



**A G H**

**AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE**

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,  
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI I ROBOTYKI

**Projekt dyplomowy**

*Opracowanie, konstrukcja oraz uruchomienie robota mobilnego,  
kołowego z napędem różnicowym*

*Development, construction and commissioning of mobile, wheeled  
robot with differential drive*

Autor:

*Adrian Moskwik*

Kierunek studiów:

*Automatyka i Robotyka*

Opiekun pracy:

*dr inż. Łukasz Więckowski*

Kraków, 2020



*Serdecznie dziękuję mojemu promotorowi dr inż.  
Łukaszowi Więckowskiemu za poświęcony czas i  
pomoc w realizacji tego projektu.*



# **Spis treści**

<b>1. Wstęp.....</b>	7
1.1. Cel i założenia pracy.....	7
1.2. Układ pracy.....	8
<b>2. Wprowadzenie teoretyczne .....</b>	9
2.1. Roboty mobilne .....	9
2.2. Napęd różnicowy .....	10
2.3. Robot Operating System.....	10
2.4. TurtleBot.....	11
<b>3. Konstrukcja robota.....</b>	13
3.1. Komunikacja podzespołów.....	15
3.2. Układ napędowy .....	17
3.2.1. Silniki prądu stałego.....	17
3.2.2. Koła .....	18
3.3. Układ sterujący .....	18
3.3.1. Układ mocy .....	18
3.3.2. Mikrokontroler .....	20
3.3.3. Sterownik główny i zdalny komputer .....	21
3.4. Układ sensoryczny.....	23
3.4.1. System wizyjny .....	23
3.4.2. Jednostka inercyjna .....	23
3.4.3. Lidar .....	24
3.5. Zasilanie .....	25
3.6. Złożenie konstrukcji .....	25
<b>4. Oprogramowanie sterownika silników.....</b>	27
4.1. Zadawanie i stabilizacja prędkości .....	27
4.1.1. Wartość zadana.....	28
4.1.2. Wartość pomiarowa.....	30

4.2. Odometria .....	31
4.3. Jednostka inercyjna.....	32
<b>5. Komunikacja między wątkowa w środowisku ROS .....</b>	<b>33</b>
5.1. Komunikacja jednostki zdalnej.....	33
5.1.1. Sterowanie w wykorzystaniem klawiatury .....	33
5.1.2. Sterowanie z wykorzystaniem joysticka.....	34
5.1.3. Sterowanie z wykorzystaniem środowiska MATLAB .....	34
5.2. Komunikacja jednostki pokładowej .....	35
5.2.1. Komunikacja systemu wizyjnego.....	35
5.2.2. Komunikacja Lidaru.....	35
5.2.3. Komunikacja sterownika silnika .....	35
<b>6. Testy.....</b>	<b>37</b>
6.1. Dobór nastaw regulatora PID .....	37
6.2. Pomiar pokonanej drogi wyznaczonej za pomocą enkoderów.....	40
6.3. Trajektoria ruchu robota na podstawie odometrii .....	41
6.4. Mapowanie pomieszczenia z wykorzystaniem lidaru .....	43
6.5. Test działania systemu wizyjnego .....	44
<b>7. Podsumowanie .....</b>	<b>47</b>

# **1. Wstęp**

We współczesnym świecie roboty w coraz większym stopniu towarzyszą człowiekowi. Jednym z przykładów na to zjawisko jest rozwój branży motoryzacyjnej. Na przestrzeni ostatnich lat można zaobserwować znaczący wzrost elementów elektronicznych w samochodach. Spowodowane jest to dążeniem do coraz większej autonomiczności pojazdów. Pozwala to nie tylko na zwiększenie komfortu jazdy, ale także pozytywnie wpływa na bezpieczeństwo, gdyż większość wypadków jest spowodowana czynnikiem ludzkim. Idealnym rozwiązaniem byłoby całkowite wyeliminowanie wpływu człowieka podczas jazdy. Niestety nie jest to jeszcze możliwe, ale rozwój branży motoryzacyjnej zmierza w dobrym kierunku.

Roboty mobilne, jakimi są autonomiczne pojazdy, nie są konstruowane wyłącznie w celach komercyjnych. Cieszą się one także dużą popularnością wśród konstruktorów-amatorów, którzy bardzo chętnie projektują i budują takie platformy dydaktyczne. Proces realizacji takiego projektu jest bowiem idealną okazją do poszerzenia swojej wiedzy w tej dziedzinie.

## **1.1. Cel i założenia pracy**

Celem projektu dyplomowego jest zaprojektowanie, zbudowanie oraz uruchomienie platformy dydaktycznej robota mobilnego, kołowego o napędzie elektrycznym, różnicowym. Napędy robota zostaną zrealizowane za pomocą czterech silników DC, wraz z dedykowanym układem mocy Pololu VNH5019, posiadającym własny mikro kontroler oraz jednostkę inercyjną. Robot mobilny zostanie wyposażony w sterownik głównego typu: Intel Up-Board, wraz z dedykowanym systemem wizyjnym Intel RealSense R200. Sterownik robota zostanie oprogramowany w języku C/C++, będzie on odpowiedzialny za sterowanie napędami, czyli zadawanie oraz stabilizację prędkości, a także wyliczanie aktualnej pozycji oraz parametrów ruchu poprzez odometrię wyznaczoną na podstawie odczytów z enkoderów. Dodatkowo robot zostanie wyposażony w lidar, dzięki czemu będzie zdolny do mapowania otoczenia. Informacje z platformy zostaną udostępnione dla systemu nadziednego, pracującego pod kontrolą systemu operacyjnego, wykorzystującego komunikację między wątkową ROS. Testy działania robota mobilnego zostaną przeprowadzone z wykorzystaniem dedykowanych w robotyce mobilnej narzędzi programowych tj: ROS/RViz, MATLAB Robotics System Toolbox.

## 1.2. Układ pracy

Projekt dyplomowy składa się z siedmiu rozdziałów. Rozdział pierwszy został poświęcony wprowadzeniu w tematykę robotyki oraz określeniu celów przed rozpoczęciem pracy nad projektem. W rozdziale drugim została przybliżona tematyka robotów mobilnych, komunikacja między wątkowa oraz platforma dydaktyczna Tutrlebot. W rozdziale trzecim została przedstawiona konstrukcja robota oraz charakterystyka podzespołów wykorzystanych do jego zbudowania. Rozdział czwarty został poświęcony oprogramowaniu sterownika robota. W rozdziale piątym zostało opisane zastosowanie komunikacji między wątkowej. Rozdział szósty jest poświęcony testom działania robota mobilnego. W rozdziale siódmym zostało zawarte podsumowanie z przebiegu pracy nad projektem dyplomowym.

## **2. Wprowadzenie teoretyczne**

### **2.1. Roboty mobilne**

Robot mobilny to autonomiczny pojazd zdolny do poruszania się po swoim środowisku, bez fizycznego przywiązywania do jednej lokalizacji [1]. Oznacza to, że ich przestrzeń robocza jest konstrukcyjnie nieograniczona. Nie posiada ograniczeń takich jak np. przewodowy sterujące podłączone stacjonarnie do jednego miejsca. Dodatkowo nie wymagają one bezpośredniej ingerencji człowieka. Sprawia to, że ich konstrukcja mechaniczna może przyjmować dowolne rozmiary potrzebne do wykonywanych zadań, bez ograniczeń związanych z czynnikiem ludzkim.

Środowiskiem robota mobilnego może być ląd, woda lub powietrze. W oparciu o środowisko poruszania się, roboty mobilne można podzielić na:

- *Jeżdżące* (np. *TurtleBot3* [2], *Handle* [3], *R2-D2* [4])
- *Kroczące* (np. *Atlas* [5], *Spot* [6], *Cheetah* [7])
- *Pływające* (np. *UMIS* [8], *Bluefin-12* [9])
- *Latające* (np. *MQ-1C* [10], *DJI Inspire 2* [11])
- *Inne* (np. *Abigaille* [12])

Najbardziej popularną grupą są roboty jeżdżące. Grupa ta rozwija się w szybkim tempie, jest to spowodowane między innymi inwestycjami globalnych koncernów motoryzacyjnych. Skupiają się one na rozwoju technologicznym i dążą do coraz większej autonomiczności swoich pojazdów. Prekursorem w tej dziedzinie jest firma Tesla, która zapowiedziała, że w połowie 2020 roku, w ich pojazdach zostanie wprowadzony tryb pełnej autonomicznej jazdy.

Ze względu na dynamiczny rozwój tej grupy robotów mobilnych, są one też popularne pośród konstruktorów-amatorów, którzy zajmują się budową robotów w celach dydaktycznych, badawczych lub hobbystycznych.

## 2.2. Napęd różnicowy

Napęd różnicowy umożliwia kołom na jednej osi uzyskanie różnych prędkości obrotowych, jest to niezbędne w przypadku robotów o napędzie kołowym bez skrętnej osi. Podczas ruchu po łuku, koła pojazdu pokonują inną drogę. Zastosowanie napędu różnicowego pozwala, aby toczyły się one po swoich torach ruchu bez poślizgu, osiągając różne prędkości obrotowe.

## 2.3. Robot Operating System

ROS (*ang. Robot Operating System*) jest to meta-operacyjny systemem dla robotów, który posiada otwarty dostęp do kodu źródłowego. Podobnie jak zwyczajny system operacyjny zapewnia warstwę abstrakcji sprzętowej, kontrolę urządzeń niskopoziomowych oraz implementacje powszechnie używanych funkcjonalności [13]. ROS różni się jednak od powszechnego systemu możliwością komunikacji między wątkowej. Procesy mogą komunikować się ze sobą w czasie swojego wykonywania. Sprawia to, że ROS jest dobrym systemem do wsparcia sterowania robotem oraz czujnikami (Rys.2.1). Kolejną zaletą systemu ROS jest możliwość współpracy z większością popularnych systemów operacyjnych, takich jak Windows, Linux, Mac, Android lub iOS.

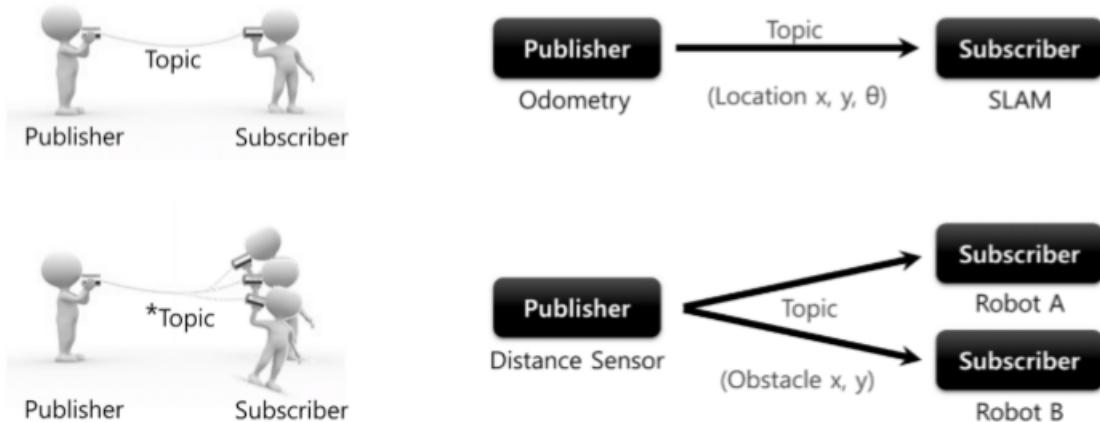


Rys. 2.1. ROS jako meta-operacyjny system [13].

Główne założenia systemu ROS można przedstawić jako [14]:

- *Komunikacja P2P (peer to peer)* - model komunikacji w sieci komputerowej zapewniający wszystkim hostom te same uprawnienia [15].

- *Oparty na narzędziach* - duża liczba małych narzędzi jest używana do budowania i uruchamiania różnych komponentów systemu.
- *Wielojęzykowy* - wspiera wiele języków programowania.
- *Niezagnieżdżony* - tworzenie tylko małych plików wykonywalnych, które ujawniają bibliotekę funkcjonalności do systemu ROS. Pozwala to na łatwiejszą ekstrakcję kodu i użycie go ponownie poza pierwotnym przeznaczeniem.
- *Darmowy i posiadający otwarty dostęp do kodu źródłowego*.



Rys. 2.2. Komunikacja z wykorzystaniem tematów [13].

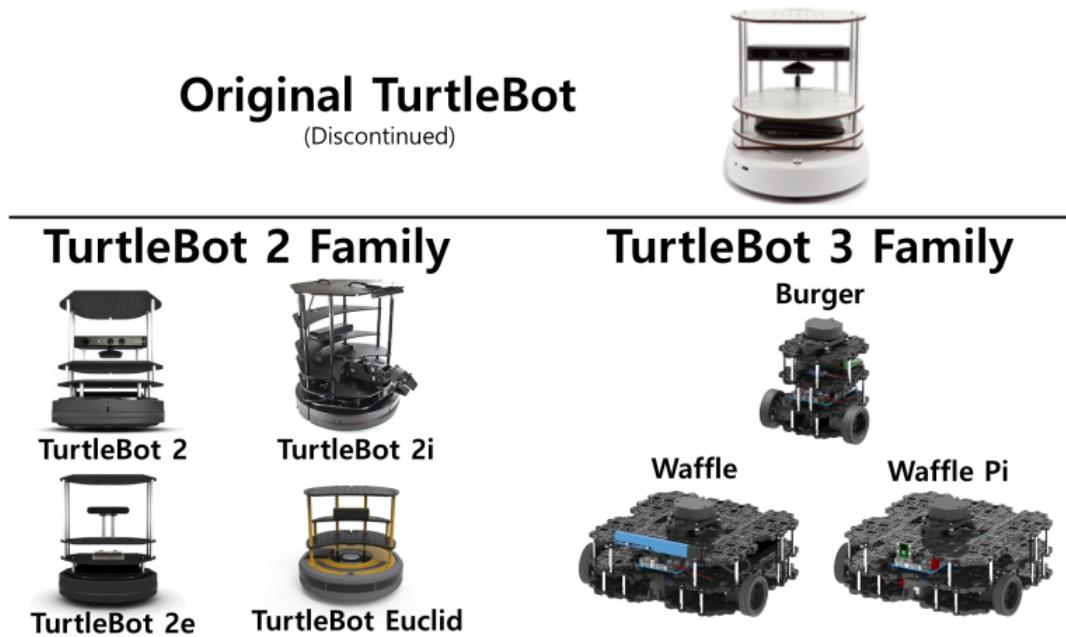
Komunikacja odbywa się między *węzłami* (ang. *nodes*), czyli pojedynczymi procesami działającymi w ramach systemu ROS. Kanał komunikacyjny między węzłami to *temat* (ang. *topic*). *Wiadomość* (ang. *message*) to dane wysypane między węzłami za pomocą tematu. Węzeł publikuje lub subskrybuje temat, czyli wysyła lub pobiera z niego wiadomość. Schemat komunikacji przy pomocy tematów został przedstawiony na Rys.2.2.

