Opisywana praca inżynierska, dotyczy robotyki mobilnej kołowej. Autor projektu zamierza stworzyć robota, wyposażonego w koła szwedzkie (znane pod inną nazwą jako Mecanum).

W nomenklaturze naukowej, takie roboty należą do klasy mobilności (3,0) - zwane są omnimobilnymi, z uwagi na pełną swobodę ruchu po płaszczyźnie – oznacza to, iż mają one zdolność do ruchu w każdym kierunku nie wymagając przy tym reorientacji [2]. Równie często spotyka się pojęcie holonomiczności robota, które jest związane z sposobem reorientacji. Jeśli robot jest w stanie zmienić swoją orientacje nie zmieniając położenia (tj. obrót wokół geometrycznego środka pojazdu), oznacza to, iż jest robotem holonomicznym. Robotem nieholonomicznym nazywamy z kolei pojazdy, które muszą wykonać serię manewrów, aby zmienić orientacje - do takich należą roboty typu „samochód” - posiadają klasę mobilności (1,1) co oznacza zdolność ruchu w jednym kierunku, przy posiadaniu jednej osi skrętnej – odpowiadającą za kierunek ruchu.

1.1 Cel i zakres projektu

Celem tej pracy, jest wykonanie od podstaw robota mobilnego kołowego, o wysokiej klasie mobilności (3,0), zawierającego czujniki pozwalające w przyszłości przeanalizować kierunek ruchu lub orientację w przestrzeni. Projektowany robot powinien umożliwiać jazdę w każdym kierunku, w tym „w bok" oraz „na skos" bez reorientacji pojazdu, z kolei sama reorientacja robota, powinna być możliwa bez zmiany położenia pojazdu. Odczyty z czujników powinny być dostępne zdalnie, dla operatora – zatem, informacje o zgromadzonych danych powinny zostać wysyłane do użytkownika na bieżąco.

Niniejszy projekt, będzie składał się z dwóch części. Części fizycznej urządzenia – tj. projekt i wykonanie samego robota (platformy jezdnej), elektroniki, mechaniki, oraz części programowej – tj. implementacji odpowiedniego oprogramowania, umożliwiającego sterowanie robotem.

Dodatkowym celem tej pracy, jest pogłębienie umiejętności z dziedziny projektowania CAD (ang. *computer-aided design*), doboru elementów elektronicznych spełniających założenia oraz programowania układów mikroprocesorowych.

Rozdział 2

Założenia projektu

2.1 Założenia wstępne

Projekt systemu sterowania robotem z kołami szwedzkimi zakłada wykonanie platformy mobilnej w taki sposób, aby umożliwiała ona jazdę we wszystkich możliwych kierunkach, dobór odpowiednich komponentów elektronicznych umożliwiających sterowanie skonstruowanym robotem oraz wykonanie prostego oprogramowania, umożliwiającego zdalne sterowanie urządzeniem.

Można zatem podzielić etap założeń na trzy elementy: konstrukcję robota, elektronikę pokładową oraz program sterownika. Szczegółowe wymagania zaprezentowano poniżej.

2.2 Założenia szczegółowe

2.2.1 Konstrukcja robota

1. Stworzenie robota holonomicznego: stopień mobilności – (3,0) co narzuca wykorzystanie wielokierunkowych kół szwedzkich, które umożliwią poruszanie się robotem „w bok” oraz na „skos”. Napęd wraz z kołami przytwierdzony na stałe do konstrukcji robota - brak osi skrętnych.
2. Uniwersalność platformy – konstrukcja robota powinna umożliwiać wykorzystywanie różnych komponentów jak np. napędy różnego rodzaju.
3. Duża przestrzeń na elektronikę – konstrukcja robota zaprojektowana tak, aby pomieściła oprócz sterownika pokładowego – baterię, czujniki oraz w przyszłości mikrokomputer jak np. Raspberry Pi, Nvidia Jetson lub pokrewne.

4. Duża ilość mocowań na nieprzewidziane w trakcie projektowania elementy
5. Prostota eksploatacji i obsługi serwisowej

2.2.2 Elektronika pokładowa robota

1. Sterownik umożliwiający zdalną komunikacje (WiFi lub Bluetooth)
2. Prostota programowania mikrokontrolera oraz powszechny, znany układ.
3. Prostota konstrukcji – układ sterowania powinien cechować się nieskomplikowaną budową w celu prostej diagnozy i naprawy sprzętu
4. Zasilanie bateryjne/akumulatorowe – zasilanie z dedykowanego zasilacza poprzez kabel, byłoby w sprzeczności z wysoką mobilnością robota.
5. Wystarczający zapas mocy uwzględniający np. wykorzystanie dodatkowo mikrokomputera typu Raspberry Pi lub pokrewnych urządzeń.
6. Robot powinien zostać wyposażony w wyłącznik bezpieczeństwa.
7. Niski koszt zakupu i montażu elementów.
8. Wyposażenie w co najmniej jeden z czujników: przyspieszenia, magnetometru lub czujnika ultradźwiękowego, umożliwiającego w przyszłości szczegółową analizę ruchu lub zorientowanie robota w przestrzeni.
9. Wyposażenie robota w silniki prądu stałego zawierające enkodery inkrementalne.
10. Cały układ sterowania zawarty na jednej wspólnej płytce PCB (ang. *Printed Circuit Board*) .

2.2.3 Program sterownika

1. Wykorzystanie prostego i popularnego środowiska programistycznego dla wybranego mikrokontrolera.
2. Możliwość komunikacji zdalnej – obsługa komunikatów – poleceń i odpowiednie reagowanie robota na zdarzenia.
3. Pobieranie danych z wszelkich czujników i wysyłanie ich do aplikacji sterującej
4. Obróbka danych z czujników głównie w aplikacji sterującej (nadrzędnej).

5. Oprogramowanie sterownika nieskomplikowane, realizujące zdalne raportowanie i stosowanie poprawek nadanych przez aplikację nadrzędną.
6. Implementacja algorytmu PID do sterowania prędkością obrotową napędów, reakcja na przeciążenia silników.
7. Bezzwłoczna reakcja na sytuację kryzysową (załączony przycisk bezpieczeństwa)
8. Aplikacja nadrzędna otrzymująca wszelkie parametry i status robota, umożliwiająca sterowanie w wielu wybranych przez użytkownika kierunkach.

Rozdział 3

Platforma mobilna

Na rynku dostępnych jest wiele rozwiązań platformowych dla robotów mobilnych. Poczynając od prostych, małych płyt laminowanych przystosowanych do dwunapędowych robotów, kończąc na niemal gotowych pojazdach gąsienicowych lub kołowych.

Wadą takich rozwiązań, jest ich wysokie ukierunkowanie na danego producenta elektroniki lub napędów, przez co zostaje narzucona z góry technologia i komponenty jakie muszą być zastosowane, by zachować kompatybilność. Z kolei tanie konstrukcje charakteryzują się niską wytrzymałością, niewielkim rozmiarem (w kontekście miejsca dostępnego pod zabudowę np. elektroniki), jakości wykonania, a nawet estetyki.

Popularność robotów mobilnych wykorzystujących koła szwedzkie, jest wciąż niewielka. Stąd zdecydowano się o skonstruowaniu własnej platformy mobilnej od podstaw, spełniającej wszystkie wyszczególnione wcześniej założenia.

3.1 Projekt

Do wykonania projektu platformy mobilnej posłużyło oprogramowanie do projektowania wspomaganego komputerowo (ang. *Computer-Aided Design*) o nazwie Autodesk Inventor Professional 2020, dostępne w ramach płatnej licencji lub jak w tym przypadku – licencji darmowej, dedykowanej studentom uczelni wyższych.

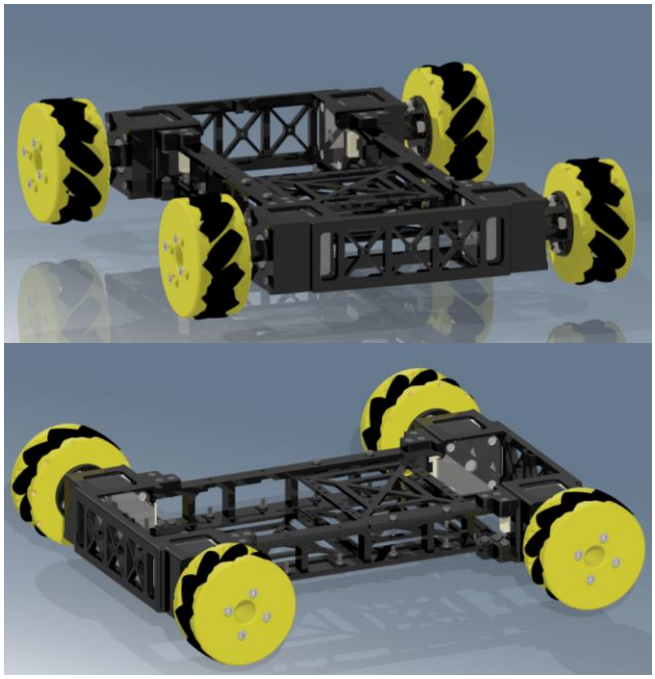
W projekcie, do wykonania zaprojektowanej konstrukcji, posłużyła drukarka 3D – Ender 3 Pro. Wykorzystanie wspomnianego wyżej oprogramowania CAD i drukarki 3D, umożliwia bardzo wydajne tworzenie elementów potrzebnych do wykonania platformy mobilnej robota, ograniczone co do zasady wielkością stołu roboczego drukarki oraz kreatywnością konstruktora. Technologia druku to standardowa technika FDM (ang. *Fused Deposition Modelling*) nakładająca rozgrzany, płynny plastik warstwa po warstwie,

docelowo tworząc plastikowy element o zadanym przez operatora wypełnieniu i zgodnym z projektem CAD – kształcie. Umożliwia to względnie szybką możliwość wydruku modeli np. celem sprawdzenia dopasowania lub ingerencję w wytrzymałość materiału (w myśl zasady im więcej wypełnienia, tym element bardziej wytrzymały). Do konfiguracji wydruku poszczególnych komponentów i ustawień drukarki 3D, posłużyło oprogramowanie Cura, które jest darmowej licencji. Jest to jeden z popularnych programów typu „Slicer", umożliwiający parametryzacje procesu wydruku i przetworzenie tych ustawień, na zrozumiały dla drukarki kod wykonawczy – tj. Gcode.

3.1.1 Proces projektowania

Jako że postawiono na dużą uniwersalność platformy i jej pojemność, zdecydowano oprzeć konstrukcję na systemie modułowym. Założono, że co do zasady główna konstrukcja robota, będzie się składać z korpusu (dalej zwanym „kadłubkiem") - do którego będą mocowane napędy - oraz głównych podłużnic, do których oprócz zamocowanych korpusów, zostaną przytwierdzone odpowiednie mocowania pod elektronikę, czujniki lub baterię. Ponadto, umieszczono wiele otworów mocujących (rozmiar M3) oraz dostosowano korpusy do zastosowania różnych rodzajów napędów, nawet uwzględniając silniki Nema17.

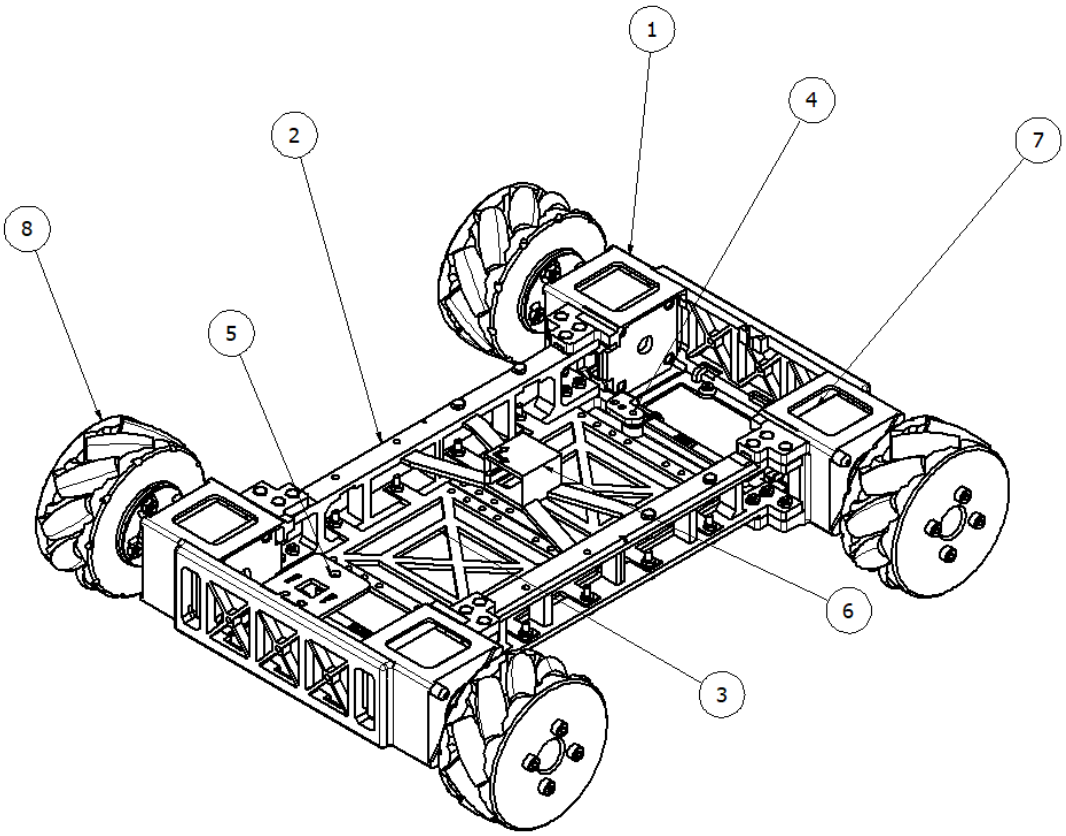
Trzymając się opisanych założeń proces projektowania zakończył się rezultatem widocznym na Rys. 3.1.



Rys. 3.1 Wizualizacja projektu robota mobilnego

3.1.2 Konstrukcja robota

W oparciu o założenia wypracowano poniższy model platformy jezdnej robota.