## 2.4. TurtleBot

Turtlebot jest standardowym robotem platformowym wykorzystującym komunikację między wątkową ROS [16]. Platforma ta jest bardzo popularna wśród deweloperów, a także studentów, ponieważ doskonale nadaje się do celów dydaktycznych. Oprogramowanie posiada otwarty dostęp do kodu źródłowego. Upraszczca to zrozumienie zasady działania platformy, ponieważ umożliwia łatwy dostęp do dokumentacji, w której znajduje się opis działania poszczególnych modułów.

Istnieją trzy wersje robotów serii TurtleBot (Rys.2.3). Najnowsza z nich została zaprezentowana w maju 2017 roku. TurtleBot3 jest małym, niskobudżetowym, programowalnym, opartym na komunikacji między wątkowej ROS robotem mobilnym, który służy do celów edukacyjnych, badawczych lub

hobbistycznych. Składa się on z wewnętrznego komputera, mikro kontrolera, systemu wizyjnego i czujników. Wewnętrzny komputer komunikuje się przy pomocy komunikacji ROS, przez wi-fi ze zdalnym komputerem. Udostępnia on dane z czujników i otrzymuje sterowanie, czyli prędkość liniową i kątową, która jest następnie przeliczana na wartość prędkości obrotowej każdego z kół. Mikro kontroler komunikuje się z komputerem wewnętrznym przy pomocy szeregowej komunikacji między wątkowej.

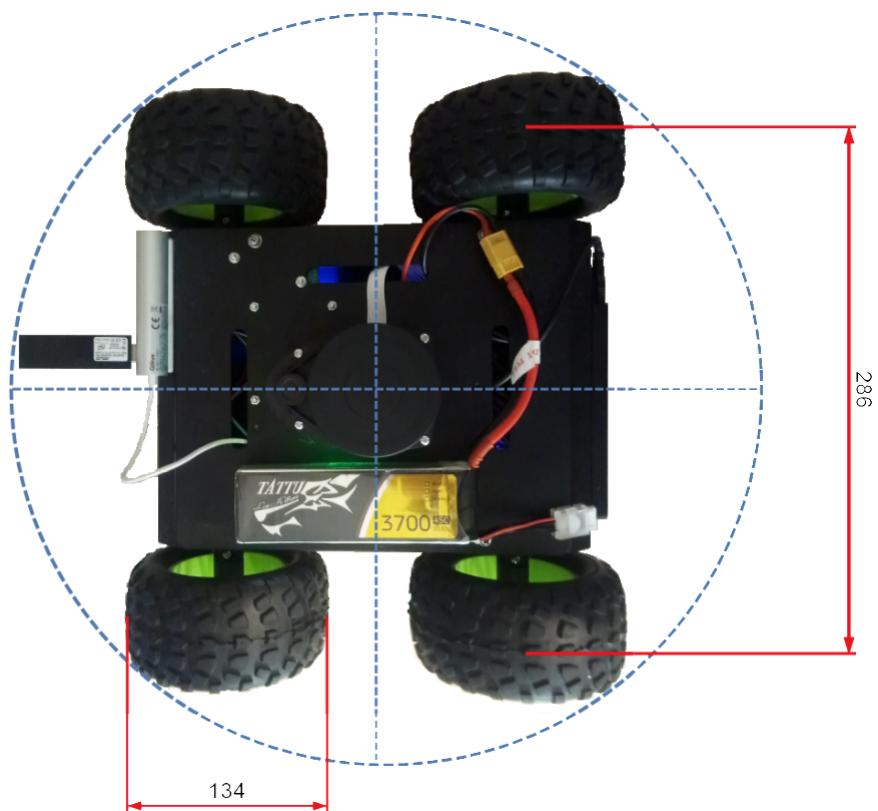


Rys. 2.3. Rodzina robotów TurtleBot [16].

### 3. Konstrukcja robota

Do konstrukcji robota wykorzystany został zestaw *4WD Outdoor Mobile Platform*. Składa się on z metalowej obudowy oraz czterech silników i kół [17]. Posiada poziomy charakter budowy, jednakże konstruowany robot składa się tylko z dolnego poziomu.

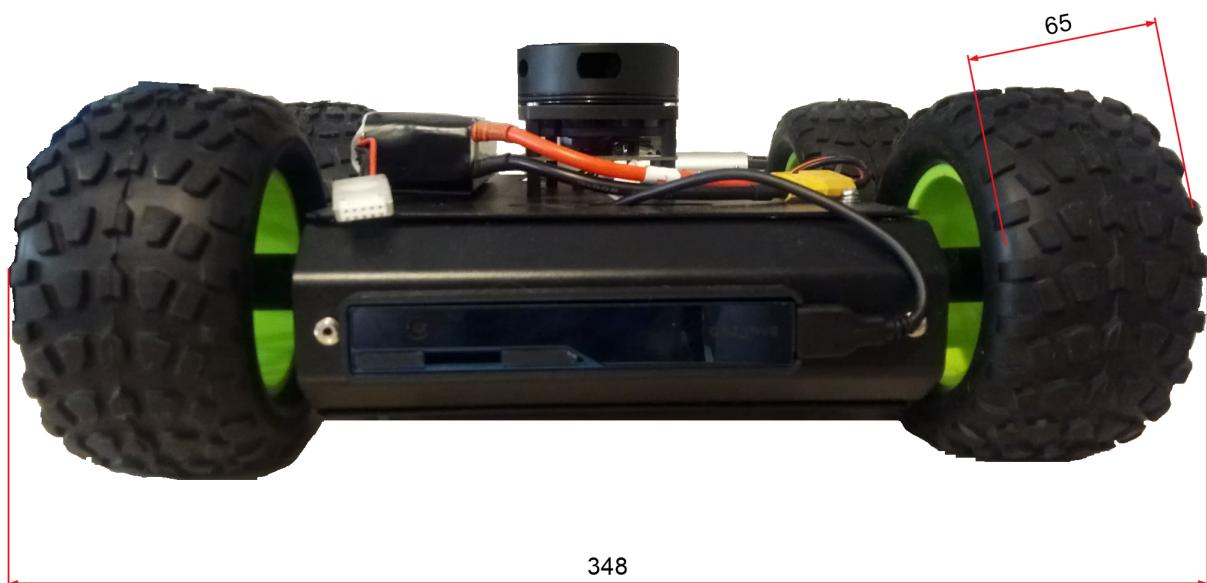
Głównym założeniem mechanicznym jest stworzenie jeżdżącego robota mobilnego poruszającego się po środowisku lądowym. Ruch robota jest możliwy zarówno po płaskiej i nierównej powierzchni. Podzespoły robota można podzielić na układ napędowy, układ sterujący, układ sensoryczny oraz zasilanie. Specyfikacja techniczna robota została przedstawiona w Tabeli 3.1. Wymiary platformy zostały zilustrowane na Rys. 3.1 - 3.3.



Rys. 3.1. Widok robota z góry (opracowanie własne).



Rys. 3.2. Widok robota z boku (opracowanie własne).



Rys. 3.3. Widok robota z przodu (opracowanie własne).

**Tabela 3.1.** Specyfikacja sprzętowa (opracowanie własne).

Maksymalna prędkość	$1.94 \frac{m}{s}$
Maksymalna prędkość obrotowa	$13.57 \frac{rad}{s}$
Wymiary (dł. x szer. x wys.)	356mm x 348mm x 152mm
Waga	3.68 kg
Główny komputer pokładowy	Intel UP Board
Mikro kontroler	STM32L476 Nucleo-64
Czujnik laserowy	RPLIDAR-A1
Kamera	Intel Realsense R200
IMU	Żyroskop 3-osiowy Akcelerometr 3-osiowy Magnetometr 3-osiowy
Porty we/wy	GPIO 51piny
Porty mikro kontrolera	USB OTG 2.0 x1, SAI x2, I2C x3, USART x5, SPI x3, CAN x1
Połączenie z komputerem	USB
Bateria	Litowo-polimerowa 14.8V 3700mAh

### 3.1. Komunikacja podzespołów

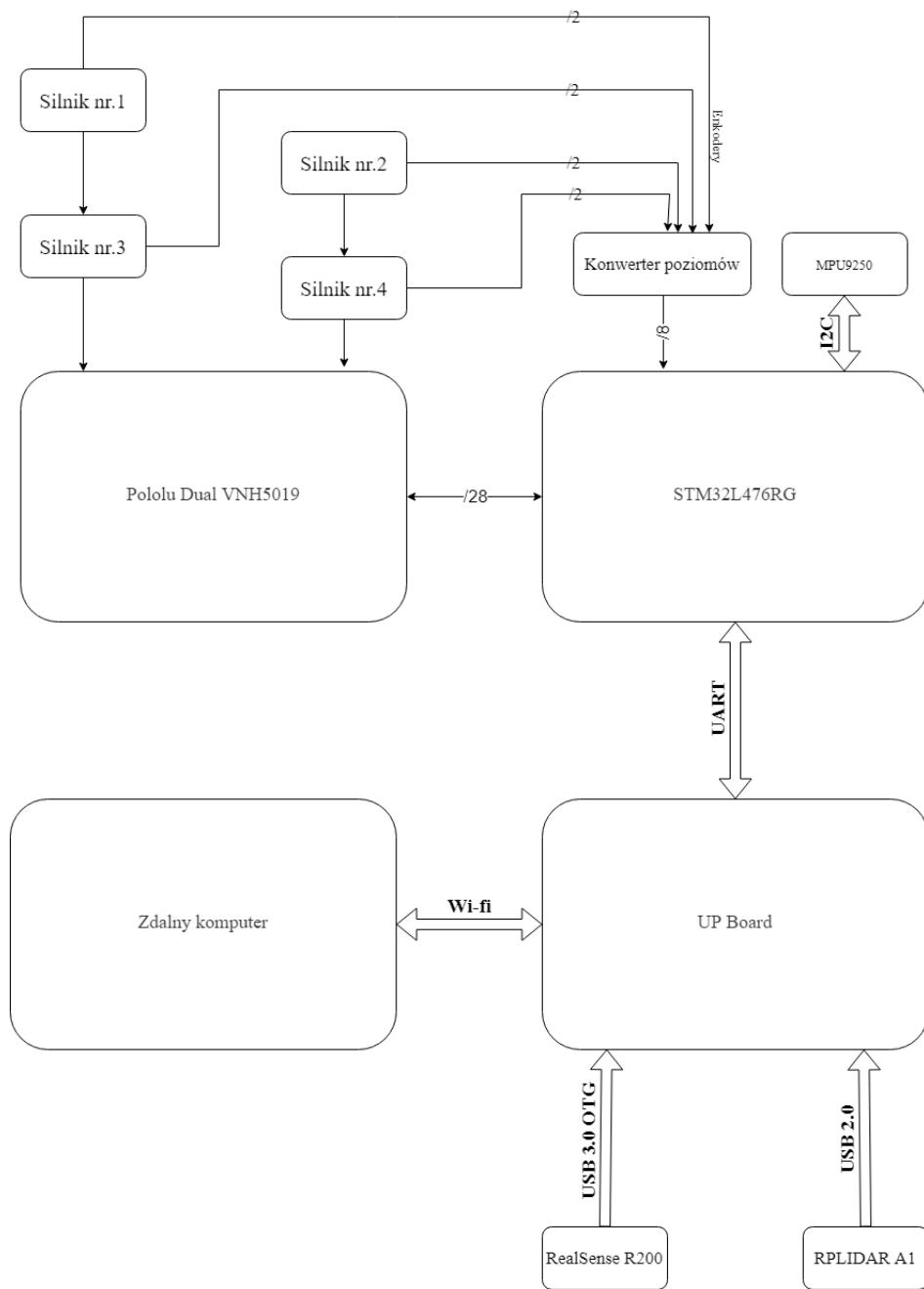
Sposób komunikacji podzespołów został przedstawiony na Rys. 3.4.

Jednostka mocy Pololu Dual VNH5019 jest dwukanałowa. Oznacza to, że możliwe jest sterowanie prędkością po dwóch stronach robota, a nie na każdym kole. Silniki po tej samej stronie zostały połączone szeregowo, a następnie trwale połączone z jednostką mocy, która z kolei została połączona z mikro kontrolerem.

Napięcie znamionowe dla pinów STM32L476RG wynosi od 1.71 do 3.6V, natomiast napięcie sygnałów A i B pochodzących od enkoderów przyjmuje wartość 5V. W tym celu został wykorzystany konwerter poziomów, który zmniejszył ich napięcie do 3.3V, poziomu bezpiecznego dla użytego mikro kontrolera.

Z mikro kontrolerem łączy się dodatkowo, poprzez komunikację I2C, jednostka inercyjna MPU9250. STM32 komunikuje się poprzez port szeregowy z komputerem pokładowym. Up Board otrzymuje ponadto dane z dedykowanego systemu wizyjnego, połączonego poprzez USB 3.0 OTG i lidaru połączonego poprzez USB 2.0. Dodatkowo na zewnątrz urządzenia został wyprowadzony Qilive Hub z 4 portami USB 2.0. Pozwali to na rozbudowanie robota o kolejne moduły bez konieczności ingerencji elektronikę znajdującą się wewnątrz obudowy.

Główny sterownik wymienia informacje ze zdalnym komputerem poprzez wi-fi. Wykorzystany do tego został zdalny moduł wi-fi ASUS WL-167g, jest on połączony z głównym sterownikiem poprzez USB 2.0. Moduł został wyprowadzony na zewnątrz robota w celu uzyskania lepszej jakości sygnału.



Rys. 3.4. Schemat komunikacji podzespołów (opracowanie własne).

**Tabela 3.2.** Parametry silnika [18].

Specyfikacja	
Napięcie zasilania	1 - 12V
Natężenie prądu bez obciążenia	350mA
Prędkość obrotowa bez obciążenia	251 + 10% RPM
Napięcie znamionowe enkodera	5V
Typ enkodera	Hall
Waga	205g

## 3.2. Układ napędowy

Elementami wykonawczymi robota są części napędowe, czyli silniki DC. Nadają one kołom prędkość obrotową zadaną sterowaniem otrzymanym z układu sterującego. Wykorzystane silniki są dedykowanymi elementami do wykorzystanej obudowy robota.

### 3.2.1. Silniki prądu stałego

Napęd robota stanowią 4 silniki prądu stałego, zintegrowane z przekładnią zębata i połączone z enkoderami kwadratowymi (Rys. 3.5). Przełożenie przekładni wynosi 43,8:1. Enkoder kwadratowy zapewnia rozdzielcość 16 pomiarów na obrót wału silnika, co odpowiada 700 pomiarom na obrót wału wyjściowego. Oznacza to, że impuls będzie występował co 0.6mm przebytej drogi. Najważniejsze dane techniczne silnika zostały przedstawione w tabeli 3.2.

**Rys. 3.5.** Silnik DC serii Gear Motor [18].

### 3.2.2. Koła

Do poruszania się robota zostały wykorzystane koła o średnicy 134mm i szerokości 65mm z gumową oponą terenową (Rys. 3.6). Szeroka opona zapewnia większą powierzchnię styku z podłożem. Skutkuje to uzyskaniem lepszej przyczepności oraz skróceniem drogi hamowania w stosunku do węższej opony.



Rys. 3.6. Koła robota [17].

Dobrane silniki i koła pozwalają robotowi mobilnemu na uzyskanie maksymalnej prędkości równej ok.  $1.94 \frac{m}{s}$ . Osiągana prędkość jest wystarczająca do planowanych zastosowań robot.

## 3.3. Układ sterujący

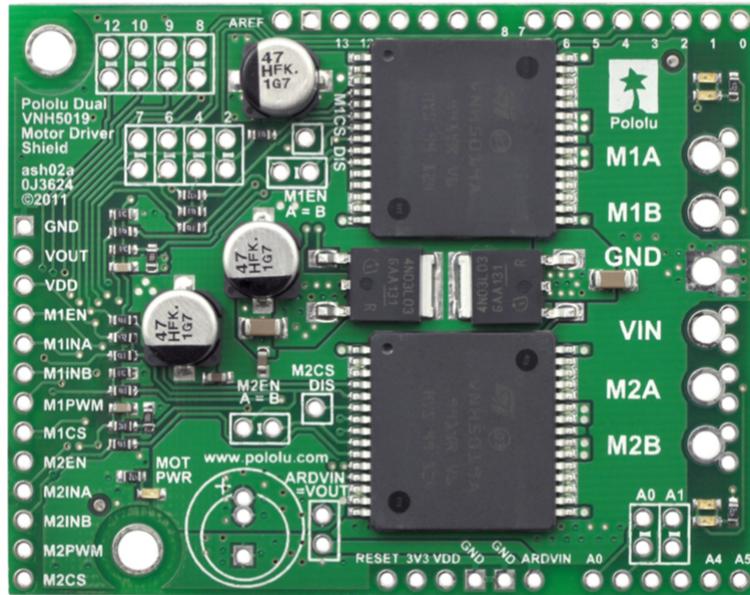
Układ sterujący ma na celu wyznaczenie i zadanie sterowania elementom wykonawczym. W tym przypadku będzie to zadawanie prędkości obrotowej na koła, poprzez zmianę wartości napięcia zasilania podawanego na silniki. Wykorzystany w tym celu został dwukanałowy układ mocy, mikro kontroler oraz nadzędny sterownik główny. Na zdalnym komputerze jest wyliczana prędkość liniowa i kątowa, którą ma osiągnąć robot. Informację tę pobiera mikro kontroler i na jej podstawie przelicza prędkość z jaką mają się poruszać prawe i lewe koła. Następnie przy pomocy układu mocy odpowiednie napięcie zasilania jest podawane na silniki.

### 3.3.1. Układ mocy

Do sterowania napędami zastosowany został układ mocy *Pololu Dual VNH5019 Motor Driver Shield* (Rys.3.7). Układ ten jest kompatybilny z platformą Arduino. Posiada własną bibliotekę, która pozwala

w bardzo prostu sposobie sterować silnikami prądu stałego. Zasilany jest napięciem z zakresu 5.5 - 24V o ciągłym poborze prądu do 12A [19]. Posiada dwa kanały do sterowania silnikami, do każdego kanału zostały wpięte dwa silniki znajdujące się po tej samej stronie, połączone w sposób równoległy.

Układ umożliwia zmianę kierunku obrotów silnika oraz regulację prędkości obrotowej poprzez sygnał PWM. Działanie poszczególnych pinów zostało przedstawione w Tabeli 3.3.



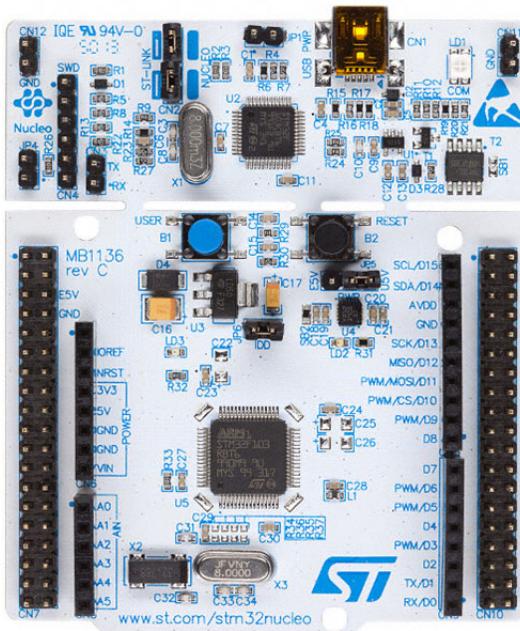
Rys. 3.7. Pololu Dual VNH5019 Motor Driver Shield [19].

Tabela 3.3. Piny sterownika silników [20].

VNH5019 Pin	Zastosowanie
M1INA	Kierunek silnika 1 - wejście A
M1INB	Kierunek silnika 1 - wejście B
M1EN/DIAG	Silnik 1 – informacja o błędzie
M2INA	Kierunek silnika 2 – wejście A
M2INB	Kierunek silnika 2 – wejście B
M1PWM	Sterowanie prędkością silnika 1
M2PWM	Sterowanie prędkością silnika 2
M2EN/DIAG	Silnik 2 – informacja o błędzie
M1CS	Odczyt prądu na silniku 1
M2CS	Odczyt prądu na silniku 2

### 3.3.2. Mikrokontroler

Do sterowania robotem wykorzystany został mikro kontroler *STM32 Nucleo-L476RG* (Rys.3.8) z wydajnym rdzeniem Arm® Cortex®-M4 32-bit RISC, taktującym do 80MHz. Wyposażony jest w 1MB pamięci flash oraz 128KB pamięci SRAM. Obsługuje interfejsy komunikacyjne takie jak SPI, I2C, USART, UART, USB, CAN. Posiada wbudowany programator i debugger ST-LINK/V2-1. Pozwala na pracę w trybie kompatybilności z Arduino Uno. Umożliwia to wykorzystanie do jego programowania środowisk dedykowanych językowi Arduino. Oferuje obsługę szesnastu liczników róznorakiego zastosowania. Posiada 51pinów typu GPIO (*general-purpose input/output*), a także 16 linii przerwań zewnętrznych. Źródłem zasilania jest przez port USB z napięciem między 3.0-3.6V [21].



Rys. 3.8. Mikrokontroler STM32 Nucleo-L476RG [22].

Zastosowany mikro kontroler pozwolił na integrację z jednostką mocy, ponieważ jego budowa umożliwia pracę w trybie kompatybilnym z Arduino Uno. Wykorzystany STM pozwala więc na wykorzystanie bibliotek dostępnych dla Arduino i dodatkowo oferuje większą pamięć podręczną oraz moc obliczeniową. Znaczącą zaletą jest także duża liczba pinów GPIO. Zostały one wykorzystane do uzyskania komunikacji z enkoderami, jednostką inercyjną, a także jednostką mocy.

W dokumentacji technicznej mikro kontrolera zostały odnalezione piny odpowiedzialne za poszczególne funkcje, a następnie stworzono schemat wszystkich potrzebnych połączeń (Tabela. 3.4) [22]. Połączenia związane z enkoderami i jednostką inercyjną zostały trwale wykonane na płytce prototypowej do lutowania, natomiast piny związane z jednostką mocy posiadają możliwośćwyjęcia.

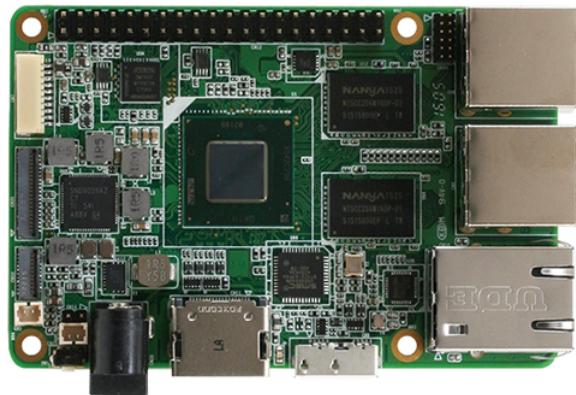
**Tabela 3.4.** Połączenia pinów STM32 Nucleo-L476RG (opracowanie własne).

<b>STM32 Pin</b>	<b>Zastosowanie</b>
PA10	VNH5019 - M1INA
PB5	VNH5019 - M1INB
PB10	VNH5019 - M1EN/DIAG
PA8	VNH5019 - M2INA
PA9	VNH5019 - M2INB
PC7	VNH5019 - M1PWM
PB6	VNH5019 - M2PWM
PA6	VNH5019 - M2EN/DIAG
PA0	VNH5019 - M1CS
PA1	VNH5019 - M2CS
PB13	MPU9250 - SCL
PB14	MPU9250 - SDA
PC8	Enkoder 1 - Sygnał A
PC6	Enkoder 1 - Sygnał B
PC5	Enkoder 2 - Sygnał A
PA12	Enkoder 2 - Sygnał B
PB2	Enkoder 3 - Sygnał A
PB1	Enkoder 3 - Sygnał B
PB15	Enkoder 4 - Sygnał A
PC4	Enkoder 4 - Sygnał B

### 3.3.3. Sterownik główny i zdalny komputer

Jednostką nadzcznąą, którą wykorzystuje robot jest minikomputer *UP Board* (Rys.3.9) [23]. Główne parametry platformy zostały przedstawione w Tabeli 3.5. Wykorzystywany system operacyjny to Ubuntu 16.04.6 LTS. System ten jest kompatybilny z pakietem ROS Kinetic Kame, w oparciu o niego odbywa się komunikacja między wszystkimi modułami. Zdalny komputer także posiada zainstalowaną tę samą wersję systemu oraz pakietu ROS, umożliwia to zastosowanie komunikacji między wątkowej między nimi.

Platforma posiada jeden port USB 3.0 OGT, przez który jest połączona ze swoim dedykowanym systemem wizyjnym. Liczba dostępnych portów USB 2.0 została zwiększena do siedmiu, poprzez wykorzystanie Qilive Hub. Przy ich pomocy odbywa się komunikacja z lidarem, a także mikro kontrolerem. Moduł wi-fi został wyprowadzony na zewnątrz robota, aby zwiększyć siłę sygnału, ponieważ od niej zależy szybkość przesyłu danych do zdalnego komputera.



Rys. 3.9. Minikomputer Up Board [23].

Na platformie został zainstalowany także program RealVNC w trybie serwera. Pozwala on na zdalny dostęp do minikomputera przez wi-fi ze zdalnego komputera, na którym jest zainstalowany pakiet RealVNC w trybie obserwatora. RealVNC na minikomputerze jest uruchamiany przy starcie systemu, dzięki temu zdalny komputer posiada dostęp do *pulpitu* od momentu uruchomienia całego robota. Prędkość transmisji zdalnego obrazu zależy od prędkości sieci na komputerze pokładowym, sprawia to problemy przy słabej jakości połączenia.

Zasilanie odbywa się z wykorzystaniem przetwornicy napięcia, podłączonej na stałe do wyprowadzeń na płytce. Pozwoliło to na zasilanie platformy z tego samego źródła co napędy. Up Board nie posiada przełącznika odpowiadającego za uruchamianie. Sprawia to, że w momencie podłączenia zasilania platforma uruchamia się samoistnie.

**Tabela 3.5.** Specyfikacja minikomputera Up Board [23].

Specyfikacja	
Procesor	Intel® Atom™ x5-Z8350 Processor (2M Cache, do 1.92 GHz) 64-bitowy, 4-rdzeniowy
Pamięć RAM	1GB / 2GB / 4GB DDR3L-1600
Dysk eMMC	16GB / 32 GB / 64 GB
Grafika	Intel® HD 400 Graphics
Złącza	4x gniazdo USB 2.0 2x wyprowadzenie USB 2.0 1x gniazdo USB 3.0 OTG 1x gniazdo HDMI 1.4a
Wymiary	85.60mm x 56.5mm
Zasilanie	5V DC-in

**Tabela 3.6.** Parametry kamery [24].

Specyfikacja	
Zakres pracy	0.5m - 3.5m
Rozdzielcość głębokości	480 x 360
Szybkość w kl/s	60fps
Głębia ostrości i pola widzenia	H: 59, V: 46, D: 70
Typ interfejsu systemu	USB 3.0
Wymiary	101.6mm x 9.6mm x 3.8mm

## 3.4. Układ sensoryczny

Układ sensoryczny służy do mierzenia wybranych wartości fizycznych. Pozwala on na uzyskanie informacji o zachowaniu systemu pod wpływem zadanego sterowania. W budowie robota zostały wykorzystane: system wizyjny, jednostka inercyjna oraz lidar. Wysyłają one informacje na temat otoczenia, a także prędkości i przyśpieszenia robota.

### 3.4.1. System wizyjny

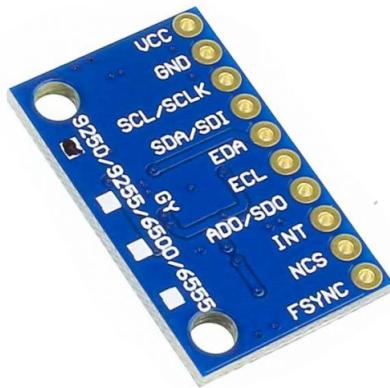
Wykorzystanym systemem wizyjnym jest dedykowana dla Up Board kamera *Intel RealSense R200* (Rys. 3.10). Najważniejsze dane techniczne zostały przedstawione w Tabeli 3.6 [24]. Dzięki interfejsowi USB 3.0 OTG prędkość przesyłu danych jest wystarczająca na uzyskanie obrazu otoczenia w czasie rzeczywistym. Posiada także bibliotekę umożliwiającą jej komunikację między wątkową ROS.