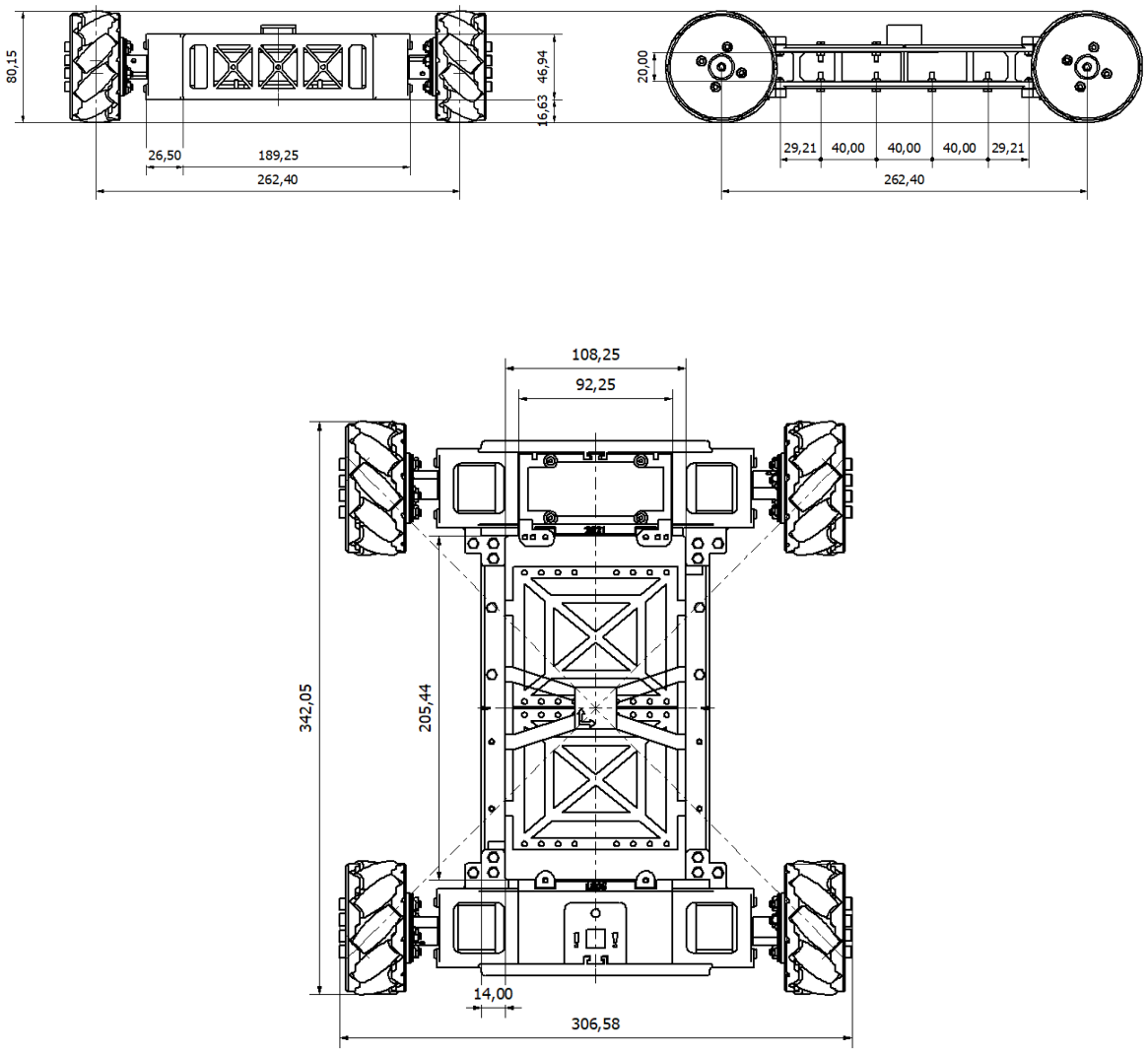


Rys. 3.2 Prezentacja poszczególnych elementów platformy mobilnej

Elementy pojazdu (Rys. 3.2):

1- *Korpus* (zwany kadłubkiem), 2- *podłużnica*, 3- *platformy mocujące elektronikę*, 4- *mocowanie przetwornicy napięć*, 5- *mocowanie diody powiadomień i przycisku bezpieczeństwa*, 6 – *obudowa czujnikó*w (akcelerometru, magnetometru), 7 – *napęd* (na rysunku Nema17, w rzeczywistości JGA25-371), 8 – *koło Mecanum*

Podczas projektowania tej konstrukcji, bardzo ważnym było zachowanie geometrii rozmieszczenia poszczególnych kół, na bazie kwadratu – tj. dwie osie (przednia i tylna) oraz odległość między punktami styku poszczególnych kół na osi powinny mieć ten sam wymiar. Co równie istotne, miejsce przecięcia przekątnych powstałego kwadratu, powinno przypadać na geometryczny środek pojazdu. Rysunki techniczne przestawiające najważniejsze wymiary i rzutowania poszczególnych osi znajdują się na Rys. 3.3



Rys. 3.3 Wymiary całkowite pojazdu.

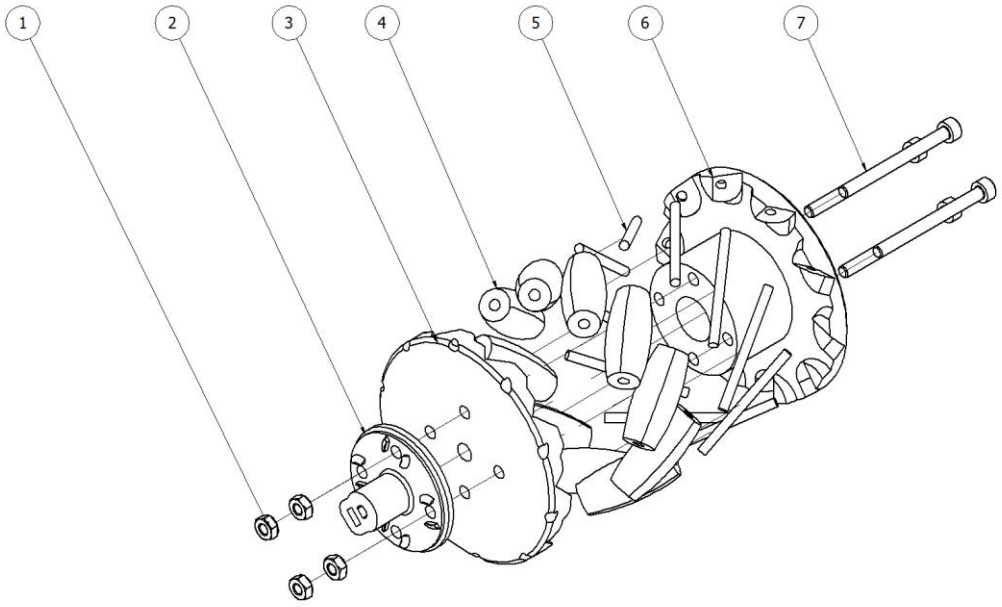
Tabela 3.1 *Podsumowanie najważniejszych parametrów platformy jezdnej (Rys. 3.3)*

Wymiary całkowite pojazdu:	Długość: 342,05 mm	Szerokość: 306,05 mm
Rozsunięcie osi:	Wzdłuż: 262,40 mm	Wszerz: 262,40 mm
Wymiary głównej przestrzeni zabudowy:	Długość: 205,44 m	Szerokość: 108,25 mm
Główna powierzchnia zabudowy:	222,38 cm²	
Minimalny prześwit:	16,63 mm	
Maksymalna wysokość pojazdu:	80,15 mm	
Średnica koła:	80 mm	
Masa całkowita pojazdu:	~1,3 kg	

3.1.3 Koła szwedzkie – Mecanum

Kluczowym elementem tego projektu są koła omnikierunkowe. Dzięki nim, robot jest w stanie poruszać się w niemal każdym kierunku, bez względu na położenie koła (tj. np. kąt skrętu). Takie rozwiązanie, składa się z korpusu koła (zwanego piastą), w którym zamontowano różną (w zależności od projektu) ilość rolek obracających się wokół własnej osi. Na rynku istnieją dwa rodzaje kół omnikierunkowych:

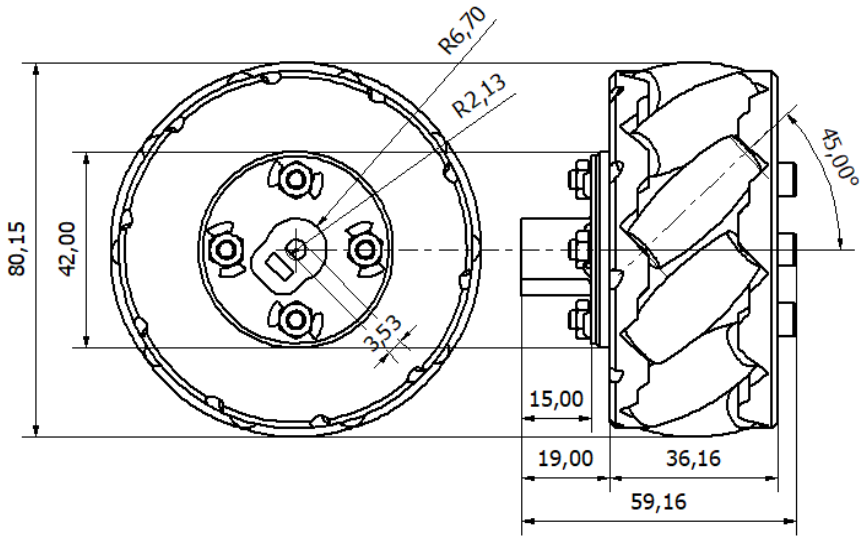
- klasyczne – koła omnikierunkowe
- koła Mecanum – zastosowane w tym projekcie



Rys. 3.4 Prezentacja poszczególnych elementów koła Mecanum

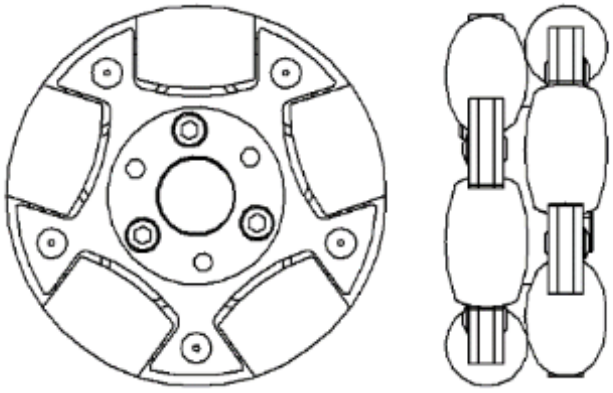
Elementy koła Mecanum (Rys. 3.4):

1 – *Nakrętka sześciokątna M4* (4szt.), 2 – *Adapter wału* – przejściówka umożliwiająca wykorzystanie różnych silników, bez konieczności wymiany całych kół, 3 – *wewnętrzny korpus (piasta) koła*, 4 – *rolka z rurką termokurczliwą* zwiększającą przyczepność kół (10 szt.), 5 – *oś rolki* (pręt stalowy nierdzewny, średnica: 3mm, 10szt), 6 – *zewnętrzna piasta* (korpus) koła, 7 – *śruby imbusowe M4* (4szt.).



Rys. 3.5 Rysunek techniczny zastosowanych kół Mecanum

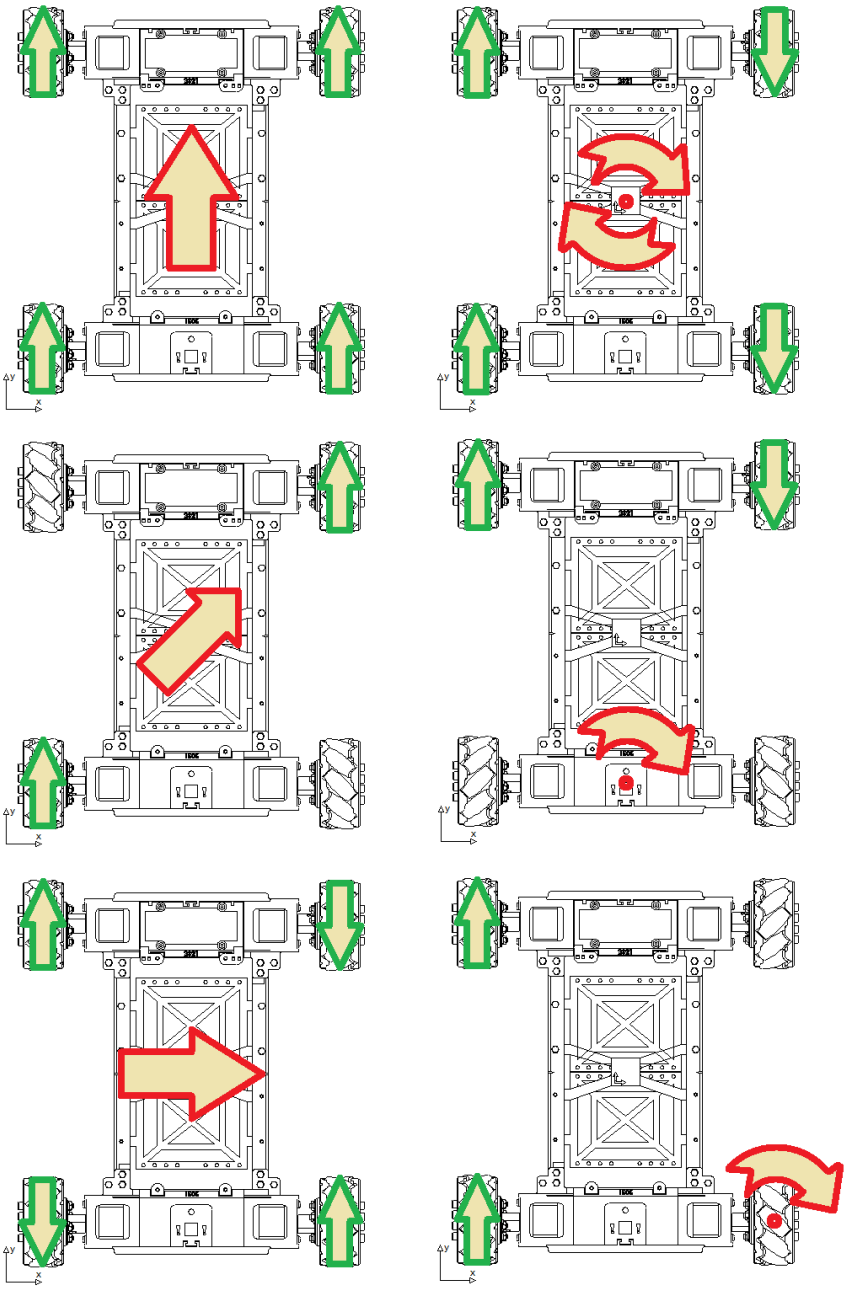
Projekt kół Mecanum zaczerpnięto z projektu podobnego pojazdu, udostępnionego w Internecie [3]. Został on odpowiednio zmodyfikowany i dostosowany do projektowanego robota np. w postaci innego adaptera, wydrukowanego z materiału ABS (akrylonitryl-butadien-styren) charakteryzującego się bardzo wysoką twardością i wytrzymałością. Wydruki za pomocą plastiku PLA (poliaktyd) po czasie użytkowania zaczęły się odkształcać, generując luz między wałem napędowym silnika, a adapterem koła.



Rys. 3.6 Rysunek poglądowy kół omnikierunkowych klasycznych [4]

Zasadniczą różnicą pomiędzy dwoma rodzajami kół, jest kąt pod jakim zostały umieszczone rolki. W przypadku klasycznych kół, rolka jest skierowana prostopadle (kąt 90º) do osi obrotu koła. Z kolei koła Mecanum posiadają rolki skierowane pod kątem 45º do osi obrotu koła.

Rys. 3.7 przedstawia metodykę sterowania kołami Mecanum i rezultat w postaci ruchu platformy mobilnej. Zielone strzałki odpowiadają rotacji poszczególnych z kół. Czerwona strzałka informuje o wynikowym ruchu wykonanym przez platformę mobilną. W przypadku rotacji, czerwony punkt informuje o powstałej osi obrotu.