**Rys. 3.10.** RealSense R200 [24].

### 3.4.2. Jednostka inercyjna

Wykorzystana jednostka inercyjna to *MPU-9250* (Rys. 3.11), jest to urządzenie 9-osiowe, które wyposażone jest w 3-osiowy żyroskop, 3-osiowy akcelerometr i 3-osiowy magnetometr. Posiada także czujnik temperatury. Umożliwia komunikację magistralami I2C oraz SPI [25]. W tworzonym robocie mobilnym zastosowana została komunikacja I2C. Żyroskop informuje o prędkości kątowej robota, pozwala on na ustalenie orientacji. Akcelerometr podaje informacje na temat wypadkowej siły działającej na platformę. W przypadku braku zewnętrznych sił robot jest poddany działaniu tylko siły grawitacji,

akcelerometr pozwala więc na określenie pozycji platformy względem podłożu. Magnetometr dostarcza informacji na temat ziemskiego pola magnetycznego. MPU-9250 posiada własną bibliotekę Arduino ułatwiającą odczyt danych z czujników.



Rys. 3.11. Jednostka inercyjna MPU-9250 [26].

### 3.4.3. Lidar

Lidar (*Light Detection and Ranging*) jest skanerem laserowym, który pozwala na detekcję elementów otoczenia i poznania odległości od nich. Umożliwia to wykonanie mapy środowiska, które jest skanowane przez laser.

W robocie mobilnym został zamontowany *RPLIDAR A1* (Rys. 3.12). Lidar ten posiada możliwość obrotu o 360 stopni, zasięg do 12m i częstotliwości do 10Hz. Komunikacja następuje przez interfejs UART, połączony z jednostką nadzcznąą przez port USB 2.0 [27]. Wyposażony jest w dedykowaną bibliotekę do komunikacji między wątkowej ROS.



Rys. 3.12. Skaner laserowy RPLIDAR A1 [27].

### 3.5. Zasilanie

Zasilanie konieczne jest dla silników napędowych i platformy Up Board. W tym celu został wykorzystany akumulator litowo-polimerowy *Gens Ace TATTU 3700mAh 14.8V* (Rys. 3.12). Jego prąd rozładowania wynosi 45C [28]. Umieszczony jest na zewnątrz robota w celu wygodnego dostępu do ładowania. Połączony jest z układem mocy VNH5019 oraz przetwornicą napięcia w celu obniżenia do 5V, aby możliwe było zasilenie minikomputera Up Board.



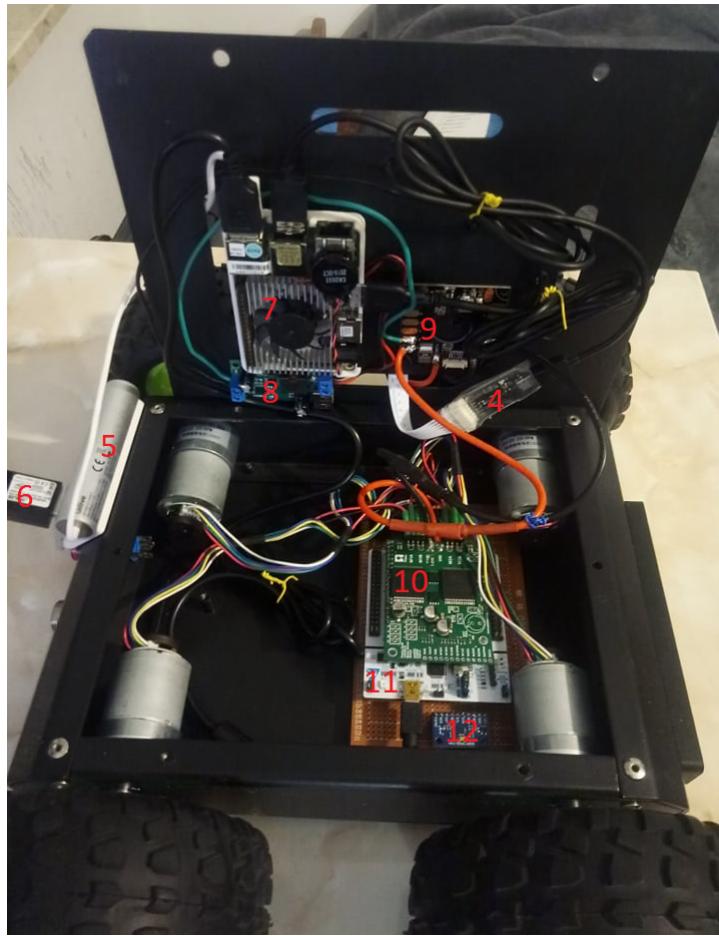
Rys. 3.13. Akumulator z serii TATTU [28].

### 3.6. Złożenie konstrukcji

Platforma została złożona na podstawie stworzonego schematu elektrycznego (Rys. 3.4). Większość połączeń i elementów została umieszczona wewnątrz obudowy robota. Złożony robot został przedstawiony na Rys. 3.14, natomiast połączenia i elementy znajdujące się w środku robota zobrazowano na Rys. 3.15.



Rys. 3.14. Złożony robot mobilny (opracowanie własne).



Rys. 3.15. Wnętrze zbudowanego robota mobilnego (opracowanie własne).

gdzie

- 1 – Kamera RealSense R200.
- 2 – Akumulator 14.8V 3700mAh.
- 3 – Połączenia między akumulatorem i rozdzielaczem napięciowym.
- 4 – Lidar RPLIDAR A1
- 5 – Qilive UBS Hub.
- 6 – Moduł wi-fi.
- 7 – Komputer pokładowy Intel UP Board.
- 8 – Przetwornica napięciowa.
- 9 – Rozdzielacz napięciowy.
- 10 – Jednostka mocy Pololu Dual VNH5019 Motor Driver Shield.
- 11 – Mikro kontroler STM32I476RG Nucleo-64.
- 12 – Jednostka inercyjna MPU9250.

## 4. Oprogramowanie sterownika silników

Sterowanie silnikami zrealizowano za pomocą układu mocy VNH5019 połączonego z STM32 Nucleo-L476RG. Oprogramowanie silników zostało stworzone w języku C/C++. Wiele modułów wykorzystywanych do tworzenia robota mobilnego posiada gotowe biblioteki Arduino. Mikro kontroler jest kompatybilny z platformą Arduino Uno, pozwala to na uruchomienie programu Arduino na wykorzystanej platformie STM32.

Do tworzenia oprogramowania użyte zostało środowisko Sloeber, które jest nakładką na zintegrowane środowisko programistyczne Eclipse. Główną zaletą Sloebera jest możliwość tworzenia oprogramowania dla platform Arduino. Środowisko rekomendowane przez producenta, Arduino IDE, jest problematyczne w obsłudze w przypadku większych projektów, dlatego do tworzenia oprogramowania został użyty Sloeber.

Stworzony program, składa się z dwóch głównych funkcji, które są standardem w przypadku programów Arduino. Pierwsza z nich *setup()* odpowiada za inicjalizacje wszystkich wykorzystanych modułów. Drugą z nich jest funkcja *loop()*, która jest pętlą wykonującą się nieprzerwanie od uruchomienia programu. W niej znajduje się główny program odpowiedzialny za sterowanie i wymianę informacji z jednostką nadzorzącą.

### 4.1. Zadawanie i stabilizacja prędkości

Sterowanie silnikami odbywa się z wykorzystaniem biblioteki od producenta jednostki mocy VNH5019 *DualVNH5019MotorShield.h* [29]. Pozwala ona na sterowanie prędkością kół poprzez zadawanie wartości sygnału PWM (ang. *Pulse-Width Modulation*).

W celu stabilizacji prędkości zastosowane zostały dwa regulatory PID, po jednym na kanał jednostki mocy. Składa się on z trzech członów: proporcjonalnego, całkującego i różniczkującego. Ma on na celu utrzymanie prędkości obrotowej kół na zadanym poziomie. Regulator jest on opisany wzorem 4.1 [30].

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \quad (4.1)$$

gdzie

$e$  – uchyb regulacji

$K_{p,i,d}$  – wzmocnienia członów proporcjonalnego, całkującego i różniczkującego

Uchyb regulacji to różnica między wartością zadaną oraz wartością obecną. Współczynnik  $K_i$  i  $K_d$  mogą być także zapisane jako:

$$K_i = K_p T_i, K_d = K_p \frac{1}{T_d} \quad (4.2)$$

gdzie

$T_i$  – czas całkowania(zdwojenia)

$T_d$  – czas różniczkowania(wyprzedzenia)

Regulator opisany równaniem 4.1 działa w sposób ciągły. Takie działanie jest niemożliwe do zastosowania w praktyce, ponieważ istnieje tylko skończona liczba próbek jakie można wykonać w danej jednostce czasu. W związku z tym zastosowany został dyskretny regulator PID opisany wzorem 4.3 [31], w którym okres próbkowania zostaje odgórnie ustalony.

$$u[kh] = K_p e[kh] + K_i \sum_{i=0}^k e[ih] + K_d (e[kh] - e[kh-h]) \quad (4.3)$$

gdzie

$h$  – okres próbkowania

Okres próbkowania został ustalony na 10ms. Wyjście z regulatora jest poddane przeskalowaniu do wartości odpowiadającej sygnałowi PWM, a następnie podane jako sterowanie na silniki.

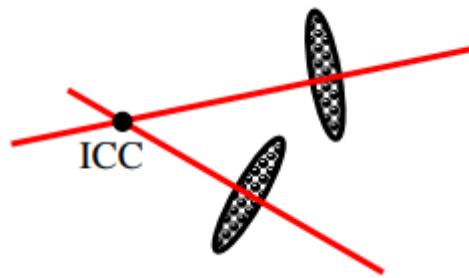
#### 4.1.1. Wartość zadana

Sterownik silników otrzymuje wartość zadaną od jednostki nadzędnej. Otrzymana wartość jest w postaci prędkości liniowej i kątowej, z jaką ma poruszać się robot. Na podstawie tych prędkości została wyznaczona prędkość z jaką poruszają się prawe i lewe koła robota [32].

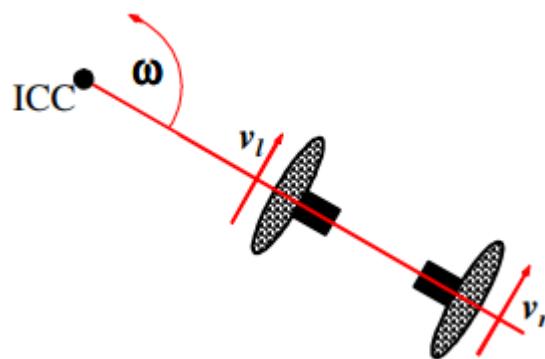
W przypadku poruszania się jednego koła, jego jeden pełen obrót powoduje przesunięcie środka wzdłuż własnej osi o dystans równy  $2\pi r$ , gdzie  $r$  jest promieniem koła. W przypadku robota posiadającego kilka obrotowych kół, każde koło obraca się wokół własnej osi. Muszą one posiadać wspólny środek obrotu (Rys. 4.1). Ten punkt jest nazywany ICC (ang. *Instantaneous Center of Curvature*). Prędkość każdego koła musi być zgodna ze sztywnym obrotem pojazdu. Oznacza to, że koła nie mogą się poruszać względem siebie.

W tworzonym robocie mobilnym koła znajdujące się po tej samej stronie poruszają się z tą samą prędkością, w związku z tym model poruszania się został uproszczony do modelu robota dwu kołowego o napędzie różnicowym (Rys. 4.2).

Głównym parametrem wyprowadzenia równań kinematycznych jest prędkość kątowa  $\omega$ , zdefiniowana jako obrót każdego koła wokół punktu ICC wzdłuż koła o promieniu  $r$ .



Rys. 4.1. Dwa obrotowe koła muszą mieć wspólny punkt obrotu [32].



Rys. 4.2. Konfiguracja kół robota z napędem różnicowym [32].

Prędkość kół jest wyrażona wzorem  $v = 2\pi r / T$ , gdzie  $T$  jest czasem pełnego obrotu wokół punktu ICC. Prędkość kątowa jest zdefiniowana jako  $\omega = 2\pi / T$  i jest wyrażona w radianach na sekundę. Łącząc równania prędkości  $v$  i  $\omega$  otrzymujemy równanie 4.4.

$$\omega r = v \quad (4.4)$$

Lewe i prawe koła poruszają się z różnymi prędkościami liniowymi oraz po trajektoriach posiadający różny promień, dlatego wyznaczone zostały prędkości liniowe osobne dla koła prawego (4.5) i lewego (4.6).

$$\omega(R + \frac{l}{2}) = v_r \quad (4.5)$$

$$\omega(R - \frac{l}{2}) = v_l \quad (4.6)$$

gdzie

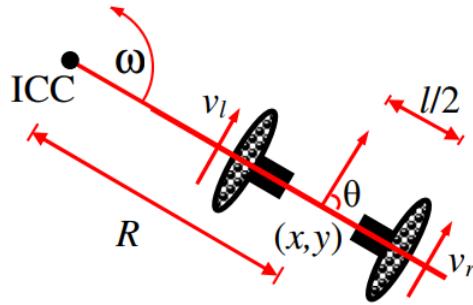
$R$  – odległość punktu ICC od środka robota

$l$  – odległość między kołami

Rozwiązujeając układ równań otrzymano równania opisujące prędkość kątową (4.7) i promień (4.8) układu. Ruch robota został zilustrowany na Rys. 4.3.

$$\omega = \frac{v_r - v_l}{l} \quad (4.7)$$

$$R = \frac{l}{2} \frac{v_l + v_r}{v_r - v_l} \quad (4.8)$$



Rys. 4.3. Ruch robota przy różnych prędkościach kół [32].

Podstawiając  $\omega$  i  $R$  do równania 4.4 otrzymano prędkość liniową z jaką porusza się robot (4.9).

$$v = \frac{v_l + v_r}{2} \quad (4.9)$$

Następnie wykorzystując równania 4.7 i 4.9 wyznaczono prędkość liniową prawego (4.10) i lewego (4.11) koła, w zależności od prędkości liniowej i kątowej z jaką porusza się robot.

$$v_r = v + \frac{1}{2}\omega l \quad (4.10)$$

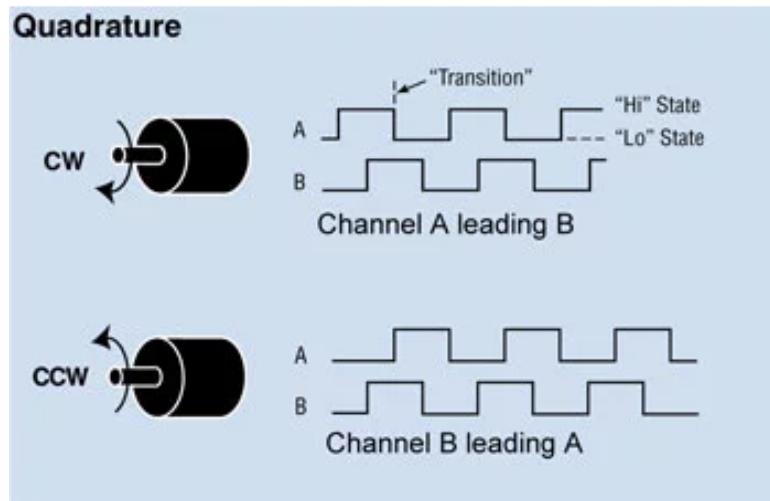
$$v_l = v - \frac{1}{2}\omega l \quad (4.11)$$

Wyznaczone prędkości liniowe lewego i prawego koła są wartościami zadanymi regulatorów. Wyrażone są w metrach na sekundę.

#### 4.1.2. Wartość pomiarowa

Wartością zmierzoną wykorzystywaną w regulatorach jest prędkość wyznaczona na podstawie enkoderów kwadraturowych. Liczniki impulsów zostały zrealizowane przy pomocy przerwań zewnętrznych, które wyzwalane są zboczem narastającym sygnału A (Rys. 4.4). W momencie wystąpienia impulsu sprawdzany jest stan sygnału B, daje to informację na temat kierunku przesunięcia fazowego sygnałów. Na tej podstawie licznik jest inkrementowany lub dekrementowany. Enkodery po przeciwnych stronach są zliczane odwrotnie.

Obliczenie prędkości kątowej możliwe jest dzięki wyznaczeniu zmiany kąta obrotu w czasie [34]. Znając liczbę impulsów enkodera na pełen obrót, wyliczono o jaki kąt wykonany został obrót. Przyjęty czas próbkowania wyniósł 5ms. Wyznaczona prędkość kątowa opisana jest wzorem 4.12. Dysponując prędkością kątową i promieniem koła łatwo wyznaczyć prędkość liniową 4.13.



Rys. 4.4. Przebieg sygnałów enkodera kwadraturowego [33].

$$\omega = \frac{d\Theta}{dt} = \frac{\Delta}{T_s} = \frac{2\pi * \Delta N}{N_p * T_s} \quad (4.12)$$

gdzie

$N_p$  – rozdzielcość enkodera

$T_s$  – czas próbkowania

$\Delta N$  – ilość zliczonych impulsów

$$v = \frac{2\pi\Delta N}{N_p T_s} R \quad (4.13)$$

Wyznaczone w ten sposób prędkości liniowe poszczególnych kół pełnią rolę wartości obecnej w regulatorach.

## 4.2. Odometria

Odometria polega na wykorzystaniu danych z czujników ruchów do oszacowania pozycji robota mobilnego względem jego początkowej lokalizacji [35]. Najprostszą metodą jest odometria kół. Służy ona do wyznaczenia położenia robota mobilnego poprzez zliczenie obrotów kół mających kontakt z podłożem. Obroty te można dokładnie przełożyć na przesunięcie liniowe względem podłożu.

Pozycja robota jest reprezentowana przez wektor  $X_t = [x_R, y_R, \Theta_R]^T$  [36]. Posiadając informacje na temat przemieszczenia robota oraz zmiany orientacji możliwe jest wyznaczenie jego obecnej lokalizacji.

Przemieszczenie punktu środkowego robota wyznaczone zostało na podstawie przemieszczeń lewych i prawych kół [37]:

$$\Delta U_i = \frac{\Delta U_R + \Delta U_L}{2} \quad (4.14)$$

Następnie wyznaczona została przyrostowa zmiana orientacji robota:

$$\Delta\Theta_i = \frac{\Delta U_R + \Delta U_L}{l} \quad (4.15)$$

gdzie

$l$  – odległość między kołami

Nowa orientacja robota może być wyznaczona jako:

$$\Theta_i = \Theta_{i-1} + \Delta\Theta_i \quad (4.16)$$

Względne położenie punktu środkowego można przedstawić w postaci

$$x_i = x_{i-1} + \Delta U_i \cos \Theta_i \quad (4.17)$$

$$y_i = y_{i-1} + \Delta U_i \sin \Theta_i \quad (4.18)$$

### 4.3. Jednostka inercyjna

Jednostka inercyjna MPU9250 posiada dedykowaną od producenta bibliotekę kompatybilną z platformą Arduino [38]. Pozwala ona na inicjalizację komunikacji SPI lub I2C deklarując przy tym odpowiednie piny przystosowane do takich magistrali komunikacyjnych. Funkcja inicjalizacyjna zwraca informację o statusie jednostki, pozwala to na detekcję błędów przed rozpoczęciem odczytów. Biblioteka umożliwia w łatwy sposób odczyt danych z czujników. Posiada możliwość odczytu:

- Przyspieszenia liniowego dla 3-osi
- Prędkości kątowa dla 3-osi
- Orientacji w przestrzeni w postaci kwaternionu
- Temperatury

## 5. Komunikacja między wątkowa w środowisku ROS

Komunikacja między minikomputerem pokładowym, a zdalnym komputerem została zrealizowana w oparciu o komunikację między wątkową ROS. Wykorzystana wersja systemu to ROS Kinetic Kame, która jest kompatybilna z systemem Ubuntu 16.04 Xenial.

Komunikacja przebiega poprzez sieć wi-fi. W tym celu w pliku `/.bashrc` zostały zdefiniowane adresy IP wykorzystywane do komunikacji między wątkowej (Tabela 5.1). Umożliwiło to wykorzystanie tego samego rdzenia komunikacji. Węzły odpowiedzialne za generowanie wartości zadanej zostały zainicjalizowane na zdalnym komputerze, natomiast węzły odpowiedzialne za czujniki oraz połączenie z sterownikiem silnika zostały utworzone na minikomputerze pokładowym.

**Tabela 5.1.** Definicja adresów wykorzystywanych przez ROS (opracowanie własne).

	Zdalny komputer	Platforma mobilna
ROS_MASTER_URI	<code>http://IP_ZDALNEGO_KOMPUTERA:11311</code>	<code>http://IP_ZDALNEGO_KOMPUTERA:11311</code>
ROS_HOSTNAME	<code>IP_ZDALNEGO_KOMPUTERA</code>	<code>IP_PLATFORMY_MOBILNEJ</code>

### 5.1. Komunikacja jednostki zdalnej

Rdzeń komunikacji jest zainicjalizowany na zdalnej jednostce, ponieważ posiada większą moc obliczeniową. Dodatkowo są na niej uruchamiane węzły odpowiedzialne za zadawanie sterowania, czyli prędkości kątowej i liniowej. Zostało to zrealizowane na trzy sposoby: z wykorzystaniem klawiatury, joysticka oraz MATLABA.

#### 5.1.1. Sterowanie w wykorzystaniem klawiatury

Zadawanie prędkości liniowej i kątowej z wykorzystaniem klawiatury zostało zrealizowane przy pomocy biblioteki *ros-kinetic-teleop-twist-keyboard* [39]. Sterowanie odbywa się poprzez zwiększenie lub zmniejszenie prędkości i wybór, która z prędkości ma być aktywna (Rys. 5.1). Publikowana wiadomość jest w formacie *geometry\_msgs/Twist*, składa się ona z dwóch trójwymiarowych wektorów, jeden z nich odpowiada za prędkość liniową, a drugi za prędkość kątową.

```

Moving around:
    u      i      o
    j      k      l
    m      ,      .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
anything else : stop

CTRL-C to quit

```

Rys. 5.1. Sterowanie z wykorzystaniem klawiatury [39].

### 5.1.2. Sterowanie z wykorzystaniem joysticka

Zostało zrealizowane z wykorzystaniem biblioteki *ros-kinetic-joy* [40]. Wychylenie lewej gałki powoduje zmianę wartości prędkości liniowej i kątowej w zależności od kierunku wychylenia. Wiadomość jest publikowana w formacie *sensor\_msgs/Joy*. Składa się on z wektora określającego poziom wychylenia gałek oraz wektora ze stanem wszystkich dostępnych guzików. Informacja o przyciskach nie jest wykorzystywana, lecz pozwala na rozbudowanie sterowania z wykorzystaniem joysticka o dodatkowe funkcjonalności.

### 5.1.3. Sterowanie z wykorzystaniem środowiska MATLAB

Środowisko MATLAB oferuje pakiet *ROS Toolbox*, który pozwala na współpracę z systemem ROS. Zawiera on gotowe funkcje odpowiedzialne za inicjalizację węzła oraz publikowanie i subskrybowanie tematu. Dodatkowo obsługuje formaty danych wykorzystywane w komunikacji między wątkowej. ROS Toolbox posiada także rozszerzenie do Simulink, dzięki któremu dostępne są nowe bloki umożliwiające obsługę ROS.

Sterowanie z wykorzystaniem MATLABa opiera się na stworzeniu węzła w tym środowisku. Program umożliwia stworzenie skryptu, w którym można zadawać sterowanie poprzez publikowanie wiadomości. Wykorzystany format to *geometry\_msgs/Twist*. Takie rozwiązanie posiada przewagę nad dwoma poprzednimi, ponieważ możliwe jest generowanie sterowania na podstawie odczytów z czujników, które węzeł subskrybuje. Pozwala to na stworzenie algorytmów, generujących trajektorię poruszania się robota. Dodatkowo węzeł stworzony w MATLABie pozwala w prosty sposób zmieniać nastawy regulatora wykorzystywanego przez sterownik silników.

## 5.2. Komunikacja jednostki pokładowej

Minikomputer wbudowany łączy się z rdzeniem uruchomionym na jednostce zdalnej. Odpowiada on za uruchomienie trzech węzłów, odpowiadających za system wizyjny, lidar oraz sterownik silników. Węzły są inicjalizowane przy pomocy skryptu napisanego w języku python.

### 5.2.1. Komunikacja systemu wizyjnego

Wykorzystanym systemem wizyjnym jest kamera RealSense R200, która jest kompatybilna z minikomputerem pokładowym UpBoard. Komunikacja między wątkowa ROS została przeprowadzona w oparciu o bibliotekę *realsense\_camera* [41]. Umożliwia ona inicjalizację węzła, który publikuje wiadomości o odczytach z kamery (Tabela 5.2). Kamera umożliwia uzyskanie obrazu w formacie RGB oraz w podczerwieni. Dodatkowo pozwala uzyskać głębie obrazu poprzez wykorzystanie chmury punktów stworzonej dzięki wbudowanej drugiej kamerze.

**Tabela 5.2.** Wiadomości publikowane przez kamerę RealSense R200 [41].

Temat	Opis wiadomości	Format wiadomości
color/camera_info	kalibracja kamery	sensor_msgs/CameraInfo
color/image_raw	obraz w formacie RGB	sensor_msgs/Image
depth/camera_info	kalibracja kamery	sensor_msgs/CameraInfo
depth/image_raw	surowy obraz z urządzenia, zawiera głębokość w mm	sensor_msgs/Image
depth/points	zarejestrowana chmura punktów XYZRGB	sensor_msgs/PointCloud2
ir/camera_info	kalibracja kamery	sensor_msgs/CameraInfo
ir/image_raw	surowy obraz IR z pierwszej kamery	sensor_msgs/Image
ir2/camera_info	kalibracja kamery	sensor_msgs/CameraInfo
ir2/image_raw	surowy obraz IR z drugiej kamery	sensor_msgs/Image

### 5.2.2. Komunikacja Lidaru

Komunikacja z laserowym skanerem została przeprowadzona z wykorzystaniem biblioteki *rplidar\_ros* [42]. Pozwala ona na inicjalizację węzła odpowiedzialnego za lidar. Publikuje on w temacie */scan* wiadomość w formacie *sensor\_msgs/LaserScan*. Pozwala to na stworzenie mapy punktów, które zostały zeskanowane przez laser. Lidar umożliwia zbudowanie mapowa otoczenia robota.

### 5.2.3. Komunikacja sterownika silnika

Komunikacja między wątkowa sterownika silnika jest możliwa dzięki bibliotece *roserial*. Pozwala ona na inicjalizację własnego węzła przez wykorzystany mikro kontroler STM32. Połączenie odbywa się z prędkością transmisji wynoszącą 57600 i wykorzystuje port USB */dev/ttyACM0*.

W oprogramowaniu mikro kontrolera został utworzony węzeł, który pozwala na subskrybowanie tematów i publikację wiadomości.

### 5.2.3.1. Tematy subskrybowane przez sterownik silników

Subskrypcja tematów pozwala na uzyskanie dostępu do wiadomości, która pojawiła się na obserwowanym temacie. W momencie pojawienia się nowej wiadomości wywołana zostaje funkcja przypisana do danego tematu. Wewnątrz takiej funkcji znajduje się kod programu odpowiedzialny za przypisanie wiadomości do zmiennych, skąd mogą być wykorzystane w programie.

Głównymi tematami subskrybowanymi przez sterownik silników są tematy związane z zadawaniem prędkości liniowej i kątowej (Tabela 5.3). Dodatkowo subskrybowany jest temat zawierający nastawy regulatora PID, pozwala to w łatwy sposób nimi manipulować.

**Tabela 5.3.** Tematy subskrybowane przez sterownik silników (opracowanie własne).

Temat	Opis wiadomości	Format wiadomości
cmd_vel	prędkość liniowa i kątowa zadana poprzez konsolę systemową	geometry_msgs/Twist
joy	prędkość liniowa i kątowa zadana poprzez joystick	sensor_msgs/Joy
mat_vel	prędkość liniowa i kątowa zadana poprzez środowisko MATLAB	geometry_msgs/Twist
pid_param	nastawy regulatora PID	std_msgs/Float32MultiArray

### 5.2.3.2. Wiadomości publikowane przez sterownik silników

Sterownik silników publikuje także własne wiadomości na tematy subskrybowane przez zewnętrzne węzły. Podaje on informację o statusie silników, odczytach z czujników, a także wyznaczonej odometrii (Tabela. 5.4). Stworzony został także temat pozwalający na detekcję błędów w oprogramowaniu. Tematy posiadają różną częstotliwość publikowania wiadomości, w zależności od ich zastosowania.