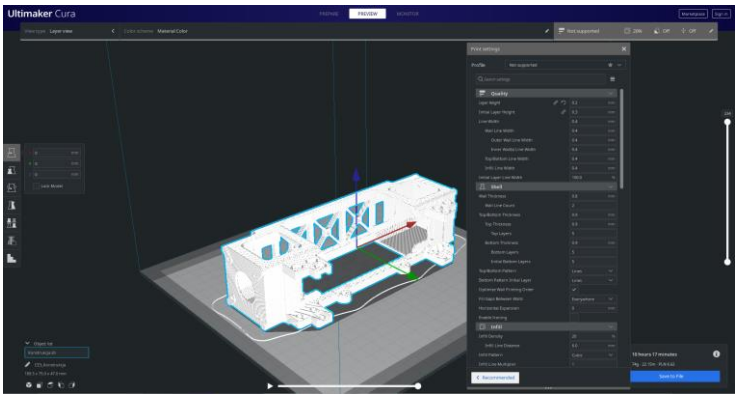


Rys. 3.7 Sekwencja ruchów dla robota z kołami Mecanum

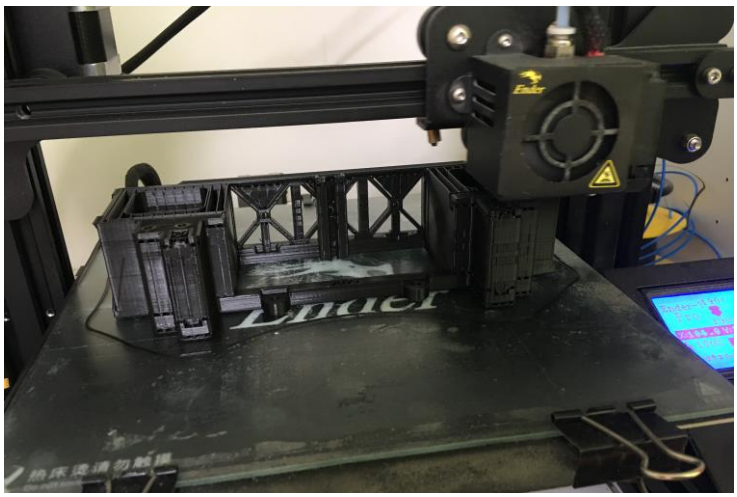
3.1.4 Proces wykonawczy

Po zakończonym procesie renderingu, projekt został wyeksportowany do oprogramowania umożliwiającego przygotowanie do druku wygenerowanej bryły 3D. Proces ten polegał na doborze odpowiednich ustawień drukarki i parametrów wydruku, mając na uwadze względnie krótki czas wydruku i akceptowalną wytrzymałość i jakość wydruku. Kwestie dotyczące jakości i wytrzymałości drukowanych elementów, oparto na kilkuletnim doświadczeniu w druku 3D autora pracy inżynierskiej.

Największym wyzwaniem, był jednak wydruk głównego elementu korpusu robota. Wymagał on wielu podpór i optymalizacji w kategorii wypełnienia elementu (mającego wpływ na czas druku), szybkości ruchu głowicy drukarki, chłodzenia wydruku i wielu innych.

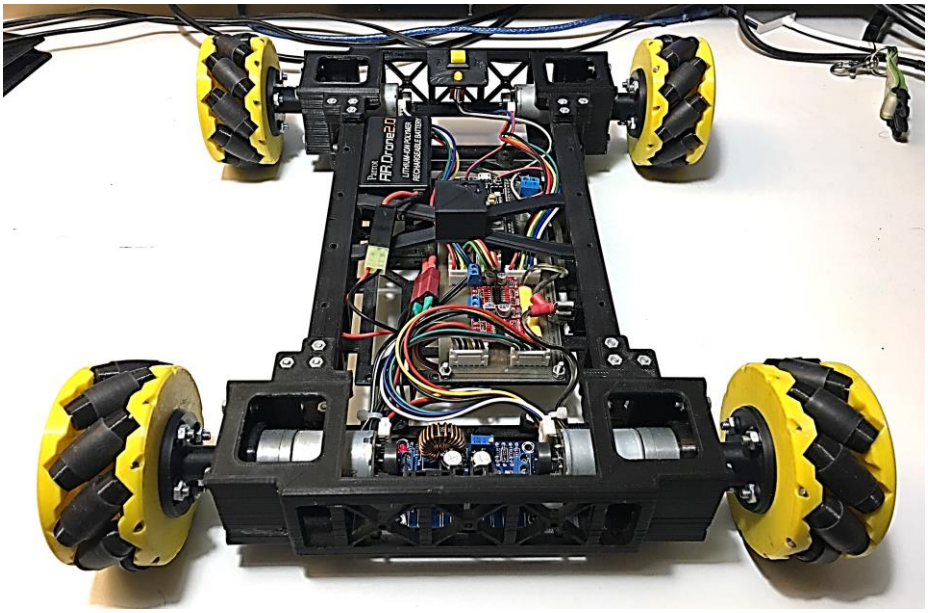


Rys. 3.8 Widok z programu Cura, podczas przygotowania elementu do druku

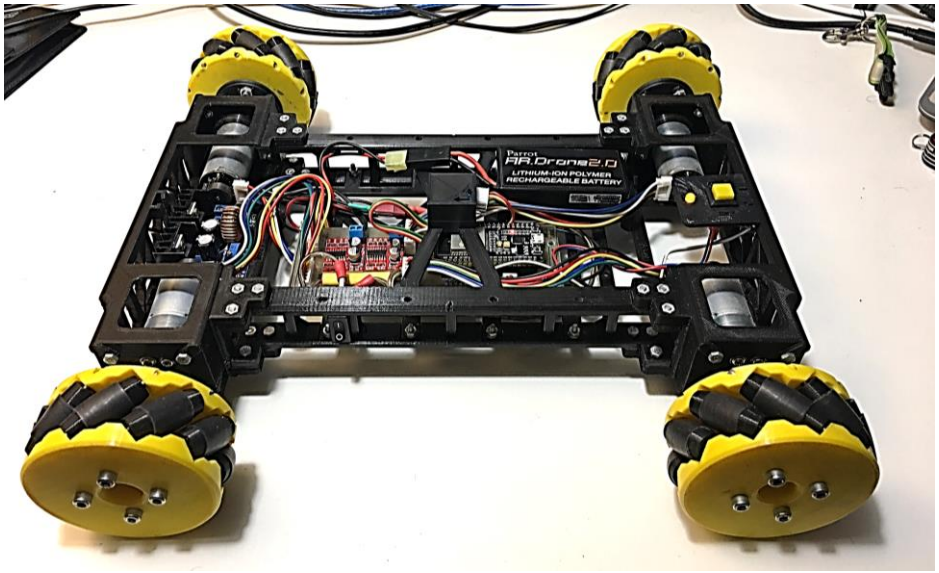


Rys. 3.9 Proces wydruku kadłubka (10 godzina druku)

Łączny czas druku wszystkich elementów robota, szacuje się na ok. 60h. Po zamontowaniu wszystkich potrzebnych elementów, platforma jezdna została zaprezentowana na Rys. 3.10 oraz Rys. 3.11



Rys. 3.10 Wykonana platforma jezdna (widok z przodu)



Rys. 3.11 Wykonana platforma jezdna (widok z boku)

Rozdział 4

Elektronika

Projekt mobilnego robota kołowego, obejmował również dobór elementów elektronicznych, umożliwiających sterowanie i odpowiednie zaprogramowanie urządzenia. Na rynku dostępne jest wiele gotowych sterowników pokładowych, często ukierunkowanych na daną markę lub producenta sprzętu. Istotną wadą takiego rozwiązania jest bardzo wysoki koszt zakupu i ograniczenie reszty elementów pokładowych do danego sterownika.

Postanowiono znaleźć złoty środek pomiędzy kosztami budowy/zakupu sterownika, a jego uniwersalnością i możliwościami. W tym celu wydzielono kilka komponentów przeznaczonych do analizy, kupna i zbudowania pełnowartościowego sterownika.

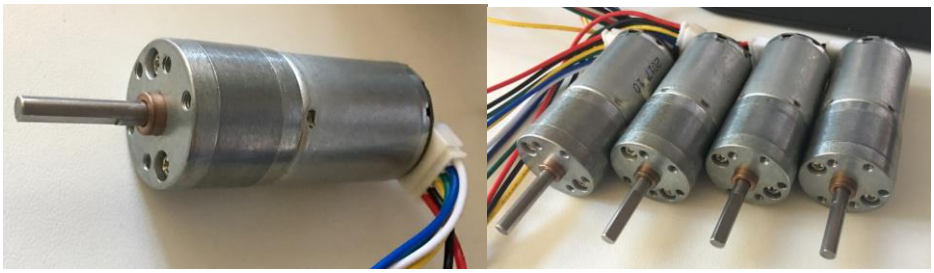
Producenci robotów mobilnych konstruują swoje urządzenia z wykorzystaniem wielu komponentów. Mogą istnieć różnice w oprogramowaniu lub wyglądzie, lecz sam podzbiór elementów wykorzystywanych do poprawnego funkcjonowania jest zazwyczaj następujący:

- napęd robota (np. silniki z przekładnią, silniki bezszczotkowe – stosowane w dronach),
- główna jednostka decyzyjna – mikroprocesor (np.procesor ARM, Atmega, itp.),
- sterownik napędów (Często opracowywane na potrzeby producenta),
- różnorodne czujniki (Przyspieszeń, mechaniczne, prądowe, lokalizacyjne),
- zasilanie robota (Bateria, układ zasilający),
- urządzenia/interfejsy komunikacyjne (względem medium transmisyjnego np. WiFi, Bluetooth, połączenie przewodowe).

Pełny schemat elektryczny przedstawiono w *Dodatku A* w sekcji *Schemat elektryczny*.

4.1 Napęd

W pojeździe zastosowano silniki szczotkowe prądu stałego, wyposażone w przekładnie zębatą oraz enkoder inkrementalny (działający na zjawisku Halla, wraz z detekcją kierunku obrotu). Konstrukcyjnie wykorzystany napęd bazuje na popularnym zestawie JGA 25-371.



Rys. 4.1 Zastosowany silnik prądu stałego wraz z przekładnią i enkoderem

Tabela 4.1. *Dane techniczne zastosowanego napędu (JGA25-371) (wg. sprzedającego)*

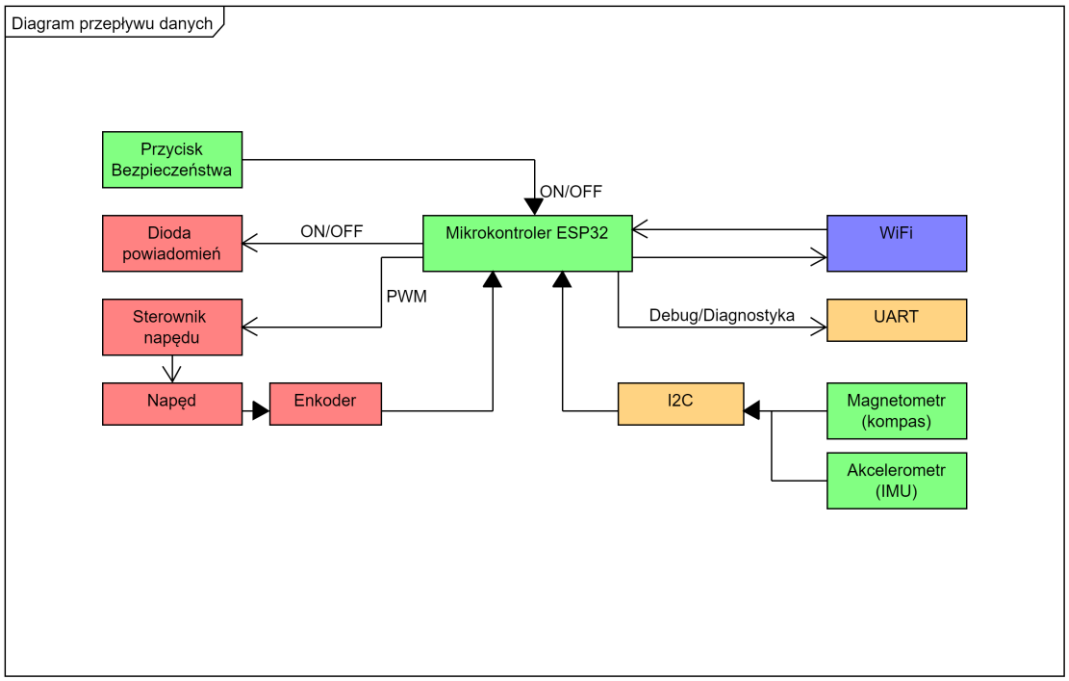
Deklarowane napięcie:	5V	6V
Prędkość obrotowa (bez obciążenia)	158 RPM	180 RPM
Pobór prądu (bez obciążenia)	120mA	130mA
Pobór prądu (skrajne obciążenie)	~1.5A	
Rodzaj przekładni:	Bezstopniowa, zębata	
Przełożenie:	1:35.5	
Waga całkowita:	~0.093 kg	
Wymiary:	Średnica:	∅ 25 mm
	Długość (bez wału):	52.8 mm
	Średnica wału wyjściowego:	∅ 4 mm
	Długość wału wyjściowego:	20 mm
Enkoder:	Obrotowy, inkrementalny, rozdzielczość 16-bit.	

4.2 Projekt sterownika pokładowego

Proces projektowania sterownika pokładowego przeprowadzono dzięki darmowemu oprogramowaniu do projektowania schematów elektrycznych i płytek PCB – KiCad.

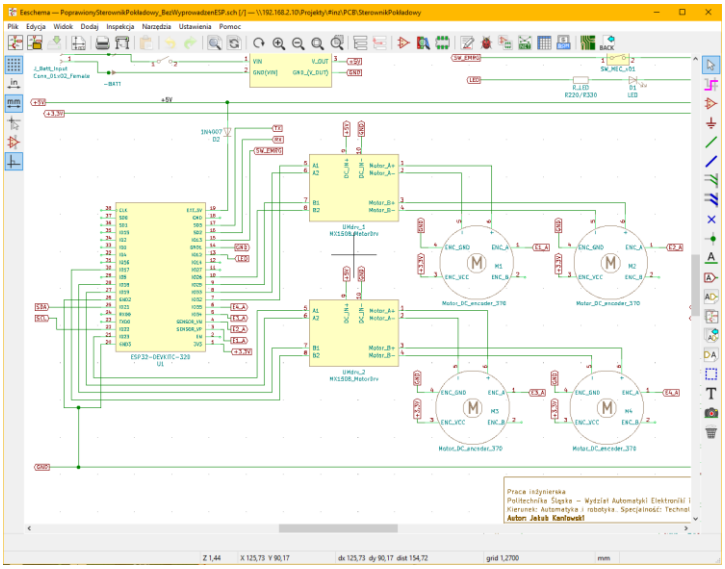
Pierwotnie zakładano zrealizowanie całego układu sterującego, na płytce uniwersalnej, charakteryzującej się wieloma otworami montażowymi i brakiem bezpośrednich połączeń między poszczególnymi punktami. Zrezygnowano jednak z tego rozwiązania, z uwagi na wysoce zawodne połączenia między elementami. Należy pamiętać, że koła szwedzkie podczas obrotu powodują wibracje całej platformy, co narażało tak wykonaną elektronikę na uszkodzenia – w postaci urwanych połączeń kablowych, braku styku i innych zdarzeń losowych, które powodowały niestabilną pracę całej elektroniki robota.

Projekt elektroniki rozpoczęto od rozplanowania hierarchii, według której będą tworzone połączenia.



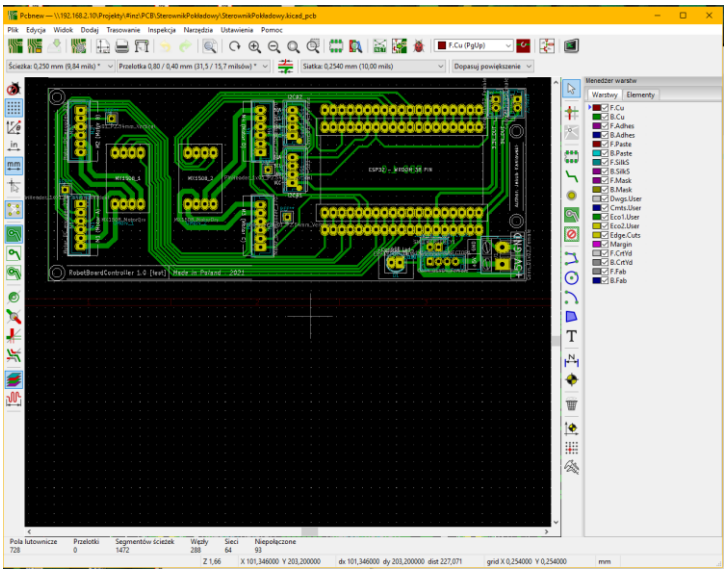
Rys. 4.2 Diagram hierarchiczny elementów systemu sterującego

Następnie, w narzędziu Eschema, wykonano odpowiednie połączenia modułów pomiędzy sobą – zgodnie z rozplanowaną wcześniej hierarchią i sposobem podłączenia.