**Tabela 5.4.** Wiadomości publikowane przez sterownik silników (opracowanie własne).

Temat	Opis wiadomości	Format wiadomości	Częstotliwość nadawania [Hz]
imu	odczyt z czujników jednostki inercyjnej	sensor_msgs::Imu	200
odom	wartości wyznaczonej odometrii robota	nav_msgs::Odometry	200
imu_state	stan jednostki inercyjnej, wartość ujemna oznacza błąd	std_msgs::Int32	30
l_engines_state	status silników po lewej stronie, '1' oznacza błąd	std_msgs::Bool	30
r_engines_state	status silników po prawej stronie, '1' oznacza błąd	std_msgs::Bool	30
debug	temat służący do debugowania programu sterownika	geometry_msgs::Twist	200

## 6. Testy

Przeprowadzone testy dzielą się na badanie odpowiedzi robota na zadane sterowanie oraz odczyt i przetworzenie danych z elementów układu sensorycznego. Do pierwszej grupy testów należy dobór nastaw regulatora PID, pomiar przejechanej odległości oraz wyznaczenie trajektorii ruchu robota. Do drugiej kategorii zalicza się SLAM (*ang. Simultaneous localization and mapping*), czyli mapowanie otoczenia oraz odczyt danych z systemu wizyjnego.

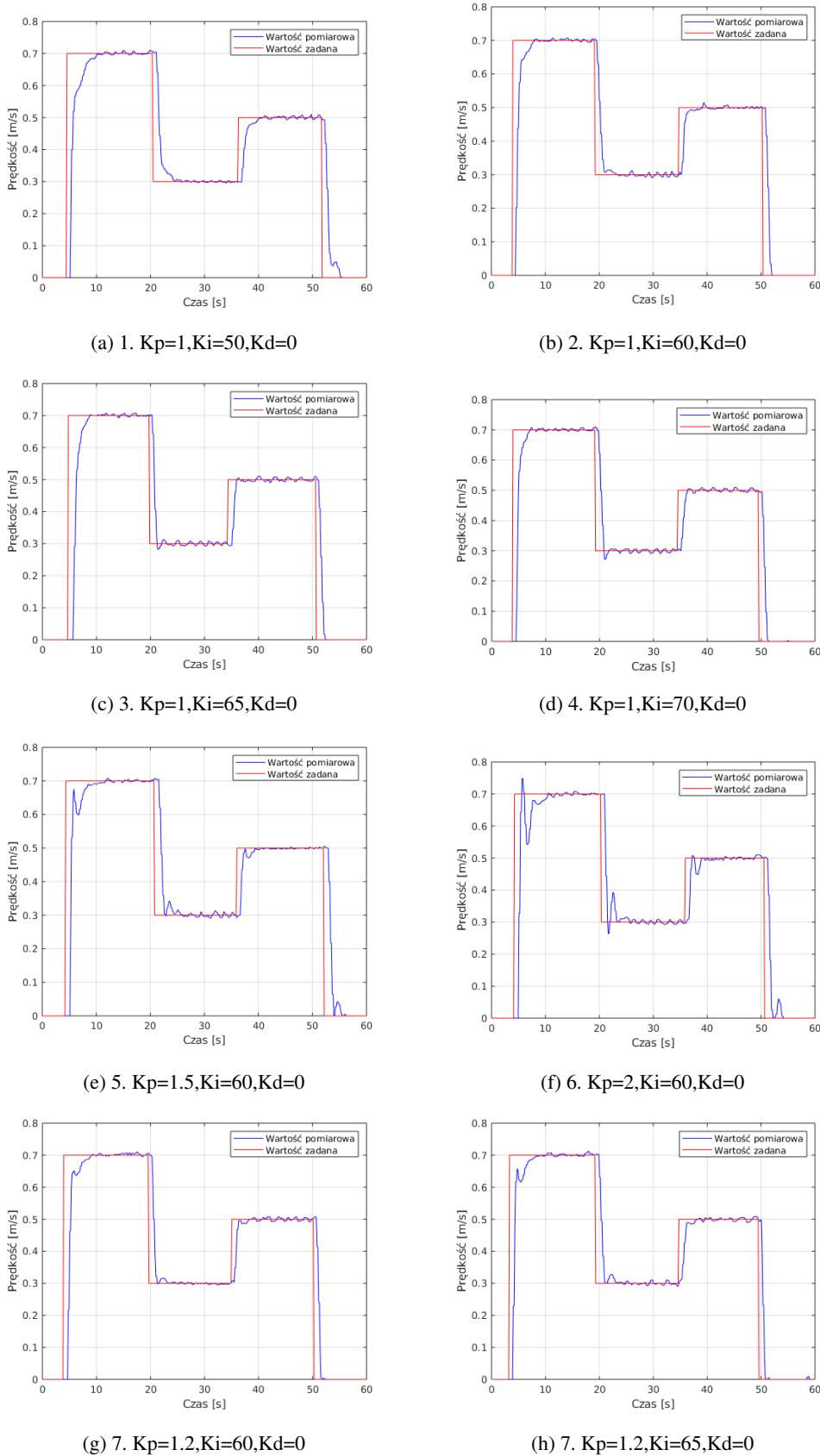
Testy zostały przeprowadzone z wykorzystaniem narzędzi programowych RViz oraz MATLAB Robotics System Toolbox.

### 6.1. Dobór nastaw regulatora PID

Dobór nastaw regulatora został przeprowadzony metodą eksperymentalną, poprzez test zachowania układu dla różnych nastaw regulatora. Człon części różniczkującej został wyzerowany, ponieważ jest on wrażliwy na zakłóczenia, których w przypadku robota jeżdżącego nie da się wyeliminować. Wykorzystany więc został regulator PI. Punktem wyjściowym testowanych wzmacnień było ich wyznaczenie metodą wzmacnienia krytycznego. Wyznaczone nastawy powodowały jednak duży błąd, w związku z tym przeprowadzone zostały testy działania dla innych, ręcznie dobranych wzmacnień.

W pierwszej kolejności została przeprowadzony test działania nastaw dla nieobciążonych kół. Zrealizowane to zostało za pomocą skryptu napisanego w środowisku MATLAB. Zadawał on wzmacnienia i zadaną dla silników prędkość. Subskrybował także temat z informacją o obecnej prędkości wyznaczonej na podstawie enkoderów. Zadawana prędkość była zmieniana w czasie, co pozwoliło na lepszą obserwację zachowania układu (Rys. 6.1). Kryterium wyboru nastaw był kwadrat uchybu, czyli różnicy między wartością zadaną, a obecną (Tabela 6.1).

Na podstawie wyznaczonych wartości błędów oraz odczytanych z wykresu przeregulowań, jako najbardziej optymalne przyjęte zostały nastawy 2, 7 i 8. Przeprowadzono dla nich testy pod obciążeniem. Przez określoną chwilę czasową zadawana była stała prędkość. Na tej podstawie został wyznaczony kwadrat błędu dla różnych zadawanych prędkości (Tabela 6.2). Maksymalną prędkością dla których został wykonany test jest  $1\frac{m}{s}$ , ponieważ przestrzeń testowa pozwoliła na zadawanie tej prędkości przez maksymalnie 3 sekundy, co jest niewystarczającym czasem do osiągnięcia stanu ustalonego. Na podstawie przeprowadzonych testów przyjęte zostały nastawy  $K_p = 1$ ,  $K_i = 60$  i  $K_d = 0$ .



**Rys. 6.1.** Odpowiedź układu na zadane sterowanie dla różnych nastaw regulatora PID (pracowanie własne).

**Tabela 6.1.** Kwadrat błędu dla różnych nastaw regulatora PID bez obciążenia (opracowanie własne).

Lp.	Nastawy regulatora	Kwadrat błędu
1.	$K_p = 1, K_i = 50, K_d = 0$	6,547
2.	$K_p = 1, K_i = 60, K_d = 0$	6,063
3.	$K_p = 1, K_i = 65, K_d = 0$	6,148
4.	$K_p = 1, K_i = 70, K_d = 0$	6,758
5.	$K_p = 1.5, K_i = 60, K_d = 0$	6,119
6.	$K_p = 2, K_i = 60, K_d = 0$	6,599
7.	$K_p = 1.2, K_i = 60, K_d = 0$	6,020
8.	$K_p = 1.2, K_i = 65, K_d = 0$	6,121

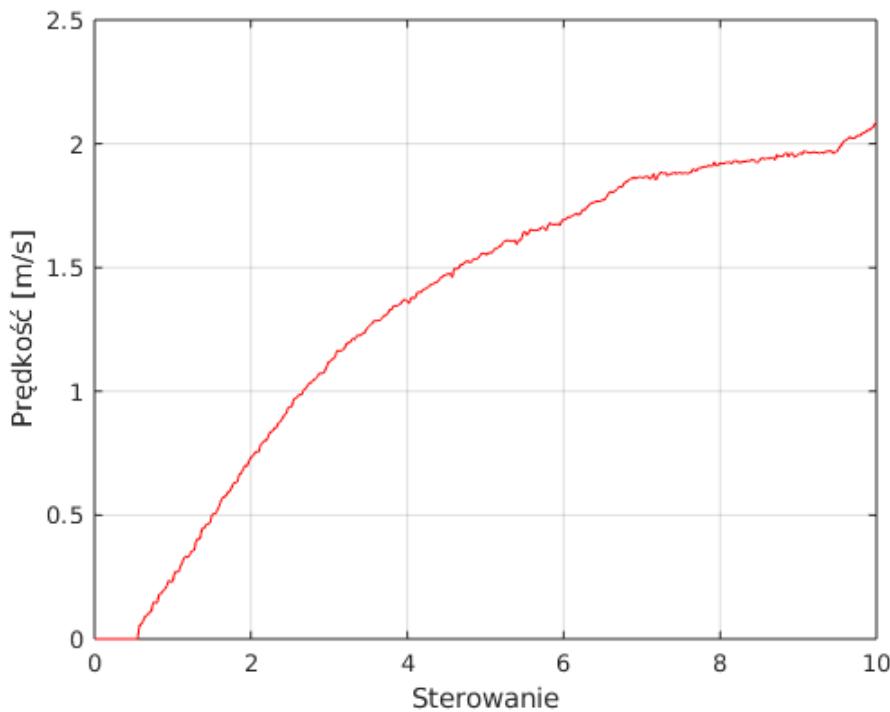
**Tabela 6.2.** Kwadrat błędu dla nastaw regulatora przy różnych prędkościach pod obciążeniem (opracowanie własne).

Prędkość [ $\frac{m}{s}$ ] \ Nastawy	2. $K_p = 1, K_i = 60$	7. $K_p = 1.2, K_i = 60$	8. $K_p = 1.2, K_i = 65$
0.05	0.079	0.1404	0.1022
0.1	0.2212	0.2703	0.2111
0.2	0.7867	0.8328	0.8358
0.3	1.2575	1.4784	1.309
0.5	4.7334	4.5579	4.3669
0.6	4.3067	4.7457	4.4634
0.8	7.9647	8.0086	7.7945
1	12.1673	12.7481	12.7223

Wyjście z regulatora przyjmuje wartości z przedziału [-10, 10], gdzie znak odpowiada za kierunek obrotu. Sygnał ten jest skalowany do wartości z przedziału [-400, 400], a następnie podawany jako wejściowy sygnał PWM dla silników.

Regulator działa z okresem próbkowania wynoszącym 10ms. Pozwala to na szybkie działanie regulatora oraz nie powoduje dużych błędów związanych z małą liczbą impulsów enkodera.

Kolejnym etapem było przeprowadzenie badania prędkości kół na skutek sterowania zadanego przez regulator. Pozwoliło to na uzyskanie informacji o minimalnej wartości sterowania dla którego następuje rozruch, minimalnej oraz maksymalnej prędkości (Rys. 6.2). Test został przeprowadzony bez obciążenia. Rozruchową wartością sterowania jest 0.6, pozwala to na osiągnięcie prędkości  $0.05 \frac{m}{s}$ . Maksymalną uzyskaną prędkością jest w przybliżeniu  $2 \frac{m}{s}$ . Wyznaczona teoretyczna prędkość maksymalna ma wartość  $1.94 \frac{m}{s}$ , różnica może wynikać z błędów pomiaru prędkości.



Rys. 6.2. Przebieg prędkości w zależności od sterowania zadanego przez regulator (opracowanie własne).

## 6.2. Pomiar pokonanej drogi wyznaczonej za pomocą enkoderów

Kolejnym przeprowadzonym testem był pomiar odległości na podstawie zliczonej liczny impulsów enkoderów. Przebyta droga została wyznaczona wg. wzoru:

$$s = \frac{2\pi R \Delta N}{N_p} \quad (6.1)$$

gdzie

$N_p$  – rozdzielczość enkodera

$R$  – promień koła - 0.067m

$\Delta N$  – ilość zliczonych impulsów

Test został przeprowadzony poprzez zadanie prędkości liniowej i rzeczywistym zmierzeniu przemieszczenia robota. Zrealizowane to zostało za pomocą skryptu napisanego w środowisku MATLAB. Przez określoną chwilę czasową zadawana była stała prędkość liniowa. Pomiary zostały wykonane dla różnych wartości prędkości (Tabela 6.3). Testy przeprowadzone były dla enkoderów po dwóch różnych stronach robota. Otrzymane błędy względne pomiaru odległości nie przekraczają 5%, oznacza to, że mieszczą się w granicach normy, jednakże mogą wprowadzać błędy w pomiarze prędkości.