Rys. 4.3 Widok projektu z programu KiCad Eschema

Ostatnim etapem projektowania elektroniki było przeniesienie schematu do programu PCB Layout, co umożliwiło bezbłędne rozplanowanie tras ścieżek na laminacie PCB. Postawiono na modułowość – w razie potrzeby moduły są podłączane za pomocą listwy Gold-Pin lub złącz JST.



Rys. 4.4 Widok projektu z programu KiCad PCB Layout

Po zakończeniu projektu, przystąpiono do wykonania płytki drukowanej. Przeniesienie matrycy ścieżek wykonano za pomocą metody termotransferu – tj. ścieżki i pady zostały wydrukowane na papierze kredowym, a następnie po odpowiednim przygotowaniu laminatu

PCB (tj. przeszlifowaniu i odtłuszczeniu powierzchni miedzianej) za pomocą rozgrzanego żelazka – wydruk z papieru został przeniesiony na warstwę miedzi. Dokonano ostatniej rewizji i poprawek w postaci wyczyszczenia gęstych połączeń, a następnie przygotowaną płytkę z nadrukiem położono do kąpieli w kwasie.

4.3 Mikrokontroler – Espressif ESP32

W systemie sterowania postawiono na wykorzystanie popularnego mikrokontrolera w formie modułu deweloperskiego NodeMCU-32S firmy Ai-Thinker. Parametry techniczne tego mikrokontrolera i jego modułowość zdecydowały o wyborze tego układu, jako głównej jednostki wykorzystywanej do sterowania całym robotem mobilnym.

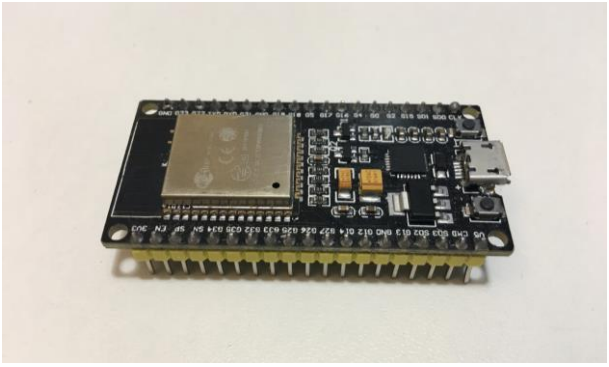
Płytka deweloperska składa się z modułu mikrokontrolera firmy Espressif [5], model ESP32-WROOM-32, który umożliwia zdalną komunikacje poprzez WiFi lub Bluetooth, co pokrywa się z wymaganiami projektowymi. Podczas wyboru mikrokontrolera były brane także pod uwagę układy firmy Atmel, które charakteryzowały się równie prostą implementacją programową jak i również dobrymi parametrami, jednakże główną wadą było to, iż takie mikrokontrolery wymagały doposażenia w odrębny moduł komunikacyjny, który de facto sprowadzał się np. do użycia modułów ESP firmy Espressif. Należy również wspomnieć o dużym potencjale wybranego modułu deweloperskiego, co w przypadku przyszłościowego rozwijania tego projektu jest bardzo ważnym atutem. Zastosowano tutaj mikroprocesor Xtensa® 32-bit LX, który jest układem dwurdzeniowym, a samo środowisko programowania np. ESP-IDF [6] umożliwia na implementacje systemu czasu rzeczywistego FreeRTOS, co złoży się na determinizm czasowy wykonywanych operacji oraz wielozadaniowość systemu, która w robotyce mobilnej jest bardzo pożądana.

Mikrokontroler wyposażono również w wiele wyjść cyfrowych oraz analogowych co przydaje się w przypadku sterowania robotem i zupełnie wystarcza w tym projekcie. Co bardzo istotne, każde wyjście cyfrowe tego mikrokontrolera (w odróżnieniu od innych modułów) może być wykorzystane jako wyjście sterujące sygnałem PWM (ang. *pulse-width modulation*) [6].

Tabela 4.2 *Dane techniczne mikrokontrolera* [5] [6]

NodeMCU-32S		
Napięcie operacyjne:	min: 3.3V	max: 15V
Pobór prądu:	min: 5μA (tryb uśpienia)	max: 1,5A
Środowisko programowania:	Arduino-IDE, ESP-IDF	
ESP32-WROOM-32		
Zastosowany mikroprocesor:	Xtensa® dual-core 32-bit LX6	
Architektura:	32 bitowa (wsparcie sprzętowe dla FPU[1])	
Częstotliwość pracy:	2 rdzenie: 240 MHz	
Pamięć ROM:	448 KB	
Pamięć SRAM:	520 KB	
Pamięć Flash:	4 MB	
Peryferia		
Komunikacja radiowa:	WiFi (802.11b/g/n 2.4GHz), Bluetooth (BT Low-Energy)	
Interfejsy komunikacyjne:	4 x SPI, 2 x I2S, 2 x I2C, 3 x UART	
Wyprowadzenia:	GPIO: 34 (z czego ADC: 18, DAC: 2, PWM)	

Płytka deweloperska NodeMCU-32s oferuje również wyprowadzenia zasilające +5V oraz +3.3V DC, które zostały wykorzystane do zasilania czujników pokładowych.



Rys. 4.5 Wykorzystany mikrokontroler

[1] FPU- Floating Point Unit – wsparcie sprzętowe dla obliczeń zmiennoprzecinkowych (koprocesor). Procesory nie posiadające koprocesora FPU mają duży problem z wydajnością podczas obliczeń zmiennoprzecinkowych – dlatego wymagane jest wtedy przejście z liczb zmiennoprzecinkowych, na liczby całkowite. Dla tego mikroprocesora można dokonywać obliczeń bezpośrednio na liczbach zmiennoprzecinkowych zachowując przy tym wydajność aplikacji i prostotę programowania.

4.4 Sterownik silników prądu stałego

W trakcie projektowania i wyboru potrzebnych komponentów rozważano kilka możliwych do kupienia na rynku sterowników silnika DC w układzie mostka H – jest to bardzo proste i popularne rozwiązanie kwestii sterowania silnikami prądu stałego. Pierwotnie, rozważano zastosowanie popularnego układu L293D, gdyż według noty katalogowej, było względnie możliwe wykorzystanie tego komponentu. Za wybór tego elementu, przemawiała prosta implementacja, doświadczenie autora i możliwość umieszczenia go, na jednym wspólnym laminacie PCB. Testy obciążeniowe wykazały jednak, że w momencie krytycznego obciążenia – tj. zatrzymania silnika, układ nie wytrzymuje pobieranego przez napęd prądu i wydziela nadmiernie ciepło (jak również ulega uszkodzeniu). Według noty katalogowej układ L293D jest w stanie wytrzymać prąd o natężeniu do 1,2A w piku. Siłowe zatrzymanie napędu, generuje średnio natężenie od 1.5A do 1.8A, zatem istniała realna możliwość uszkodzenia układu.

Tabela 4.3. *Nota katalogowa sterownika L293D* [7]

Deklarowane napięcie wejściowe:	zalecane min: 4.5V	max. zalecane: 7V, max. 36V
Ilość kanałów:	2	
Napięcie wyjścia:	min. 2.3V	max. zalecane: 7V, max. 36V
Prąd wyjściowy:	od. -600mA	do 600mA na kanał
Prąd szczytowy:	do 1.2A na kanał	
Rodzaj układu:	Mostek H	
Sposób sterowania:	PWM oraz określenie kierunku (EN)	

Kolejnym rozważanym układem, był układ L298N. Układ ten, w kontekście gotowego modułu elektronicznego jest wyposażony w chłodzenie pasywne (radiator), a cała elektronika mieści się na odrębnym laminacie PCB, o wymiarach ok. 4,5 cm x 4,5 cm. Zaletą tego układu, jest stosunkowo wysoka odporność na przeciążenia i duży zapas mocy.

Istnieje możliwość kupna samego układu scalonego L298N, jednak jest to ekonomicznie nieuzasadnione, a dostępność na rynku tychże układów jest stosunkowo niska. Wadą wykorzystania tego modułu w projekcie jest:

- ilość zajmowanego miejsca (wymagane dwa moduły, dla czterech napędów) co daje ok. 40cm^2, co w gruncie rzeczy stanowi prawie 10% dostępnej powierzchni zabudowy robota,

- wymagana większa ilość wyprowadzeń mikrokontrolera do sterowania modułem,
- zastosowanie samego układu scalonego, wiąże się z nieadekwatnie dużymi kosztami,
- wysoka cena modułu.

Tabela 4.4. *Nota katalogowa układu L298N* [8]

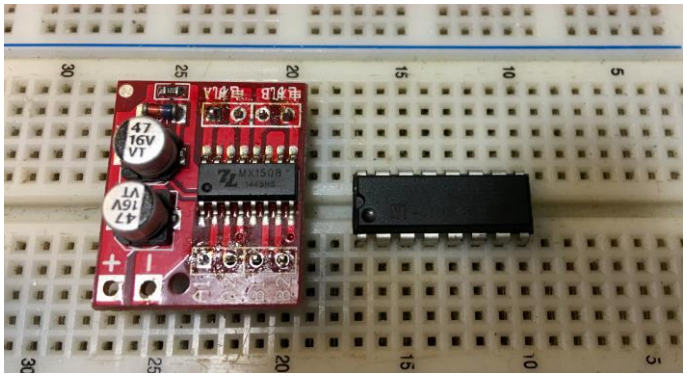
Deklarowane napięcie wejściowe:	Zalecane min: 4.8V	max. 46V (część logiczna: 7V)
Ilość kanałów:	2	
Napięcie wyjścia:	min. 2.3V	do 46V
Prąd wyjściowy:	do 2A na kanał	
Prąd szczytowy:	do 2.5A na kanał (niekiedy deklarowane 3A)	
Rodzaj układu:	Mostek H	
Sposób sterowania:	PWM oraz określenie kierunku (EN)	

Ostatni z rozważanych układów został finalnie wykorzystany podczas budowy robota mobilnego. Jest to mało popularny układ MX1508, często mylony na aukcjach bądź w sklepach z modułem L298N „w wersji SMD” [9]. Zastosowany układ cechuje się bardzo niską ceną w stosunku do swoich możliwości, kompaktowymi wymiarami (Rys. 4.6) oraz możliwością bezpośredniego przylutowania do płytki PCB. Daje to bardzo dużą przewagę względem L298N, który jak już wykazano, jest stosunkowo sporych rozmiarów i ponadto nie jest możliwe jego bezpośrednie wlutowanie w główną płytkę PCB. Co również istotne, sterownik MX1508 ma bardzo ciekawie rozwiązaną kwestie sterowania. Otóż, o ile w poprzednich układach były wymagane sygnały logiczne określające kierunek obrotu oraz odrębny sygnał PWM nadający prędkość obrotową, tak w przypadku sterownika MX1508, do sterowania wykorzystywane są tylko i wyłącznie dwa sygnały PWM. Daje to pewnego rodzaju oszczędność wyprowadzeń, które w przypadku mikrokontrolera ESP32 są znacznie ograniczone.

Tabela 4.5. *Nota katalogowa sterownika MX1508* [9]

Deklarowane napięcie wejściowe:	min. 2V	max. 10V
Napięcie wyjścia:	min. 1.8V	max. 7V
Pobór prądu:	min. 0.1μA	max. 1.5A na kanał
Prąd szczytowy:	do 2.5A na kanał	
Rodzaj układu:	Mostek H	
Sposób sterowania:	PWM	
Wykorzystane napięcie wejściowe:	5V	
Moc maksymalna (dla 5V):	do 7.5W na kanał	
Sumaryczna moc (dla 4 napędów):	30W	
Pobór prądu (dla 4 napędów):	do 6A	

Jak można zauważyć, sterownik ten w trybie normalnej pracy, zapewnia wystarczający prąd, do sterowania napędami. Problem może natomiast nastąpić w momencie celowego zatrzymania napędu. Wielokrotnie przeprowadzone testy empiryczne dowiodły, iż układ jest wystarczająco wytrzymały na tego typu przypadki. Ponadto należy pamiętać, że otrzymywane sygnały z enkoderów napędu można przeanalizować pod kątem celowego zatrzymania silnika. W momencie wykrycia takiego zjawiska, oprogramowanie mikrokontrolera powinno podjąć decyzje o zaprzestaniu dalszego sterowania danym napędem i zgłoszeniu odpowiedniego błędu operatorowi. W odróżnieniu do układu L293D, podczas normalnej pracy układ nie nagrzewa się, co zapewnia mu bezpieczne warunki pracy. Z noty katalogowej wynika, iż układ ten – podobnie jak L298N ma zabezpieczenie przed przegrzaniem.



Rys. 4.6 Porównanie rozmiarów sterowników MX1508 oraz L293D

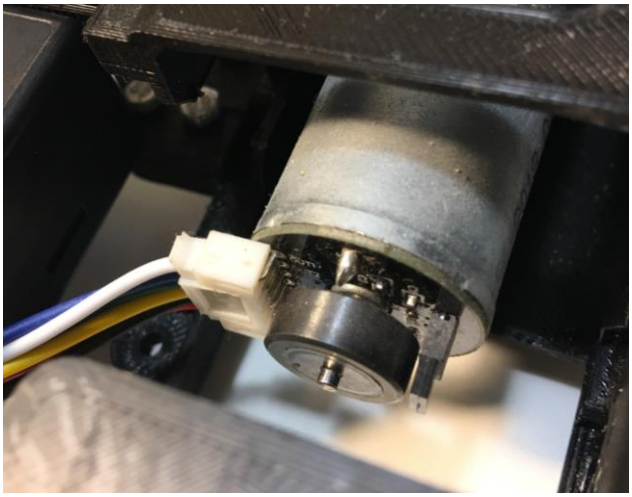
4.5 Czujniki

Każdy robot mobilny, musi być wyposażony w czujniki, dające informacje o otaczającym go świecie lub stanie samego urządzenia i jego podzespołów. W konstruowanym systemie sterowania postawiono na czujniki położenia napędów oraz czujników przydatnych w nawigacji i orientowaniu robota – akcelerometru, żyroskopu i magnetometru.

4.5.1 Enkodery

W projekcie postawiono na jedną z najbardziej podstawowych rodzajów nawigacji mobilnej – odometrię, która wykorzystuje w głównej mierze informacje o przebytej drodze przez pojazd. W robotyce lub automatyce do tego celu stosuje się różnego typu enkodery. Można wyróżnić główne dwa typy enkoderów inkrementalnych: magnetyczne oraz optyczne. Enkodery optyczne charakteryzują się dużą dokładnością i prędkością pomiaru oraz brakiem zakłóceń spowodowanych szumem elektromagnetycznym – występującym np. w pobliżu linii energetycznych wysokiego napięcia. Enkodery magnetyczne są tańsze i odporne na zabrudzenia lub wibracje, które przy zastosowaniu kół Mecanum będą występowały. Wymienione powyżej zalety, zadecydowały o wyborze enkoderów magnetycznych, które w pełni sprawdzają się w powierzonej roli.

Przedstawiony na Rys. 4.7 zestaw obrotowego enkodera inkrementalnego, składa się z koła magnetycznego i czujnika Halla (MT-1450 EN), umiejscowionego po prawej stronie od wału silnika. Wraz z obrotem wału silnika, obraca się koło magnetyczne - jego obrót powoduje zmiany strumienia magnetycznego, które są wykrywane przez czujnik Halla. W praktyce powstaje więc sygnał napięciowy sinusoidalny, który jest interpolowany bezpośrednio w zastosowanym czujniku do sygnału prostokątnego. Jest to możliwe dzięki wbudowanemu komparatorowi napięcia. Opisywany enkoder, umożliwia detekcje kierunku obrotów (wyjścia A i B enkodera), poprzez sygnał przesunięty o 90º w fazie [10]. Ta funkcjonalność nie jest jednak wykorzystywana w tym projekcie bezpośrednio.



Rys. 4.7 Enkoder inkrementalny napędu robota

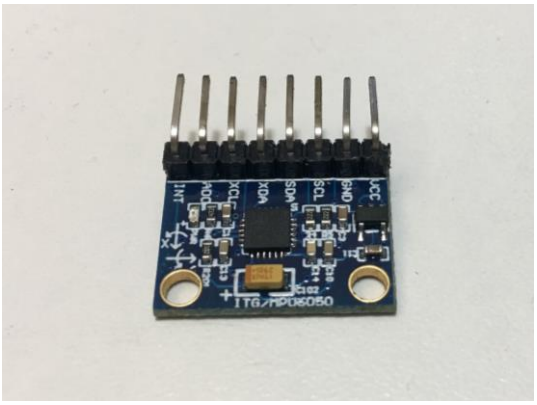
Wykorzystane enkodery magnetyczne zliczają ilość obrotów wału silnika DC. Następnie dane te, zostają przeliczone na wartość obrotów na minutę (RPM) uwzględniając przełożenie napędu (Tabela 4.1). Mając taki zestaw informacji, można wyznaczyć prędkość kątową, liniową lub drogę jaką przebył każdy z napędów pamiętając o uwzględnieniu w obliczeniach średnicy zastosowanych kół mecanum (Rys. 3.5). Więcej szczegółów zostanie poruszone w rozdziale 5.3.2.

4.5.2 Czujnik IMU

W robocie mobilnym, zastosowano również czujnik nawigacji inercyjnej IMU (ang. *Inertial Measurement Unit*) – akcelerometr i żyroskop GY-521 (Rys. 4.8). W przypadku wystąpienia poślizgu kół, enkodery inkrementalne opisane w podrozdziale 4.5.1 nie będą w stanie rozróżnić wynikowego ruchu pojazdu. Koła mogą się obracać, mimo że położenie robota w przestrzeni nie zmieni się. Sterownik może uważać w takim przypadku, że robot przebył pewną odległość, co będzie istotnym zakłamaniem. Dlatego, dopuszcza się możliwość odczytu danych inercyjnych. Możliwy jest odczyt przyspieszeń pojazdu wzdłuż trzech osi X, Y, Z. Dane te można dwukrotnie scałkować, co da wiedzę o drodze przebytej przez pojazd lub prędkości wzdłuż wymienionych osi (przy jednokrotnym całkowaniu). Ponadto, czujnik umożliwia określenie rotacji obiektu względem trzech osi współrzędnych, dlatego też jest to przydatna funkcjonalność w kontekście weryfikacji obrotu pojazdu w przestrzeni lub np. jego kąta nachylenia do podłoża. Akcelerometr i żyroskop opiera się na 16-bitowym przetworniku

analogowo-cyfrowym. Zakres pomiarowy to od ±250º/s do ±2000º/s (dla żyroskopu), a zakres mierzonego przyspieszenia to od ±2g do ±16g [11].

W aktualnym stopniu zaawansowania projektu, bezpośrednio odczytane (nieobrobione) dane z czujnika, są przekazywane do operatora. Komunikacja z czujnikiem odbywa się po szynie I^2C [11]. Dopuszcza się implementacje programu interpretującego wspomniane wielkości fizyczne, co wpłynie na zwiększenie jakości sterowania robotem.



Rys. 4.8 Akcelerometr GY-521 (MPU6050)

4.5.3 Magnetometr (Kompas)

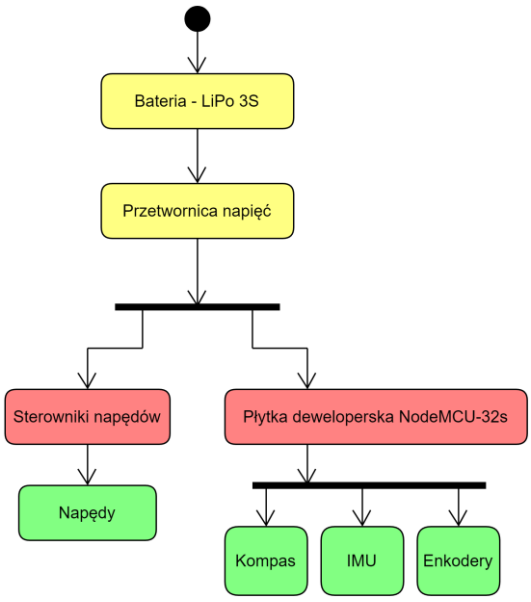
System sterowania robotem mobilnym wyposażono również w czujnik pola magnetycznego HMC-5883L (Rys. 4.9), który pozwala np. na określenie kierunku względem północnego bieguna magnetycznego Ziemi. Jest to przydatna funkcjonalność, umożliwiająca zorientowanie robota w przestrzeni, co może być zwłaszcza przydatne dla operatora znajdującego się np. w znacznej odległości od pojazdu lub może być przydatne w precyzyjnym określaniu i planowaniu obrotu platformy o np. zadany kąt. Dane odczytywane przez czujnik przesyłane są bezpośrednio do operatora. Konstrukcję czujnika oparto na sensorze magnetyczno-oporowym (zamiana strumienia magnetycznego na opór przewodnika) i 12-bitowym przetworniku analogowo-cyfrowym, co pozwala osiągnąć dokładność ok. 1-2º. Czułość elementu deklaruje się na zakres od 1 mili-Gaussa, do 8 Gaussów. Komunikacja z czujnikiem odbywa się po szynie I^2C [12].

Należy wspomnieć, iż umiejscowienie czujnika IMU oraz magnetometru w pojeździe, powinno być w geometrycznym środku platformy mobilnej, aby odczyty były zgodne z rzeczywistością (zostało to zaprezentowane na Rys. 3.2).

Rys. 4.9 Magnetometr HMC-5883L

4.6 Zasilanie

Zgodnie z określonymi wymaganiami w podrozdziale 2.2.2, postanowiono wyposażyć platformę mobilną w wydajne źródło zasilania, zapewniające dostateczną moc dla wykorzystanej elektroniki, ale również z zachowaniem pewnej nadwyżki (przewymiarowania) energii - z uwagi na możliwość rozbudowy robota mobilnego o inne moduły o dużym poborze prądu, w tym przede wszystkim mikrokomputery typu Raspberry Pi lub pokrewne. Sposób w jaki rozwiązano temat zasilania robota, przedstawiono hierarchicznie na Rys. 4.10.



Rys. 4.10 Diagram zasilania robota

4.6.1 Zasilanie bateryjne

W doborze prezentowanego poniżej źródła zasilania, rolę decydującą odegrały trzy czynniki: wydajność, pojemność baterii oraz jej dostępność w zasobach autora pracy.

W istocie, do wyboru na rynku jest wiele rodzajów akumulatorów. Różnią się one wieloma parametrami, co wpływa na np. długość działania urządzenia na baterii, ilość dostarczanej energii z akumulatora (jeśli byłaby ona zbyt mała, elektronika może w ogóle nie funkcjonować lub działać niestabilnie z bliżej nieokreślonymi błędami).

Prezentowane poniżej źródło zasilania (Rys. 4.11), to popularny akumulator litowo-polimerowy, charakteryzujący się dużą wydajnością prądową do 10A, i napięciu ok. 11.1V. Jest to tak zwany akumulator Li-Po 3S. Jak można zauważyć, maksymalny dostarczany prąd to 10A, który co prawda nie jest wykorzystywany w pełni, jednak zapewnia to bezpieczne warunki pracy dla samego akumulatora jak i również stabilne działanie sterownika, czujników i silników nawet przy ich maksymalnym obciążeniu.

Tabela 4.6 *Dane techniczne wykorzystanego akumulatora*

Zakres napięć:	minimalne: 9.3V	maksymalne: 12.6V
Nominalne napięcie operacyjne:	11.1V	
Pojemność:	1000mAh	
Wydajność prądowa:	zalecany ciągły prąd do 6A	maksymalnie: 10A
Maksymalny prąd ładowania:	1A	
Rodzaj baterii:	litowo - polimerowa (Li-Po)	
Ilość ogniw:	3S (3 ogniwa połączone szeregowo)	
Ogólne oznaczenie baterii:	Li-Po 3S 10C	

Rys. 4.11 Wykorzystane źródło zasilania

4.6.2 Przetwornica napięć

Podczas przeglądu ofert na rynku dostępnej elektroniki, postawiono na prostą i powszechnie stosowaną przetwornicę impulsową (Rys. 4.12), obniżającą napięcie prądu stałego. Jest to urządzenie składające się głównie ze stabilizatora(ów) napięcia, które dobrze dobrane nie generują dużej ilości strat charakteryzujących się oddawaniem ciepła do otoczenia. W przypadku robotyki mobilnej, (zwłaszcza np. w przypadku dronów – urządzeń latających) straty energii elektrycznej są bardzo istotne – im mniej, tym dłużej urządzenie może funkcjonować na baterii.

Tabela 4.7 *Dane techniczne wykorzystanej przetwornicy*

Napięcie wejściowe:	minimalne: 4V	maksymalne: 40V
Napięcie wyjściowe:	minimalne: 1.3V	maksymalne: 36V
Prąd wyjściowy:	rekomendowany: 5A	maksymalny: 8A
Maksymalna moc:	200W	
Efektywność konwersji:	94%	
Wykorzystywane napięcie wyjściowe:	5V	
Typ przetwornicy:	Step-Down (Obniżająca napięcie DC-DC)	
Zastosowany regulator napięcia:	XL4016E1	

Bazując na diagramie zasilania przedstawionym na Rys. 4.10, sterowniki napędów są zasilane napięciem prądu stałego o wartości 5V. Z kolei mikrokontroler sterujący – układ ESP32 na platformie NodeMCU – został wyposażony w dodatkowy wbudowany układ stabilizujący napięcie, co w zasadzie pozwala na zasilanie NodeMCU napięciem do 15V prądu stałego. Czujniki zostały zasilone z wyprowadzenia 3.3V dostępnego w NodeMCU. Pobór prądu przez czujniki jest niewielki, dlatego nie występuje ryzyko związane z przeciążeniem i uszkodzeniem wyjścia mikrokontrolera.

Rys. 4.12 Wykorzystana przetwornica napięć

4.7 Wykorzystane interfejsy komunikacyjne

Tworząc sterownik pokładowy robota, postanowiono skorzystać z ogólnodostępnych metod wymiany informacji pomiędzy wykorzystywanymi zewnętrznymi układami cyfrowymi (czujnikami), a mikroprocesorem.

4.7.1 Interfejs I^2C

W projekcie wykorzystano szynę I^2C (ang. *inter-integrated circuit*), która znalazła zastosowanie w komunikacji mikrokontrolera z zastosowanymi czujnikami (IMU oraz kompasem). Jest to bardzo prosta w implementacji metoda wymiany danych pomiędzy różnymi układami cyfrowymi, co bardzo ważne jest łatwo skalowalna – możliwe jest podłączenie do 128 urządzeń.

Interfejs I^2C umożliwia transmisję danych szeregowo, w dwóch kierunkach. Wykorzystywane są do tego dwie linie: pierwsza - SCL – (*Serial Clock Line*) – odpowiadająca za synchronizacje transmisji (przesyłane są takty zegarowe) oraz druga – SDA – (*Serial Data Line*) – dzięki której, dane są transmitowane dalej. Do sterowania interfejsem jest wymagany mikrokontroler lub mikroprocesor.

Należy wiedzieć, że w tym interfejsie zawsze mamy do czynienia z komunikacją typu „Master-Slave”. Wykorzystana płytka deweloperska NodeMCU-32S została wyposażona w odpowiednie wyprowadzenia, dzięki którym może pełnić funkcję tzw. „Mastera” – czyli jednostki zarządzającej komunikacją szyny I^2C. To z poziomu mikrokontrolera wydawane są polecenia – np. zapytania do podrzędnych układów cyfrowych (tzw. „Slave”) – w tym projekcie są to akcelerometr GY-521 obsługujący I^2C oraz magnetometr HMC-5883L. Każde z tych układów cyfrowych posiada swój adres, dzięki któremu każdy czujnik jest identyfikowalny w interfejsie I^2C [13].

4.7.2 Interfejs szeregowy UART

Zastosowany mikroprocesor ESP32-WROOM-32 pozwala na komunikacje z wykorzystaniem asynchronicznego interfejsu szeregowego UART (linie TX, RX) w różnorodny sposób – pierwszy, powszechnie stosowany – to połączenie z komputerem poprzez złącze micro-USB. Jest ono wykorzystywane w projekcie, do wyświetlania informacji o statusie programu, błędach i zdarzeniach (funkcja dla debugingu – sprawdzania poprawności działania). Kolejną możliwością, jest skorzystanie z dwóch niezależnych

portów, udostępnionych przez producenta mikrokontrolera. W tym przypadku wykorzystano jeden interfejs UART, którego wyprowadzenie można znaleźć na płytce PCB. Jest ono aktualnie nieużywane, lecz zostało przewidziane głównie pod czujnik LiDAR (ang. *Light Detection and Ranging*), które w wielu przypadkach wymieniają informacje właśnie poprzez ten interfejs.

W odróżnieniu do interfejsu I^2C, dane są przesyłane bez sygnałów synchronizujących, transmisja jest wykonywana bajt po bajcie. Co więcej, transmisja UART umożliwia równoczesne wysyłanie i nadawanie danych, stąd wymagane są dwie linie: TX (transmit – służąca do wysyłania danych) oraz RX (receive – służąca do odbioru danych) [13].

4.7.3 Komunikacja bezprzewodowa

Ponieważ postanowiono, że sterowanie robotem będzie odbywało się zdalnie z pomocą operatora, uznano iż dobrym wyborem będzie wybranie komunikacji bezprzewodowej (radiowej) w standardzie 802.11b/g/n - 2.4GHz, czyli tak zwane Wi-Fi. Układ nadajnika i odbiornika fal radiowych jest wbudowany w mikrokontroler ESP32 i zapewnia moc sygnału na poziomie do 15,5 dBm, a dla odbiornika czułość wynosi do -98 dBm [6]. Takie parametry transmisji definiują zasięg urządzenia na poziomie co najmniej kilku metrów w przypadku wysokiego tłumienia (przeszkody w postaci ścian itp.). Doświadczalnie sprawdzono, że tego typu mikrokontrolery posiadają zasięg około 15 metrów.

Układ mikrokontrolera będzie pełnił funkcję punktu dostępowego, do którego w celu sterowania robotem operator urządzenia będzie mógł się przyłączyć np. za pomocą komputera, bądź telefonu.

Rozdział 5

Oprogramowanie sterownika

W tym rozdziale, poruszona zostanie tematyka oprogramowania sterownika pokładowego zbudowanego robota mobilnego. Bazując na wymaganiach projektowych zawartych w podrozdziale 2.2.3, etap tworzenia oprogramowania rozpoczęto od rozplanowania przepływu informacji oraz określenia potrzebnych funkcjonalności do zaimplementowania.

5.1 Problematyka, przedstawienie hierarchii

W robotyce istnieje bardzo silne powiązanie pomiędzy sprzętem a programem. Platforma jezdna spełniła wszelkie założenia, zatem „umiejętność" robota do poruszania się, zależy w głównej mierze od złożoności programu sterującego. Mamy zatem do czynienia z oprogramowaniem niskopoziomowym, gdyż program wykonuje się bezpośrednio w sterowniku (mikrokontrolerze) i jest oprogramowany za pomocą języka C/C++. Sygnały sterujące generowane przez program powinny być przekazywane bezpośrednio na elementy wykonawcze, dlatego z planistycznego punktu widzenia należy rozróżnić pięć głównych składowych realizowanego programu, których funkcjonalność została zaprezentowana na Rys. 5.1.