**Tabela 6.3.** Błąd odległości na podstawie enkoderów (opracowanie własne).

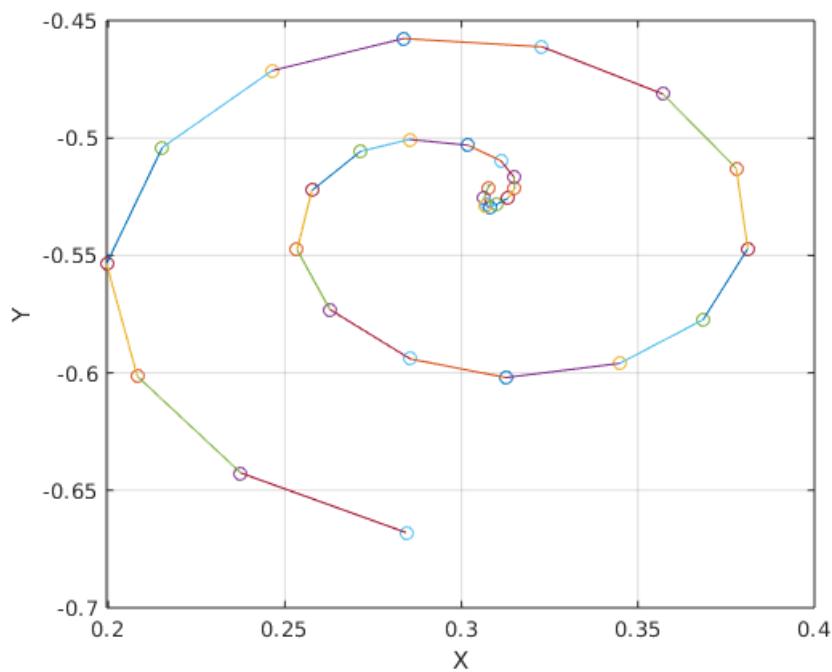
<b>P<small>rz</small>edkość</b> [ $\frac{m}{s}$ ]	<b>D<small>roga</small></b> <b>enkoder</b> <b>lewy [m]</b>	<b>D<small>roga</small></b> <b>enkoder</b> <b>prawy [m]</b>	<b>D<small>roga</small></b> <b>rzeczywista</b> [m]	<b>B<small>łąd</small></b> <b>wzgl<small>ędny</small></b> <b>enkoder lewy</b> [%]	<b>B<small>łąd</small></b> <b>wzgl<small>ędny</small></b> <b>enkoder prawy</b> [%]
0.1	0.886	0.955	0.93	4.75	2.68
0.2	0.966	0.974	0.94	2.74	3.65
0.3	1.773	1.783	1.72	3.11	3.63
0.4	1.535	1.512	1.47	4.40	2.89
0.5	1.923	1.966	1.93	0.32	1.86
0.6	1.558	1.586	1.54	1.18	2.98
0.7	1.978	2.005	1.97	0.40	1.78
0.8	1.973	1.968	1.95	1.19	0.94
0.9	1.949	1.945	1.91	2.08	1.86
1	1.936	1.950	1.92	0.82	1.58
1.1	2.212	2.195	2.15	2.91	2.10
1.2	1.433	1.4512	1.44	0.48	0.77
1.3	1.7374	1.75	1.71	1.60	2.34
1.4	1.477	1.4632	1.45	1.862	0.91

### 6.3. Trajektoria ruchu robota na podstawie odometrii

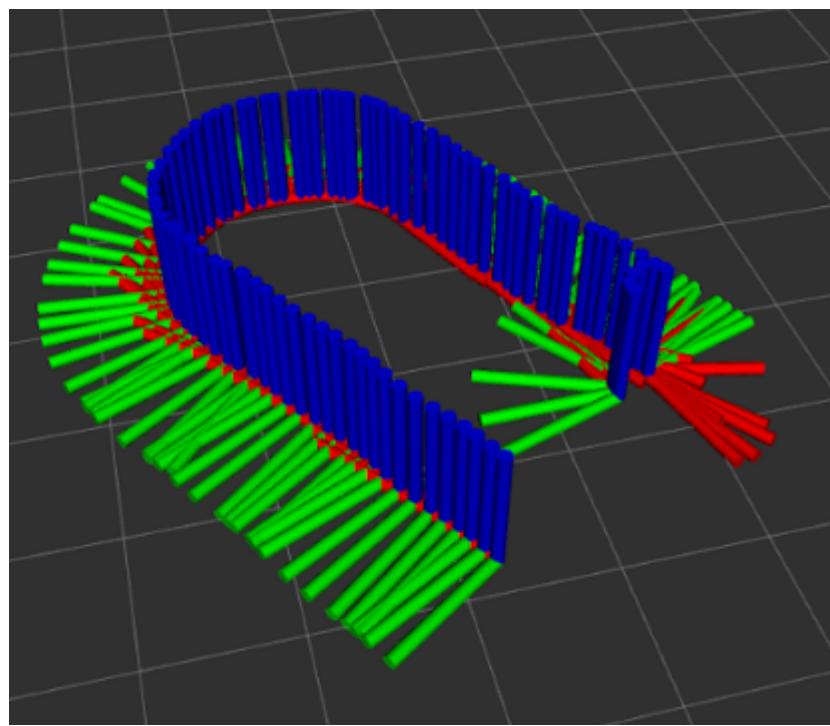
Wyznaczenie aktualnej pozycji robota zostało zrealizowane za pomocą odometrii, która została wyznaczona w rozdziale 4.2, na podstawie odczytów z enkoderów. Przykładowe trajektorie są tworzone poprzez subskrypcję tematu */odom*, w którym publikowana jest wiadomość na temat pozycji w układzie dwuwymiarowym oraz orientacja robota. Poprawność działania odometrii została przetestowana przy użyciu skryptu stworzonego w środowisku MATLAB oraz w programie RViz.

Program stworzony w MATLABie zapisywał aktualne współrzędne publikowane w temacie */odom*. Pozwoliło to na uzyskanie informacji o zmianie położenia w czasie. Zadana została stała prędkość kątowa oraz stopniowo zwiększano prędkość liniową w celu stworzenia trajektorii w kształcie spirali. Wynik działania skryptu zostały przedstawione na Rys. 6.3.

Program RViz również pozwala generować trajektorię ruchu robota na podstawie informacji publikowanych w temacie */odom*. RViz umożliwia zilustrowanie orientacji robota, pozwala to na określenie w którą stronę się porusza. Aktualne położenie jest wyświetlane w postaci strzałki lub punktu w układzie trzyosiowym. Zapisując poprzednie współrzędne, pozwala na odtworzenie drogi pokonanej przez robota. Przykładową wygenerowaną trajektorię przedstawiono na Rys. 6.4. Została ona stworzona z wykorzystaniem biblioteki *teleop-twist-keyboard* do sterowania robotem.



Rys. 6.3. Zarejestrowana trajektoria ruchu robota wygenerowana z wykorzystaniem skryptu napisanego w MATLABie (opracowanie własne).

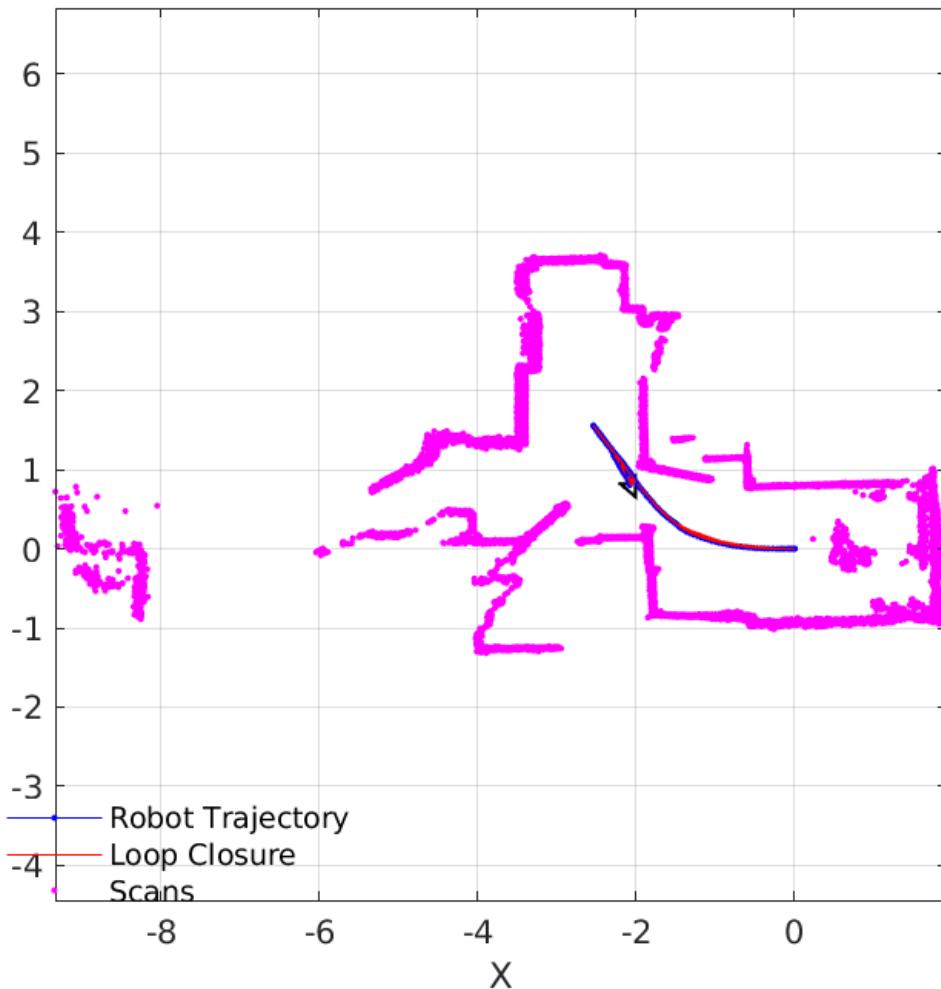


Rys. 6.4. Przykładowa trajektoria ruchu robota zarejestrowana w RViz (opracowanie własne).

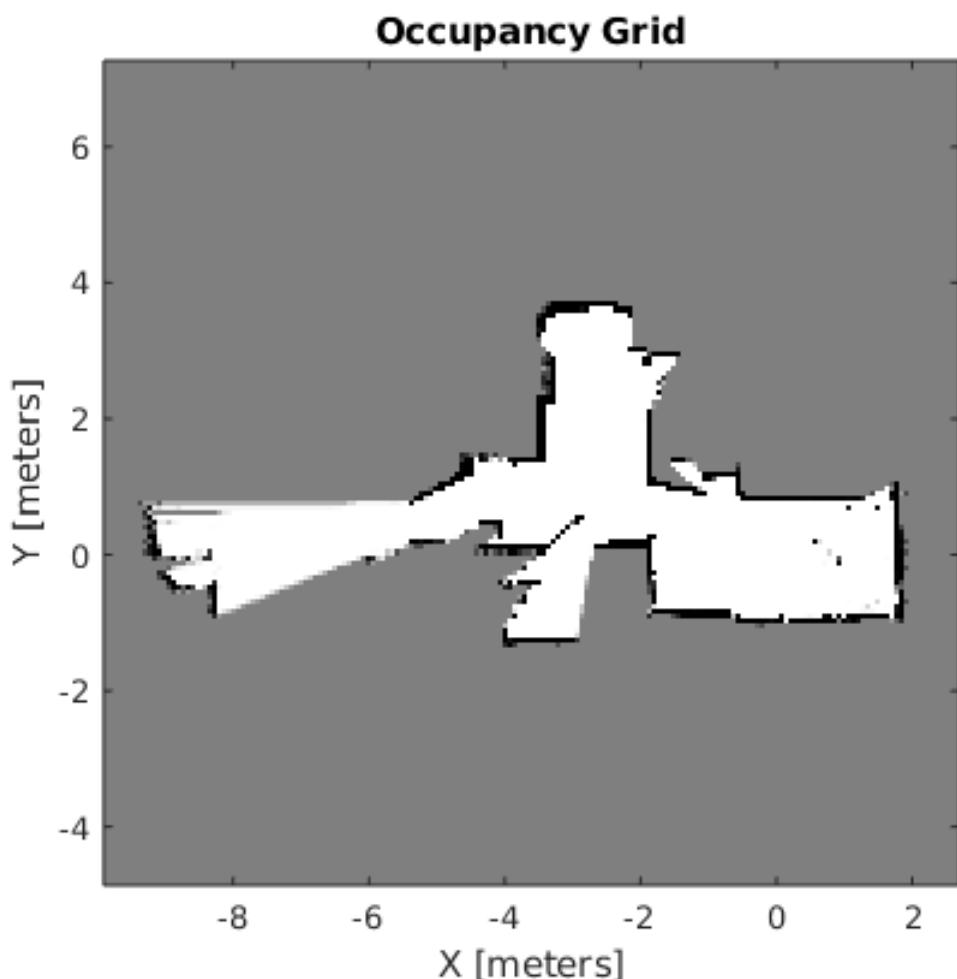
## 6.4. Mapowanie pomieszczenia z wykorzystaniem lidaru

Działanie lidaru zostało przetestowane poprzez stworzenie mapy pomieszczenia. W tym celu nagrany został plik *.bag*, w którym przez wyznaczony okres czasu, zapisywane były wiadomości publikowane w temacie */scan*. Wiadomości te, to informacje o punktach zeskanowanych przez lidar. Na podstawie zapisanych informacji została zbudowana mapa otoczenia (SLAM).

W celu stworzenia SLAM, wykorzystane zostało narzędzie MATLAB'a jakim jest *SLAM Map Builder*. Na podstawie wgranego pliku *.bag*, tworzy podgląd zeskanowanych punktów oraz trajektorii ruchu robota (Rys. 6.5). Umożliwia także wybór procentu punktów wykorzystanych do zbudowania mapy otoczenia. W tym przypadku wykorzystano 100% danych. Na tej podstawie zbudowany został obraz SLAM (Rys. 6.6). W tym przypadku także wykorzystana została biblioteka *teleop-twist-keyboard* do sterowania robotem.



Rys. 6.5. Punkty zeskanowane przez lidar (opracowanie własne).



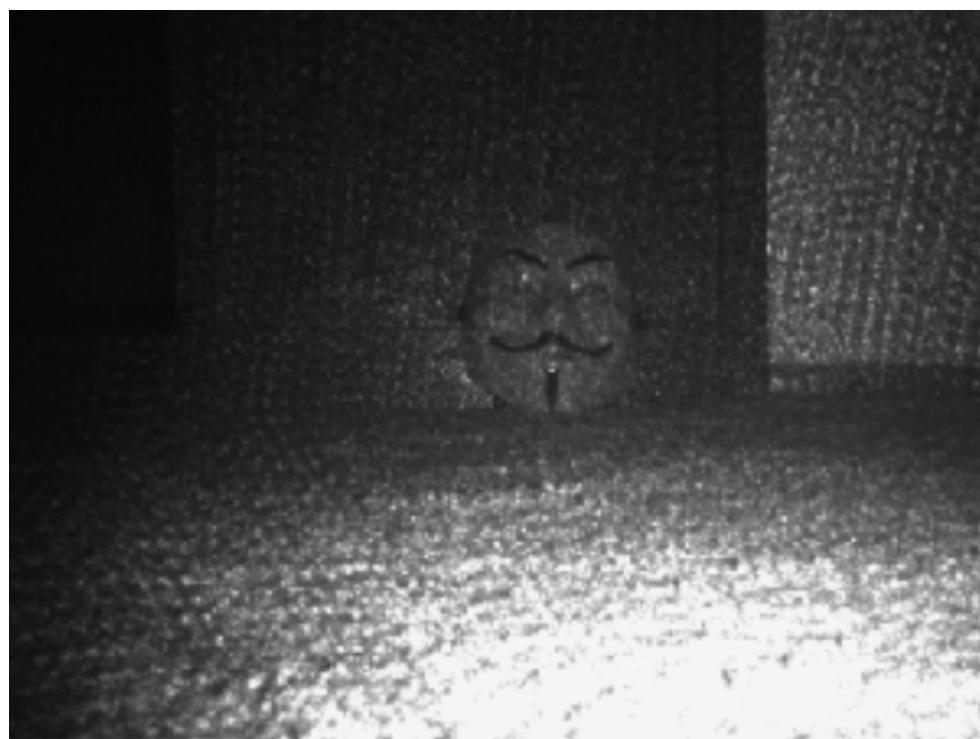
Rys. 6.6. Zbudowana mapa otoczenia (opracowanie własne).