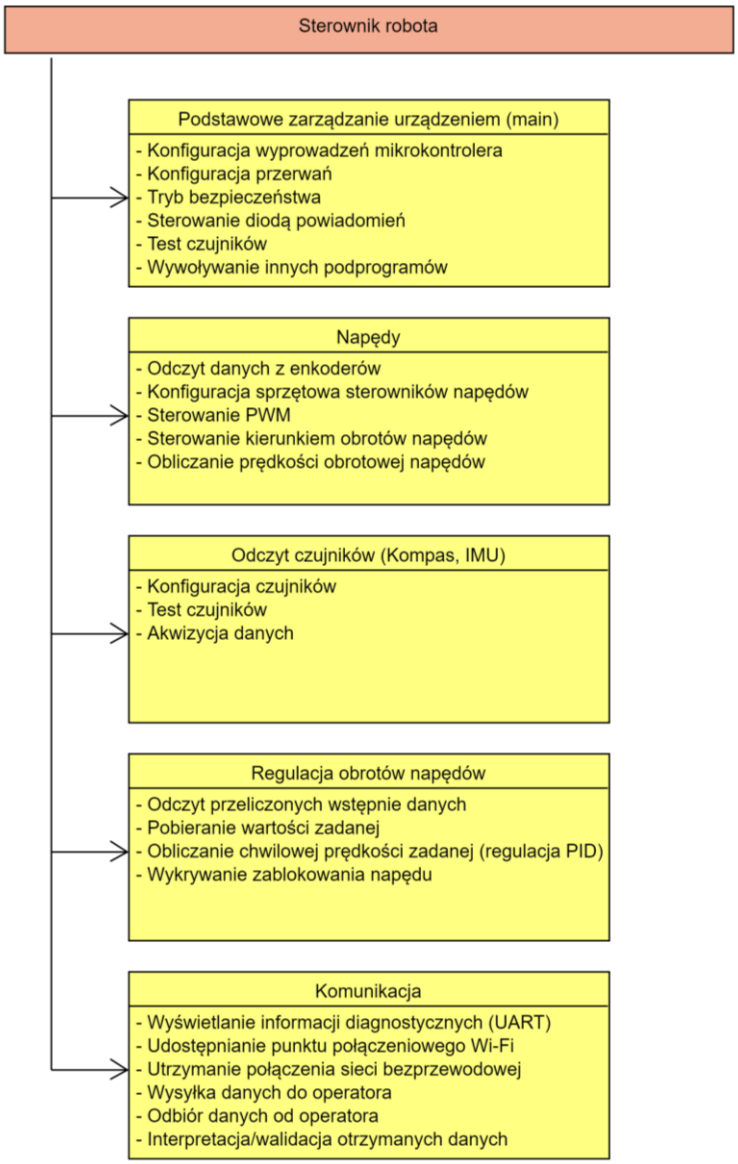
Pierwsza sekcja składająca się na wynikowy program sterownika, zawiera instrukcje dotyczące konfiguracji poszczególnych wyprowadzeń mikrokontrolera (kierunku działania – wejście/wyjście, przerwanie), konfiguracji interfejsów komunikacyjnych oraz deklaracji podstawowych zmiennych globalnych wykorzystywanych przez program.

Druga sekcja programu poświęcona jest napędom robota i związanymi z tym strategiami sterowania i akwizycją danych pomiarowych z enkoderów inkrementalnych.

Trzecią sekcję stanowią zagadnienia dotyczące obsługi czujników pokładowych robota, głównie akwizycji danych.

Czwarta sekcja programu zawiera mechanizmy regulujące prędkości obrotowe silników oraz opcjonalnie wykrywanie zablokowania napędu i podjęcie odpowiedniej akcji chroniącej sterownik przed przegrzaniem/ewentualnym uszkodzeniem.

Na piątą część oprogramowania przypada cała część odpowiedzialna za komunikację robota z zdalnym punktem docelowym – tj. operatorem oraz wyświetlanie dodatkowej informacji diagnostycznej przez interfejs UART.



Rys. 5.1 Główne funkcje programu mikrokontrolera

5.2 Wykorzystane oprogramowanie

Do zaprogramowania mikrokontrolera ESP32, posłużył dostępny w Visual Studio Code pakiet PlatformIO, wykorzystujący zintegrowane środowisko programistyczne Arduino IDE (ang. *integrated development environment*). Ponadto oprogramowanie Visual Studio Code umożliwiło wygenerowanie diagramów, schematów blokowych dzięki dodatkowi o nazwie „UMLet".

Pakiet rozszerzeń PlatformIO umożliwił wygodną pracę nad programowaniem mikrokontrolera. Oprogramowanie to, po odpowiednim skonfigurowaniu umożliwia automatyczne dołączanie bibliotek do pisanego programu, identyfikację i konfiguracje programowanego mikrokontrolera, automatyzację procesu kompilacji, a nawet bezpośredni odczyt z monitora portu szeregowego, co znacznie ułatwiło pracę nad projektem.

5.3 Rozwiązania programowe

W celu lepszej przejrzystości programu struktura plików nagłówkowych została podzielona analogicznie do opisu z podrozdziału 5.1 tj. kolejno: *main.cpp, MotorsDC.h, Sensors.h, MotorsPID.h, Communication.h*.

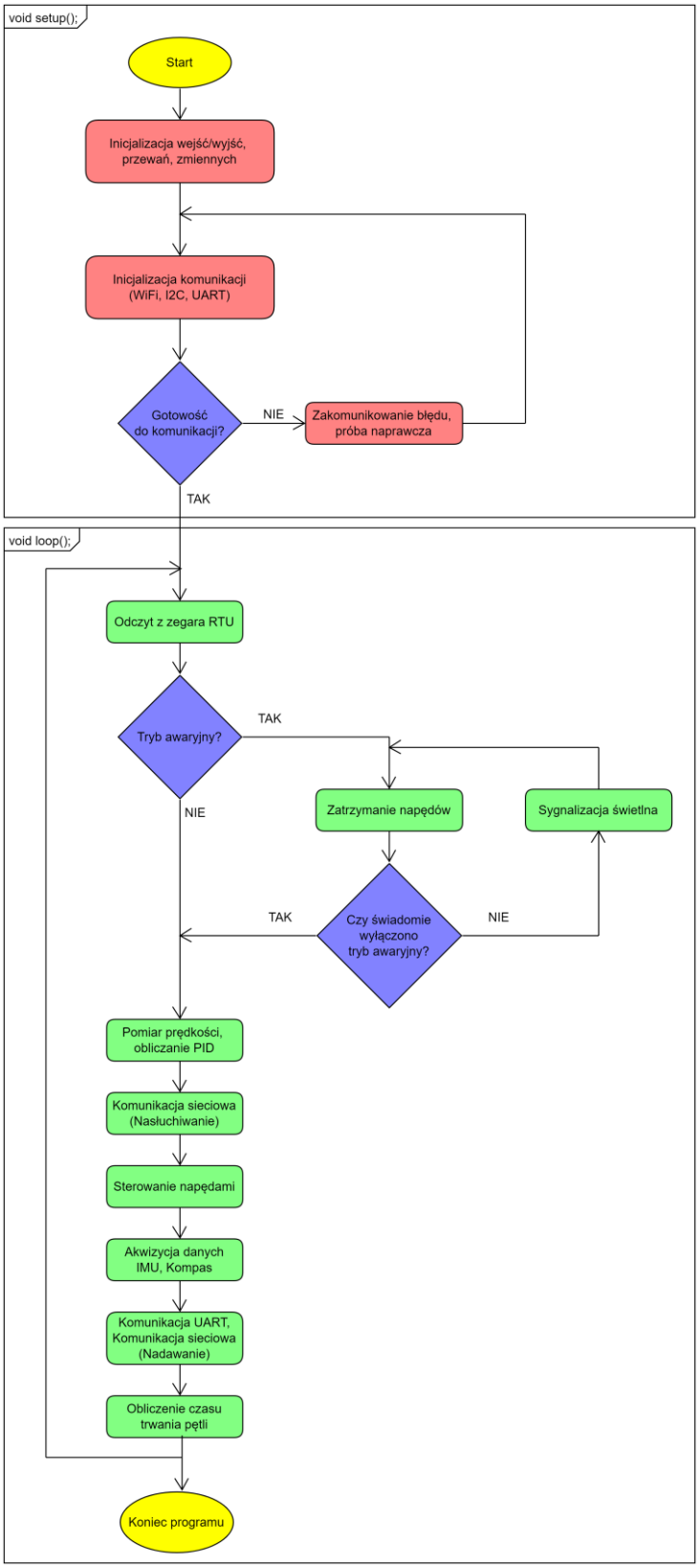
5.3.1 Podstawowe zarządzanie systemem robota mobilnego

Tworząc oprogramowanie z wykorzystaniem Arduino-IDE, program wykonywany przez mikroprocesor składa się z dwóch części (technicznie rzecz ujmując funkcji) *setup()* oraz *loop()*. Pierwsza część programu jest wywoływana w momencie uruchomienia mikrokontrolera. W tym miejscu odbywa się inicjalizacja zmiennych, przerwań, połączeń (I^2C, UART, WiFi). Po poprawnym stworzeniu punktu dostępowego program przechodzi do funkcji *loop()*, wywołującej się cały czas. Jest to pętla nieskończona. Na początku każdej iteracji pętli pobierana jest aktualna wartość z zegara czasu rzeczywistego, dzięki której jest możliwe obliczenie czasu iteracji pętli, który jest wykorzystywany przy obliczaniu prędkości obrotowej napędów. Następnie sprawdzany jest warunek, czy został wywołany tryb awaryjny urządzenia. Samo zasygnalizowanie tego trybu odbywa się poprzez naciśnięcie przycisku dostępnego na górze pojazdu, który zmienia status flagi *EMERGENCY_MODE_TRIGGER* sprawdzanej przy każdej iteracji pętli *loop()*.

Jeśli urządzenie przejdzie w stan awaryjny – będzie w nim trwało do czasu świadomego wyłączenia (tj. przytrzymania tego samego przycisku przez ok. 8 sekund) zatrzymując natychmiastowo wszystkie napędy i sygnalizując stan wyjątkowy pulsacyjnym sygnałem świetlnym z wykorzystaniem diody LED umieszczonej nieopodal opisywanego przycisku „bezpieczeństwa".

Zakładając scenariusz poprawnego działania pojazdu, po sprawdzeniu opisanego powyżej warunku następuje odczyt zgromadzonych danych z enkoderów inkrementalnych – konkretniej – następuje odczyt zliczonych wartości (zwiększenie licznika o jeden odpowiada jednemu pełnemu obrotowi wału silnika DC). Odczytane w ten sposób „surowe" dane, zostają przeliczone na prędkość obrotową (obroty/min). Następnie, sterownik nasłuchuje, czy nie zostały do niego wysłane komendy. Ewentualne dane o kierunku i prędkości zadanej są parsowane (wyłuskiwane i sprawdzane) i przypisywane do odpowiednich tymczasowych zmiennych. Posiadając dane o aktualnej i zadanej prędkości poszczególnych z kół, algorytm PID oblicza wypełnienie sygnału PWM dla każdego z napędów. Obliczona wartość PWM zostaje przekazana do funkcji realizującej bezpośrednią obsługę sterownika silników (tj. rozpędzenie silnika, nadanie kierunku jazdy, itd.). Następnie z czujników pokładowych – IMU oraz kompasu gromadzone są surowe dane wystawione przez każdy z przetworników ADC tychże czujników. Po odczytaniu danych, całość informacji o pojeździe zostaje nadawana do operatora.

Opisany powyżej cykl programu wykonuje się z bardzo dużą częstotliwością (procesor jest taktowany zegarem 240 MHz), zatem można przyjąć że program wykonuje się w sposób „ciągły". Operator robota chcąc zakończyć program, powinien wpierw zatrzymać pojazd, a następnie zwyczajnie go wyłączyć wykorzystując do tego główny wyłącznik zasilania.



Rys. 5.2 Schemat blokowy programu sterownika

5.3.2 Obsługa napędów

Sterownik silników MX1508 wymaga dwóch kanałów wejściowych sterujących (oznaczonych jako *Motor A, Motor B*), na które przekazywany jest sygnał PWM z mikrokontrolera. Sposób wysterowania napędu przedstawiono w poniższej tabeli.

Tabela 5.1 *Metodyka obsługi sterownika MX1508*

Sterowanie MX1508		
Wyprowadzenie		Działanie Wynikowe
Motor A	*Motor B*	
0…254	255	Obrót do przodu
255	0…254	Obrót do tyłu
255	255	**STOP**

Zmiany wypełnienia sygnału na jednym z kanałów sterujących powodują ruch napędu. Co istotne, producent sterownika założył, iż podanie zerowego (0%) wypełnienia sygnału PWM na wejścia sterownika, stanowi tzw. stan wysoki – tj. maksymalną wartość napięcia jaka jest dostarczana do silnika DC. Oznacza to, że napęd będzie działał z maksymalną mocą. Każda wartość zmierzająca do wartości 255, będzie spowalniała napęd – napięcie wyjściowe sterownika będzie obniżane. Przedstawiony powyżej zakres liczb 0-255 wynika z 8-bitowego kanału PWM mikrokontrolera. Wartość 0 oznacza wypełnienie sygnału o wielkości 0% (zatem fizycznie 0V napięcia), a wartość 255 to maksymalne wypełnienie sygnału – 100% (fizycznie 5V napięcia).

Z punktu widzenia programistycznego, zdefiniowano wyprowadzenia mikrokontrolera służące do sterowania napędem oraz wyprowadzenia mikrokontrolera odpowiedzialne za każde z czterech enkoderów inkrementalnych. Dane odpowiedzialne za liczniki enkoderów i prędkości poszczególnych z napędów umieszczono w strukturach danych *struct ENC_VAL* oraz *struct SPEED*.

Do sterowania napędami zaimplementowano dwie funkcje *void moveMotor()* oraz *void stopMotor()*. Pierwsza odpowiedzialna jest za wysterowanie sygnałów PWM dla sterownika, wedle przekazanego argumentu dot. wielkości wypełnienia, kierunku obrotu oraz złączy sterownika dla konkretnego napędu, którego sytuacja dotyczy. Druga odpowiada za zatrzymanie wszystkich napędów – wykorzystywane np. w trybie awaryjnym.

W projekcie przyjęto, że kierunek obrotów jest ściśle ustalony, dzięki czemu zaoszczędzono kilka wejść w mikrokontrolerze. Wiedza o kierunku jest przechowywana

w zmiennych globalnych, które dla poruszania w przód przyjmują wartość 1, z kolei dla ruchu w tył wartość -1. Jest to podyktowane sposobem zliczania impulsów generowanych przez enkodery. Czynność ta odbywa się z wykorzystaniem prostego wzoru:

$$E_n = E_{n-1} + dir \tag{1}$$

gdzie:

E_n - aktualna wartość licznika

E_{n-1} – poprzednia wartość licznika

$dir \in \{-1; 1\}$ – kierunek obrotu

Dzięki temu, w zależności od ustalonego kierunku obrotów koła następuje dekrementacja lub inkrementacja licznika. Taki zabieg był możliwy, ponieważ każde koło jest połączone bezpośrednio z przekładniami, przez co nie są w stanie obracać się swobodnie (w takim przypadku, wymagana byłaby sprzętowa identyfikacja kierunku obrotów).

5.3.3 Kontrola i obliczanie prędkości napędów

Obliczanie prędkości napędów odbywa się poprzez porównanie zmian wielkości każdego z liczników w czasie wykonania jednej iteracji pętli *loop()* i uwzględnieniem rozdzielczości zastosowanych enkoderów.