## 6.5. Test działania systemu wizyjnego

Testy działania systemu wizyjnego, czyli kamery RealSense R200 zostały przeprowadzone z wykorzystaniem programu RViz. Umożliwia on wyświetlenie wiadomości publikowanych w tematach kamery. Subskrypcja tematu *color/image\_raw* pozwala na stworzenie obrazu w formacie RGB (Rys. 6.7). Dane publikowane w temacie *ir/image\_raw* umożliwiają stworzenie obrazu w podczerwieni (Rys. 6.8), natomiast wiadomość publikowana w temacie *depth/image\_raw* pozwala na wyświetlenie głębi obrazu (Rys. 6.9). Zaletą programu RViz jest możliwość wyświetlania wiadomości publikowanej w temacie *depth/points*, jest to chmura punktów przedstawiona w trzech wymiarach (Rys. 6.10). Dzięki temu możliwe jest zaobserwowanie głębi obrazu.



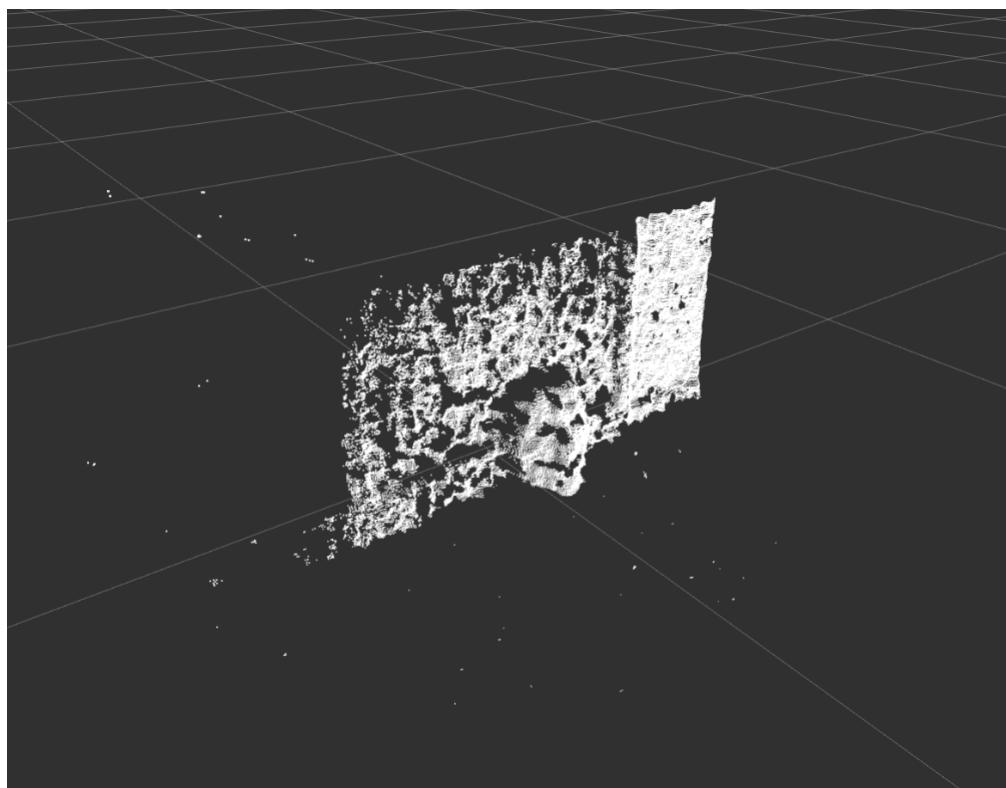
Rys. 6.7. Obraz z kamery w barwach RGB (opracowanie własne).



Rys. 6.8. Obraz z kamery w podczerwieni (opracowanie własne).



Rys. 6.9. Głębia obrazu przedstawiona w 2D (opracowanie własne).



Rys. 6.10. Głębia obrazu w postaci chmury punktów (opracowanie własne).

## **7. Podsumowanie**

Cel pracy dyplomowej, jakim było zaprojektowanie, zbudowanie oraz uruchomienie platformy dydaktycznej robota mobilnego, kołowego o napędzie elektrycznym, różnicowym, został zrealizowany. Wykorzystana obudowa, okazała się być idealną do zakładanych funkcjonalności. Zaprojektowany i wykonany układ elektryczny pozwolił na zakładaną komunikację między podzespołami. Stworzone oprogramowanie pozwoliło na sterowanie silnikami oraz publikację informacji na temat parametrów ruchu. Komunikacja między wątkowa umożliwiła przesył danych w wygodny sposób.

Przeprowadzone w rozdziale szóstym testy, potwierdzają, że zbudowana platforma działa poprawnie. Odpowiedni dobór nastaw regulatora PID, umożliwił stabilizację prędkości zadanej. Test pomiaru odległości przez enkodery, dostarczył informacji na temat błędu względnego zliczania impulsów. Nie przekracza on 5%, jest to akceptowalny wynik. Badanie działania odometrii pozwoliło zobrazować trajektorię ruchu robota. Udało się także przeprowadzić mapowanie otoczenia dzięki zainstalowanemu lidarowi. Zbudowany system wizyjny pozwala na uzyskanie obrazu otoczenia robota.

Mimo poprawnego działania wymienionych wyżej modułów, podczas przeprowadzania testów zaobserwowano problem z działaniem komunikacji między wątkowej. Polegał on na tym, że robot nie zawsze odbierał publikowaną wiadomość. Sytuacja ta miała miejsce w przypadku publikowania pojedynczej wiadomości. Spowodowane to było wykonywaniem przez mikro kontroler innego działania w momencie nadejścia wiadomości, skutkowało to niewykonaniem się funkcji wywoywanej, przypisanej do tego tematu. Problem został rozwiązany poprzez publikowanie wiadomości więcej niż jeden raz. Uzyskano w ten sposób pewność, że mikro kontroler odbierze wiadomość, jednakże nieodebranie pierwszej wiadomości zwiększa opóźnienie komunikacji.

Problem wystąpił także przy użyciu jednostki inercyjnej. Działa ona poprawnie przy wyłączonych silnikach, jednakże w momencie ich uruchomienia przestaje nadpisywać wartości w rejestrze nowymi danymi z czujników. Powoduje to, że publikowana wartość się nie zmienia. Uniemożliwia to wykorzystanie danych z jednostki inercyjnej.

Zbudowana platforma dydaktyczna w dalszym stopniu nadaje się do rozwoju. System wizyjny daje możliwość stworzenia algorytmu do detekcji i omijania przeszkód, w tym celu jednakże należy zmienić moduł wi-fi, aby zwiększyć prędkość transmisji, co wpłynie korzystnie na publikowanie danych z kamery. Inną drogą rozwoju może być wykorzystanie narzędzia MATLAB Robotics System Toolbox do zadawania bardziej skomplikowanych trajektorii ruchu. Istnieje także możliwość zainstalowania dodatkowych czujników lub innych elementów zwiększających funkcjonalność robota.



## Bibliografia

- [1] Knani Jilani Tlijani Hayet Tlijani Hatem. *Navigation Control of a Mobile Robot under Time Constraint using Genetic Algorithms, CSP Techniques, and Fuzzy Logic*. Tunis, Tunisia, 2015.
- [2] [www.emanual.robotis.com/docs/en/platform/turtlebot3/overview/](http://www.emanual.robotis.com/docs/en/platform/turtlebot3/overview/). Strona domowa platformy turtlebot3.
- [3] [www.bostondynamics.com/handle](http://www.bostondynamics.com/handle). Robot Handle firmy BostonDynamics.
- [4] [www.robotworld.pl/sphero-r2d2-star-wars](http://www.robotworld.pl/sphero-r2d2-star-wars). Sphero R2-D2 Star Wars.
- [5] [www.bostondynamics.com/atlas](http://www.bostondynamics.com/atlas). Robot Atlas firmy BostonDynamics.
- [6] [www.bostondynamics.com/spot](http://www.bostondynamics.com/spot). Robot Spot firmy BostonDynamics.
- [7] [thekidshouldseethis.com/post/boston-dynamics-cheetah-robot](http://thekidshouldseethis.com/post/boston-dynamics-cheetah-robot). Robot Cheetah firmy BostonDynamics.
- [8] [www.ecagroup.com/en/solutions/umis-a-comprehensive-unmanned-mcm-integrated-system](http://www.ecagroup.com/en/solutions/umis-a-comprehensive-unmanned-mcm-integrated-system). Autonomiczny pojazd podwodny UMIS.
- [9] [gdmissionsystems.com/products/underwater-vehicles/bluefin-12-unmanned-underwater-vehicle](http://gdmissionsystems.com/products/underwater-vehicles/bluefin-12-unmanned-underwater-vehicle). Bezzałogowa łódź podwodna Bluefin-12 firmy General Dynamics.
- [10] [www.army-technology.com/projects/mq1c-gray-eagle-uas-us-army/](http://www.army-technology.com/projects/mq1c-gray-eagle-uas-us-army/). Bezzałogowa statek powietrzny General Atomics MQ-1C Gray Eagle.
- [11] [dji-ars.pl/dji-inspire-2-craft.html](http://dji-ars.pl/dji-inspire-2-craft.html). Dron DJI Inspire 2.
- [12] [www.space.com/24392-wall-crawling-gecko-space-robot-video.html](http://www.space.com/24392-wall-crawling-gecko-space-robot-video.html). Pełzający po ścianie robot gekon - Abigaille.
- [13] RyuWoon Jung TaeHoon Lim YoonSeok Pyo HanCheol Cho. *ROS Robot Programming*. Seoul, Republic of Korea: ROBOTIS Co.,Ltd., 2017.
- [14] Morgan Quigley. *ROS: an open-source Robot Operating System*. Stanford, CA, 2009.
- [15] Krzysztof Piech. *Leksykon pojęć na temat technologii blockchain i kryptowalut*. 2016.
- [16] [www.turtlebot.com](http://www.turtlebot.com). Strona domowa robotów turtlebot.
- [17] [www.dfrobot.com/product-94.html](http://www.dfrobot.com/product-94.html). 4WD Outdoor Mobile Platform.
- [18] [www.dfrobot.com/product-634.html](http://www.dfrobot.com/product-634.html). Silnik DC serii Gear Motor.

- [19] [www.pololu.com/product/2502](http://www.pololu.com/product/2502). Strona producenta Pololu Dual VNH5019 Motor Driver Shield.
- [20] [/www.pololu.com/docs/0J49/3.c](http://www.pololu.com/docs/0J49/3.c). Pololu Dual VNH5019 Motor Driver Shield - oznaczenia pinów.
- [21] *Datasheet - STM32L476xx*. Dokumentacja techniczna rodziny mikro kontrolerów STM32L476xx.
- [22] *UM1724 User manual STM32 Nucleo-64 boards (MB1136)*. Instrukcja użytkowania mikro kontrolerów rodziny STM32 Nucleo-64.
- [23] [up-board.org/up/specifications](http://up-board.org/up/specifications). Strona producenta platformy UP Board.
- [24] [software.intel.com/en-us/articles/intel-realsense-data-ranges](http://software.intel.com/en-us/articles/intel-realsense-data-ranges). Specyfikacja techniczna systemu wizyjnego Intel® RealSense™ R200.
- [25] *MPU-9250 Product Specification Revision 1.1*. Specyfikacja techniczna jednostki inercyjnej MPU-9250.
- [26] [www.creatroninc.com/product/mpu9250-9-dof-imu/](http://www.creatroninc.com/product/mpu9250-9-dof-imu/). MPU-9250.
- [27] [www.slamtec.com/en/lidar/a1](http://www.slamtec.com/en/lidar/a1). Strona producenta lidaru RPLIDAR A1.
- [28] [www.genstattu.com/tattu-3700mah-45c-4s1p-lipo-battery-pack-with-xt60-plug.html](http://www.genstattu.com/tattu-3700mah-45c-4s1p-lipo-battery-pack-with-xt60-plug.html). Akumulator litowo-polimerowy Tattu 3700mAh 45C.
- [29] [github.com/pololu/dual-vnh5019-motor-shield.git](https://github.com/pololu/dual-vnh5019-motor-shield.git). Biblioteka jednostki mocy VNH5019.
- [30] P. R. Shreeraman Hari Om Bansal Rajamayyoor Sharma. *PID Controller Tuning Techniques: A Review*. Pilani, India, 2012.
- [31] Piotr Ostalczyk Mariusz Matusiak. *Problems in solving fractional differential equations in a microcontroller implementation of an FOPID controller*. Poland, Łódź, 2019.
- [32] Thomas Hellström. *Kinematics Equations for Differential Drive and Articulated Steering*. 2011.
- [33] [www.dynapar.com/technology/encoder\\_basics/quadrature\\_encoder/](http://www.dynapar.com/technology/encoder_basics/quadrature_encoder/). Enkoder kwadraturowy.
- [34] Luca Peretti Mauro Zigliotto Roberto Petrella Marco Tursini. *Speed Measurement Algorithms for Low-Resolution Incremental Encoder Equipped Drives: a Comparative Analysis*. IEEE, 2007.
- [35] M. Iqbal Saripan Mohammad O. A. Aqel Mohammad H. Marhaban and Napsiah Bt. Ismail. *Review of visual odometry: types, approaches, challenges, and applications*. Gaza, Palestine, 2016.
- [36] James L. Frederic Chenavier. *Position Estimation for a Mobile Robot Using Vision and Odometry*. 1992.
- [37] H. R. Everett J. Borenstein and L. Feng. *Where am I? Sensors and Methods for Mobile Robot Positioning*. 1996.
- [38] [github.com/kriswiner/MPU9250.git](https://github.com/kriswiner/MPU9250.git). Biblioteka jednostki inercyjnej MPU-9250.
- [39] [github.com/ros-teleop/teleop\\_twist\\_keyboard.git](https://github.com/ros-teleop/teleop_twist_keyboard.git). Biblioteka do zadawania prędkości poprzez klawiaturę z wykorzystaniem komunikacji między wątkowej.

- [40] *github.com/ros-drivers/joystick\_drivers.git*. Biblioteka do zadawanie prędkości poprzez joystick z wykorzystaniem komunikacji między wątkowej.
- [41] *github.com/IntelRealSense/realsense-ros.git*. Biblioteka do obsługi systemu wizyjnego.
- [42] *github.com/robopeak/rplidar\_ros.git*. Biblioteka do obsługi lidaru.