Wykonywane jest następujące obliczenie:

$$n = \frac{(E_n - E_{n-1})}{t_{loop} * \frac{1}{1000}} * \frac{60}{16 * 36} \tag{2}$$

gdzie:

n – częstotliwość wyjściowa napędu (RPM – obroty/min)

E_n – aktualna wartość licznika

E_{n-1} – wartość licznika przy poprzedniej iteracji

t_{loop} – czas (w milisekundach) wykonywania się ostatniej iteracji pętli *loop()*

Następnie obliczona wartość jest przeliczana na prędkość kątową wg. uniwersalnego wzoru:

$$\omega = n * \frac{2\pi}{60} \tag{3}$$

gdzie:

ω – prędkość kątowa napędu (rad/s)

n – prędkość obrotowa napędu (RPM – obroty/min)

Posiadając informacje nt. prędkości obrotowej napędu, można zastosować układ regulatora proporcjonalno-całkująco-różniczkującego – PID (ang. *proportional-integral-derivative*). Jest on wykorzystywany do utrzymania względnie stałej wartości obrotów napędu, tak aby robot poruszał się ze stałą prędkością włącznie z uwzględnieniem zjawisk utrudniających utrzymanie prędkości – opory, zmiana nawierzchni, zablokowanie napędów. Składa się z trzech członów – proporcjonalnego, całkującego i różniczkującego.

Algorytm zaimplementowano w pliku *MotorsPID.h*. Do tego celu stworzono strukturę danych przechowującą parametry algorytmu, aktualne obliczenia oraz wartości minimalne i maksymalne wysterowania. Z uwagi na charakter zastosowania, jest to dyskretny algorytm PID i wykonywane są następujące obliczenia:

$$e = s_p - n \tag{4}$$

$$I = \frac{T_{próbkowania} * (e_n + e_{n-1})}{2} + I_{n-1} \tag{5}$$

$$\Delta_e = \frac{e - e_n}{T_{próbkowania}} \tag{6}$$

$$y = K_p * e + K_d * \Delta_e + K_i * I \tag{7}$$

Równaniem (4) obliczany jest aktualny uchyb regulatora (tj. różnica pomiędzy wartością zadaną, a aktualną prędkością obrotową napędów). Następnie obliczana jest całka – równanie (5) - (metodą prostokątów – numerycznie) z uchybu (brana jest pod uwagę aktualna wartość i poprzednia uchybu oraz dodawana jest poprzednia wartość z iteracji całkowania). Równanie (6) odpowiada za część różniczkującą regulatora. Finalnie w równaniu (7) następuje sumowanie wszystkich członów z uwzględnieniem stałych regulatora. Wyjściową wielkością jest prędkość obrotowa. Regulacja odbywa się dla każdego napędu z osobna. Z uwagi na sposób sterowania napędami, który został opisany w podrozdziale 5.3.2, wartość wyjściowa musi zostać dodatkowo przeliczona wg. wzoru (8)

$$y_{przeliczone} = 255 - \frac{y * 255}{y_{max}} \quad (8)$$

gdzie:

$y_{przeliczone}$ – wysterowanie przeliczone na wypełnienie PWM

y_{max} – maksymalna wartość wysterowania - wynosi 156 RPM. Całkowity zakres sterowania mieści się w przedziale $< 0, 156 >$ (wynika to z zakresu prędkości obrotowej napędów).

Strojenie regulatora odbyło się doświadczalnie. Wzmocnienie K_p dobierano sprawdzając czy napęd porusza się z prędkością bliskiej zadanej (jeśli prędkość była za niska, zwiększano wzmocnienie, jeśli była za wysoka – przez co następowały oscylacje, układ chodził nierówno, zmniejszano wzmocnienie). Stała K_d podczas zwiększania pozwalała na nieznaczne zmniejszenie przeregulowania, czasu narastania i regulacji. Stała K_i w przypadku zwiększenia mogła spowodować zmniejszenie czasu narastania, kosztem wzrostu przeregulowania i pogorszenia stabilności. Finalnie po wielu różnych konfiguracjach, wystrojono regulator tak, aby spełniał swoje zadanie nie tworząc tym efektu „szarpania" i zrywania napędów.

5.3.4 Obsługa czujników

Do obsługi czujników IMU oraz magnetometru wykorzystano dedykowane biblioteki o nazwie *Adafruit_MPU6050.h* oraz *QMC5883LCompass.h.* Zawierają one metody umożliwiające odczyt surowych danych, jak i również przeliczonych. Biblioteka *QMC5883LCompass.h* zawiera specjalną metodę umożliwiającą natychmiastowe określenie kierunku (tj. zamiast danych nieprzetworzonych, można od razu odczytać czy robot jest zorientowany na np. północny wschód – NE, czy południowy zachód SW). Komunikacja z wykorzystanym interfejsem I^2C odbywa się praktycznie bez wiedzy programisty i nie wymaga konfiguracji. Podane biblioteki robią to automatycznie, bez ingerencji programisty. Inicjalizacja komunikacji (tym samym test czujników) odbywa się w głównej funkcji *setup()* programu *main.cpp*.

5.3.5 Komunikacja

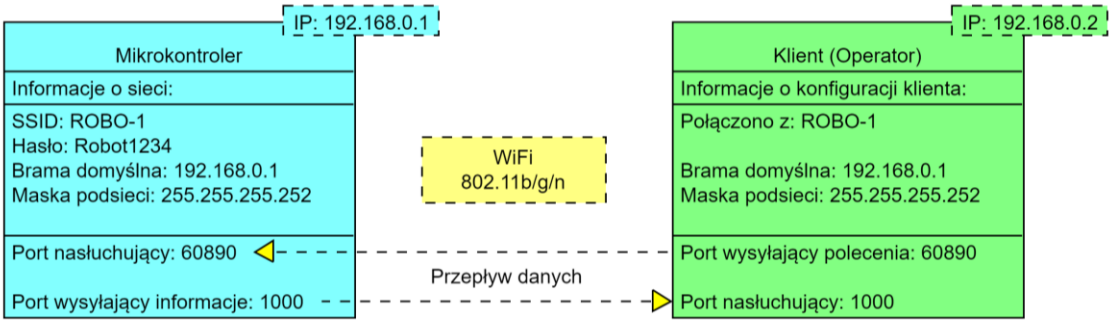
Jak opisywano w poprzednich rozdziałach pracy, zastosowano komunikację bezprzewodową Wi-Fi. Do obsługi połączenia wykorzystano bibliotekę *Wifi.h* (Arduino-IDE) stosowaną do połączeń z mikrokontrolerami typu Arduino lub ESP. Biblioteka ta umożliwia:

- przyłączenie się do innych istniejących punktów dostępu,
- tworzenie nowych punktów dostępowych sieci (jak w tym przypadku),
- komunikacje za pomocą protokołu UDP (ang. *User Datagram Protocol*) lub TCP/IP (ang. *Transmission Control Protocol/Internet Protocol*).

W projekcie wykorzystano protokół TCP/IP z uwagi na jego większą niezawodność w stosunku do protokołu UDP. W protokole połączeniowym istotne jest nawiązanie komunikacji z punktem docelowym. W protokole bezpołączeniowym jakim jest UDP, gwarancji odbioru pakietów po prostu nie ma, co w przypadku sterowania pojazdem jest niedopuszczalne. Operator w takim przypadku mógłby wysyłać polecenia do urządzenia, jednak nie miałby informacji konkretnie o tym, czy dane zostały dostarczone, co mogłoby doprowadzić do braku jakiejkolwiek kontroli nad urządzeniem.

Struktura logiczna stworzonej sieci opiera się de facto na architekturze „serwer-klient". Usługa rozgłoszeniowa jest po stronie mikrokontrolera czyli serwera. Stacją kliencką nazywany będzie np. komputer, tablet lub smartfon – w zależności od przyłączanego przez operatora urządzenia w celu sterowania pojazdem.

Warto wspomnieć o sposobie adresacji utworzonej sieci komputerowej. Wykorzystano tutaj 30-bitową maskę podsieci (*255.255.255.252/30*), dzięki czemu możliwe jest stworzenie podsieci tylko dla dwóch urządzeń – w tym przypadku - bramę domyślną (główny węzeł sieci) stanowi mikrokontroler, a pozostały (jeden) wolny adres IP jest przydzielany dla podłączającego się klienta sieci (czyli operatora urządzenia). Każde nadmiarowe urządzenie jest separowane, gdyż przydzielany jest mu adres z puli dla kolejnych podsieci, przez co teoretycznie nie ma możliwości komunikacji. Diagram stworzonej sieci przedstawiono na Rys. 5.3. Do wysyłania danych zgromadzonych przez pojazd wykorzystano port 1000, z kolei wysyłanie poleceń operatora odbywa się z wykorzystaniem portu 60890. Jest to stosunkowo odległy port komunikacyjny w systemach komunikacyjnych i nie powinien kolidować z różnymi usługami działającymi w systemach operacyjnych (np. FTP - port 21 lub RDP - port 3389).



Rys. 5.3 Struktura połączeniowa stworzonej sieci

Obecnie w celu zdalnej obsługi pojazdu należy skorzystać z darmowego oprogramowania PacketSender (Rys. 5.4), które umożliwia bezpośredni odbiór jak wysyłanie danych do urządzenia.
Ponieważ mamy do czynienia ze standardem komunikacyjnym, stworzenie dedykowanej aplikacji sterującej robotem jest możliwe dla każdego z urządzeń, które jest w stanie połączyć się bezprzewodowo (np. smartfon, laptop, tablet).
Komunikacja z robotem jest możliwa dzięki poleceniom wydawanym przez operatora:

E_Mode*, *DirA*, *ValA*, *DirB*, *ValB*, *DirC*, *ValC*, *DirD*, *ValD

gdzie:
E_mode – pojazd w trybie ruchu (0) lub całkowitego zatrzymania (1)
DirA, DirB, DirC, DirD – kierunek obrotu każdego z napędów (do przodu 1, wstecz -1)
ValA, ValB, ValC, ValD – zdana prędkość obrotowa (RPM) każdego z napędów

Przykład użycia:
Zakładamy scenariusz jazdy na wprost z prędkością obrotową 50RPM:

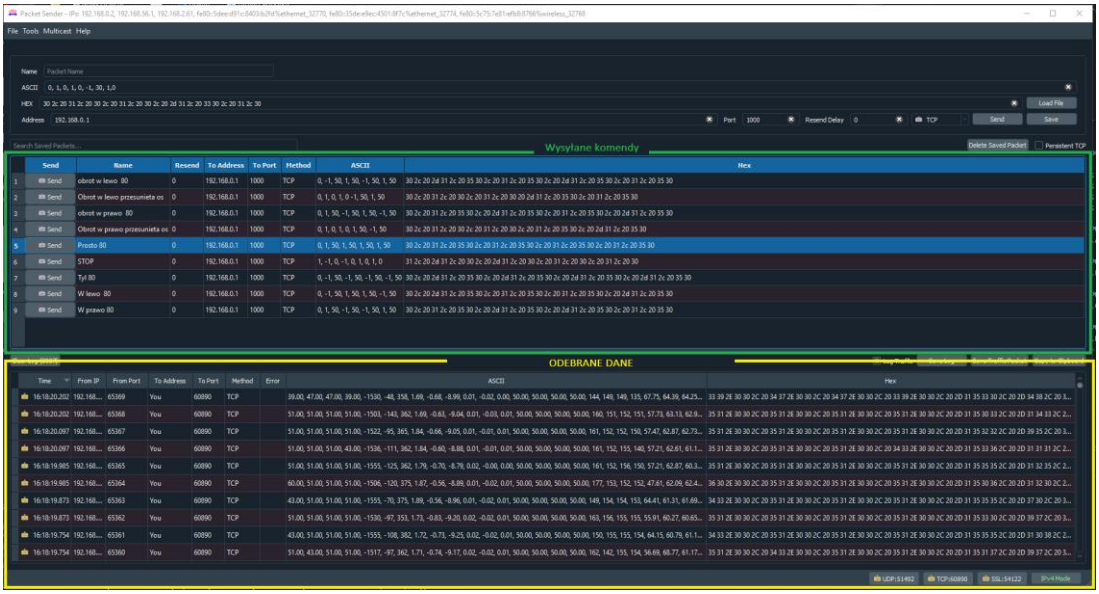
0*, *1*, *50*, *1*, *50*, *1*, *50*, *1*, *50

Zakładamy scenariusz jazdy do tyłu z prędkością obrotową 80RPM:

0*, *-1*, *80*, *-1*, *80*, *-1*, *80*, *-1*, *80

Zatrzymanie pojazdu:

1*, *-1*, *80*, *-1*, *80*, *-1*, *80*, *-1*, *80 lub: ***0*, *1*, *0*, *1*, *0*, *1*, *0*, *1*, *0***



Rys. 5.4 Widok komunikacji z programu PacketSender

W oparciu o przykładowe polecenia, można w analogiczny sposób tworzyć strategie sterowania robotem, które zostały przedstawione w podrozdziale 3.1.3 na Rys. 3.7. Dzięki temu, sposób sterowania robotem jest łatwy do zastosowania w bardziej zaawansowanym oprogramowaniu kontrolującym robota. Wysyłane przez operatora informacje są parsowane i następnie przekazywane do odpowiednich zmiennych, które mają bezpośredni wpływ na sterowanie robotem.

Sterownik pojazdu wysyła do operatora następujące dane:

- zmierzoną aktualną częstotliwość obrotów (RPM) każdego z napędów
- zmierzoną orientację względem bieguna północnego Ziemi (kompas)
- zmierzone przyspieszenia wzdłuż osi pojazdu (akcelerometr i żyroskop)
- informacje o odczytanej przez sterownik prędkości zadanej każdego z napędów (dane głównie diagnostyczne, w celu weryfikacji poprawności działania)
- wartość wysterowania (wypełnienia) sygnału PWM dla każdego z napędów
- wartość obliczona przez algorytm PID częstotliwości obrotów (RPM) każdego z napędów

Struktura wysyłanych przez sterownik danych wygląda analogicznie jak dla prezentowanych wcześniej poleceń. Separatorem danych jest przecinek „ **,** ”, przesyłane liczby zmiennoprzecinkowe jako separator dziesiętny wykorzystują kropkę „ **.** ” zatem nie dochodzi do konfliktu w interpretacji danych.

W przypadku komunikacji diagnostycznej z wykorzystaniem interfejsu szeregowego UART, oprócz identycznego pakietu danych dołączane są jeszcze czasy wykonywania każdej z iteracji pętli *loop()*.

Zaletą takiego rozwiązania jest szybka akwizycja danych lub wyświetlenie wartości w postaci generowanych w czasie rzeczywistym wykresów (funkcja „Kreślarki" w Arduino-IDE). Istnieją programy gromadzące wszystkie komunikaty z interfejsu szeregowego komputera. Ponieważ separatorem danych jest przecinek, tak przygotowane dane można łatwo przekształcić do formatu CSV (ang. *Comma-separated values*) celem dokładniejszej analizy danych np. w arkuszach kalkulacyjnych. Na tym etapie rozwoju projektu jest to bardzo przydatna funkcjonalność.

Rozdział 6

Weryfikacja i walidacja

Realizowany projekt musiał zostać przetestowany. Pierwsze testy były wykonywane tuż po wydrukowaniu platformy jezdnej i zamocowaniu napędów. Wykorzystano do tego zasilacz laboratoryjny, który podłączono bezpośrednio do konektorów odpowiedzialnych za podawanie napięcia do silników DC. W ten sposób po odpowiednim spolaryzowaniu napięcia na każdy z silników, można było sprawdzić, czy robot jest w stanie się poruszać w bok.

Początkowo występował problem z liniowością ruchu w bok lub na skos. Wynikało to z dużego poślizgu kół. Problem rozwiązano po nałożeniu koszulek termokurczliwych na każdą z rolek kół Mecanum. Przyczepność znacznie się zwiększyła, nawet na szklistych powierzchniach (takiej jak glazura o wysokim połysku) możliwy jest prostolinijny ruch w każdym kierunku.

Następnie po wykonaniu płytki PCB, przed zamontowaniem elementów elektronicznych dokonano sprawdzenia ciągłości obwodu. Testy wypadły pomyślnie. Przystąpiono do montażu elementów. Działanie układu było poprawne, z wyjątkiem jednego przypadku. Problem ujawnił się w momencie podłączania mikrokontrolera poprzez port micro-USB, przy wyłączonym zasilaniu bateryjnym. Było to zjawisko niepożądane, ponieważ wzbudzały się inne układy, w tym wysoko obciążające sterowniki napędów. Gdyby program mikrokontrolera rozpoczął sterowanie napędami, mogłoby to przeciążyć układ a w skrajnym przypadku uszkodzić port USB w komputerze. Dokonano drobnej poprawki w postaci zamontowania diody prostowniczej 1N4007 szeregowo w linie VCC (+5V) doprowadzającą napięcie z płytki PCB do mikrokontrolera. Problem został rozwiązany, po podłączeniu sterownika przez port micro-USB cała elektronika robota nie jest zasilana - możliwe jest tylko programowanie mikrokontrolera.

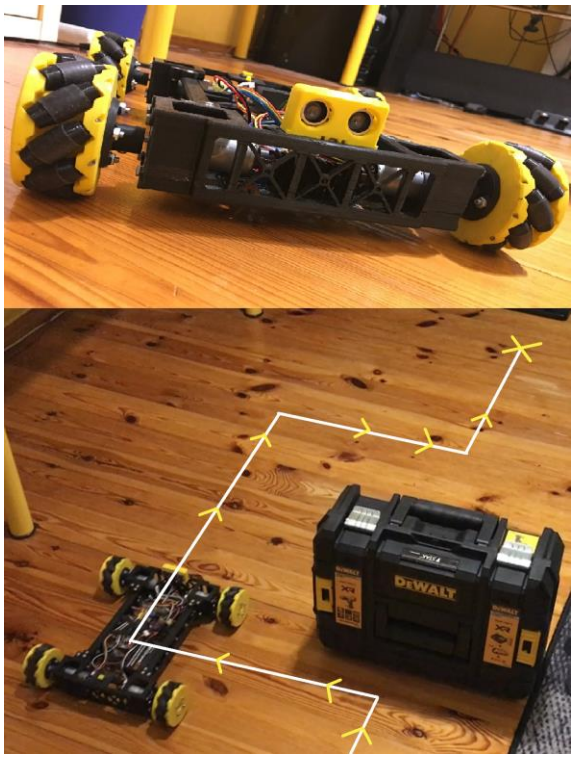
Tworzenie oprogramowania odbywało się w sposób modułowy – tzn. każdy element (np. sterownik silników) był programowany i testowany jako odrębny element (podprogram). W końcowej fazie programowania, wspomniane mniejsze fragmenty funkcjonalnego kodu

zostały połączone w całość. Weryfikacja działania polegała na kontrolowaniu wiadomości otrzymywanych przez interfejs UART. W każdym istotnym fragmencie kodu zawarta była instrukcja wysyłająca informacje z nazwą obecnego miejsca w kodzie. Dzięki temu było wiadomo, czy program wykonuje się poprawnie zgodnie z ustaloną kolejnością.

6.1 Przykład wykorzystania

Po sprawdzeniu poprawności działania i podstawowych jazdach testowych robotem, dodano ultradźwiękowy czujnik odległości (HC-SR04), który wykorzystano do prostej prezentacji unikania przeszkód. Zadaniem robota było ominięcie przeszkody z wykorzystaniem dostępnych dla tego typu pojazdów możliwości ruchu. Robot po wykryciu przeszkody zmienia strategie sterowania do jazdy w bok. Porusza się w ten sposób do momentu wykrycia pustej przestrzeni przed sobą. W ten sposób przeszkoda zostaje ominięta.

Wykazano w ten sposób podatność platformy mobilnej na modyfikacje (w postaci dołączenia dodatkowego czujnika) i zaprezentowano duży potencjał w kategorii autonomicznego podejmowania decyzji.



Rys. 6.1 Modyfikacja robota i zadanie ominięcia przeszkody

Rozdział 7

Podsumowanie i wnioski

Temat realizowanej pracy inżynierskiej wymagał wiedzy z wielu odmiennych tematów z dziedziny elektroniki, informatyki, projektowania wspomaganego komputerowo jak i również pośrednio mechaniki. Był to projekt wymagający, czasochłonny, o dużej złożoności.

Cel pracy jakim było stworzenie robota umożliwiającego jazdę w wielu kierunkach odpowiednich dla robota klasy (3,0) został osiągnięty. Zrealizowano prosty program obsługujący skonstruowany sterownik. Zastosowano skuteczny sposób komunikacji i sterowania pojazdem poprzez przesyłanie do sterownika robota krótkiej wiadomości tekstowej.

Problematyka tego projektu znacznie poszerzyła wiedzę i kompetencje autora projektu nt. budowy, sterowania i programowania robotów mobilnych i szeroko pojętego obcowania z systemami mikroprocesorowymi oraz ich programowaniem. Ponadto rozwinęła umiejętności organizacyjne i planistyczne. Brak systematyczności i wytrwałości w tak złożonym projekcie, mogłoby doprowadzić do negatywnych skutków w jego realizacji.

Zrealizowany system sterowania ma szerokie możliwości rozwoju. W głównej mierze jest to modyfikacja oprogramowania – rozumiana poprzez obsługę wyjątków, sterowanie na podstawie zadanej trajektorii lub realizacji autonomicznego sterowania robotem (np. z wykorzystaniem czujnika LiDAR). Bardzo interesującym wydaje się wykorzystanie środowiska programistycznego ESP-IDF, które (w odróżnieniu od Arduino-IDE) umożliwia implementacje systemu czasu rzeczywistego FreeRTOS wspierającego w pełni możliwości wykorzystywanego mikrokontrolera. W systemach złożonych, jakim jest z pewnością sterownik robota, wykonywanie wielu zadań w czasie zdeterminowanym (ściśle określonym) jest bardzo pożądane. Możliwość rozwoju w tym kierunku, otwiera nowe możliwości, zmierzające głównie ku autonomii robota w otaczającym go środowisku.

Nowym równie pożądanym kierunkiem, może być stworzenie dedykowanej aplikacji sterującej robotem z poziomu operatora. Główną ideą tego projektu jest zdalne podejmowanie decyzji, na podstawie przesłanych przez sterownik danych o pojeździe. Może to być aplikacja okienkowa (stacjonarna) lub mobilna, umożliwiająca sterowanie w trybie ręcznym, jak i automatycznym, w którym decyzje o wysterowaniu poszczególnych napędów realizowałaby aplikacja na podstawie przesłanych przez sterownik danych. Dodatkowo aplikacje można wyposażyć programowo w elementy wizualizacji parametrów zbieranych przez zastosowane czujniki (np. przechył urządzenia, zorientowanie względem bieguna północnego Ziemi) lub w możliwość wyznaczania trajektorii pojazdu.

Powyższe przykłady mogą dać nowy kierunek rozwoju projektu i jego nową, lepszą jakość.

Podsumowując, projekt systemu sterowania robotem mobilnym, wyposażonego w koła szwedzkie (typu Mecanum) jest projektem zgodnym z wypracowanymi założeniami i zrealizowanym do końca. Istnieje duży potencjał rozwoju platformy mobilnej, w tym samego oprogramowania robota, które mogłoby poruszać jeszcze bardziej złożone kwestie z tematu robotyki, sterowania i kontroli pojazdów zdalnie sterowanych lub autonomicznych.

Bibliografia

[1] P. Corke, Robotics, Vision and Control. Fundamental algorithms in Matlab, Springer, 2017.

[2] B. Siciliano i O. Khatib, „Handbook of Robotics," Springer, 2008.

[3] HowToMecatronics.com, „Projekt kół mecanum," HowToMecatronics, 2020. [Online]. Available: https://howtomechatronics.com/projects/arduino-mecanum-wheels-robot/. [Data uzyskania dostępu: 20 Czerwiec 2021].

[4] P. Rychlik i W. Kaczmarek, „Artykuł naukowy - Mechanik vol. 90, no.7," w *Design of Mobile Holonomic Robot With Wireless Inertial Measurement Control System*, July 2017, pp. 634-636.

[5] AiThinker, „NodeMCU ESP32 38 pin," AiThinker, [Online]. Available: https://docs.ai-thinker.com/_media/esp32/docs/nodemcu-32s_product_specification.pdf. [Data uzyskania dostępu: 01 Grudzień 2021].

[6] Espressif Systems, „ESP32-WROOM-32 Datasheet," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. [Data uzyskania dostępu: 10 Grudzień 2021].

[7] Texas Instruments, „L293x Half-H Drivers Datasheet," Texas Instruments, 2016. [Online]. Available: https://www.ti.com/lit/ds/symlink/l293.pdf. [Data uzyskania dostępu: 20 Sierpień 2021].

[8] Sparkfun, „ST L298 Dual Full-Bridge Driver Datasheet," ST Electronics, 2000. [Online]. Available: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf. [Data uzyskania dostępu: 20 Sierpień 2021].

[9] C. 101, „Using the MX1508 Brushed DC Motor Driver with an Arduino," 2018. [Online]. Available: https://components101.com/sites/default/files/component_datasheet/MX1508-DC-Motor-Driver-Datasheet.pdf. [Data uzyskania dostępu: 20 Sierpień 2021].

[10] MagnTek, „MT1450-EN Series Dual Channel Quadrature Switch Hall Sensors,” [Online]. Available: http://www.magntek.com.cn/upload/MT1450-EN.pdf. [Data uzyskania dostępu: 10 Grudzień 2021].

[11] InvenSense, „MPU-6000/6050 Product Specification,” [Online]. Available: https://www.haoyuelectronics.com/Attachment/GY-521/mpu6050.pdf. [Data uzyskania dostępu: 10 Grudzień 2021].

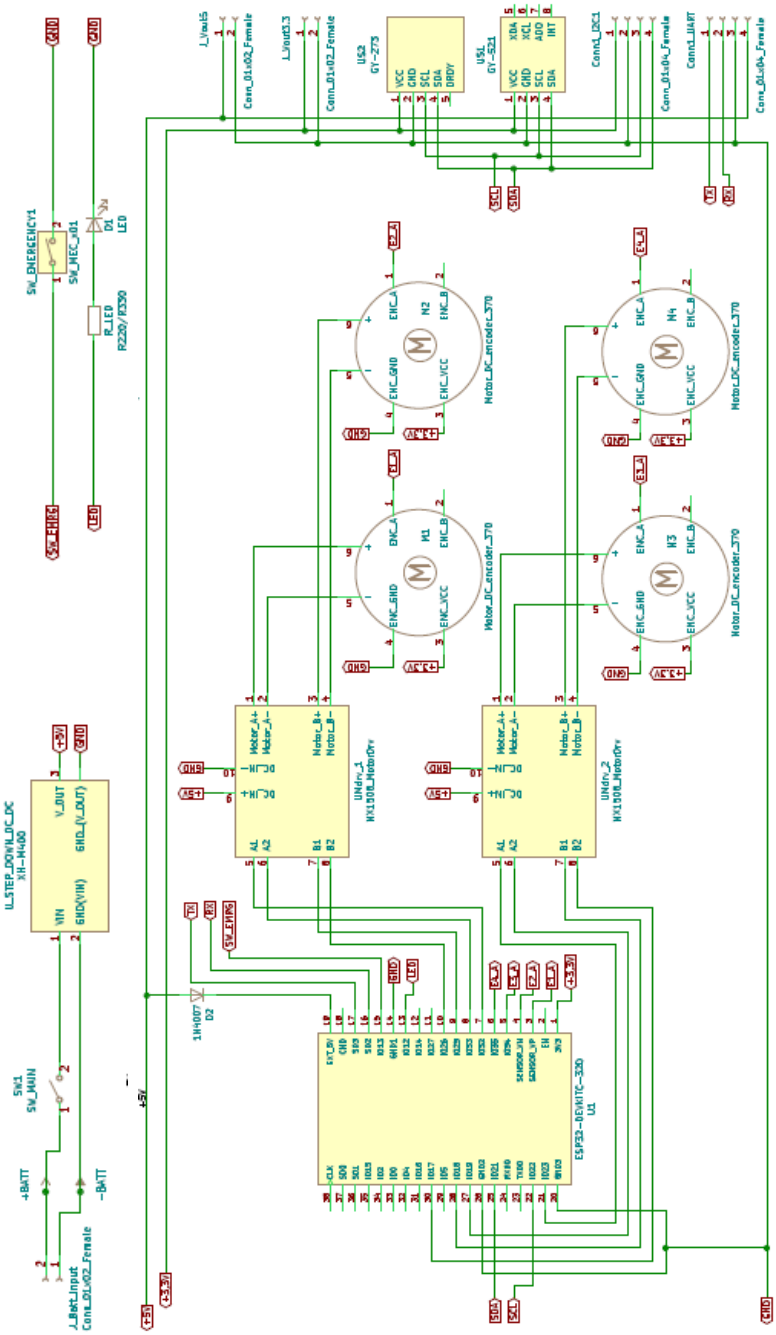
[12] Honeywell, „Adafruit - HMC5883L Magnetometer,” [Online]. Available: https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf. [Data uzyskania dostępu: 10 Grudzień 2021].

[13] R. Pełka, Mikrokontrolery architektura programowanie zastosowania, Warszawa: Wydawnictwa Komunikacji i Łączności sp. z o.o., 2000.

Dodatki

Dodatek A

Schemat elektryczny



Spis skrótów i symboli

ABS	Tworzywo sztuczne: akrylonitryl butadien styren
AC	Prąd przemienny (ang. *alternating current*)
ADC	Przetwornik analogowo-cyfrowy (ang. *analog to digital converter*)
BT	Bluetooth
CSV	Format pliku oddzielany przecinkami (ang. *Comma-separated values*)
DAC	Przetwornik cyfrowo-analogowy (ang. *digital to analog converter*)
DC	Prąd stały (ang. *direct current*)
FDM	Formowanie geometrii 3D w wysokiej temperaturze (ang. *fused deposition modelling)*
FPU	Jednostka zmiennoprzecinkowa (ang. *floating-point unit*)
GPIO	Wejście-wyjście ogólnego przeznaczenia (ang. *general-purpose input/output*)
IDE	Zintegrowane środowisko programistyczne (ang. *integrated development system*)
IMU	Czujnik nawigacji inercyjnej (ang. *intertial measurement unit*)
IP	Adres sieciowy urządzenia (ang. *Internet Protocol Address*)
I^2C	Szyna wymiany danych (ang. *inter-integrated cirquit*)
LiDAR	Czujnik laserowy do mierzenia odległości (ang. *Light Detection and Ranging*)
PID	Regulator proporcjonalno-całkująco-różniczkujący (ang. *proportional-integral-derivative)*
PLA	Tworzywo sztuczne: Poliaktyd (ang. *polyactic acid*)
PWM	Modulacja Szerokości Impulsów (ang. *pulse-width modulation*)
RPM	Prędkość obrotowa – obroty na minutę (ang. *revolutions per minute*)
SMD	Montaż niewielkich elementów elektronicznych (ang. *surface-mount devices*)
SPI	Szeregowy interfejs urządzeń peryferyjnych (ang. *serial peripheral interface*)

TCP Połączeniowy protokół komunikacyjny (ang. *Transmission Control Protocol/Internet Protocol*)

UART Uniwersalny asynchroniczny nadajnik-odbiornik (ang. *universal asynchronous receiver-transmitter*)

UDP Bezpołączeniowy protokół komunikacyjny (ang. *User Datagram Protocol*)

Wi-Fi Bezprzewodowa sieć komunikacyjna (ang. *wireless fidelity*)

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie, do pracy dołączono dodatkowe pliki zawierające:

- źródła programu – folder Robot_Program,
- filmy pokazujące działanie robota mobilnego,
- dokument w formacie PDF przedstawiający szczegółowo schemat elektryczny urządzenia

Spis rysunków

Spis tabel