Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Instytut Automatyki i Informatyki Stosowanej

Praca dyplomowa inżynierska

na kierunku Automatyka i Robotyka

System autonomicznej nawigacji dla terenowego robota mobilnego

Marcel Zawadzki
Numer albumu 310361

promotor
dr inż. Wojciech Dudek

WARSZAWA 2024

System autonomicznej nawigacji dla terenowego robota mobilnego

Streszczenie. Niniejsza praca skupia się na problemie autonomicznej nawigacji w nieusystematyzowanych terenach. Jej celem było opracowanie i implementacja systemu nawigacji autonomicznej dla terenowego robota mobilnego własnej konstrukcji, z wykorzystaniem skanera LiDAR.

Projekt obejmował opracowanie trójwymiarowego skanera LiDAR, rozbudowę platformy mobilnej o dodatkowe czujniki, utworzenie sterowników niskopoziomowych dla skanera i platformy mobilnej, opracowanie sterowników sprzętu w systemie ROS2 i utworzenie systemu nawigacji autonomicznej w ROS2. System ten jest zdolny do wykonywania kluczowych aspektów nawigacji, takich jak lokalizacja robota, mapowanie otoczenia, planowanie ścieżki i jej realizacja.

W ramach badań przeprowadzono szczegółowe testy poszczególnych modułów systemu, a następnie zweryfikowano skuteczność całego systemu w zadaniu nawigacji autonomicznej. Wyniki testów potwierdziły efektywność oraz poprawność działania systemu, wskazując na jego zdolność do skutecznego poruszania się w naturalnych, nieusystematyzowanych środowiskach.

Słowa kluczowe: robotyka, nawigacja autonomiczna, robot mobilny, łazik, LiDAR, ROS

Autonomous navigation system for the outdoor mobile robot

Abstract. This thesis addresses the problem of autonomous navigation in unstructured environments. The goal was to develop and implement an autonomous navigation system for an all-terrain mobile robot made by the author, utilising a LiDAR scanner.

The project involved the development of a 3D LiDAR scanner, an expansion of the mobile platform's capabilities with additional sensors, the creation of low-level controllers for the scanner and mobile platform, the development of hardware drivers in ROS2, and the creation of an autonomous navigation system in ROS2. This system is capable of executing the key aspects of navigation, such as robot localization, environment mapping, path planning and path execution.

As part of the research, detailed tests were conducted on individual modules of the system. Afterwards, the effectiveness of the entire system in the task of autonomous navigation was verified. The test results confirmed the effectiveness of the system, proving its ability to successfully navigate in natural, unstructured environments.

Keywords: robotics, autonomous navigation, mobile robot, rover, LiDAR, ROS

Spis treści

1. Wstęp

Współczesna robotyka stanowi obszar nasilonych badań i rozwoju, który dynamicznie kształtuje krajobraz technologiczny. Dziedzina ta łączy obszary inżynierii mechanicznej, elektronicznej i oprogramowania pozwalając inżynierom na tworzenie robotów, zaawansowanych systemów zdolnych do interakcji ze światem rzeczywistym, których głównym celem jest wspomaganie człowieka w niektórych czynnościach, lub zastępowanie go [1] - szczególnie gdy automatyzowana czynność wiąże się z niebezpieczeństwem dla wykonawcy lub cechuje się powtarzalnością i monotonią, mogącą doprowadzić do spadku dokładności wykonywanego przez człowieka zadania.

Roboty mobilne, w szczególności, stają się istotnym punktem zainteresowania ze względu na swoją zdolność do poruszania się w przestrzeni i wykonywania różnorodnych zadań, poza określonym stanowiskiem instalacji. Umożliwiają większą elastyczność wykonywania zadań niż tradycyjne rozwiązania automatyczne, jednocześnie nie wymagając zastosowania ludzkiej siły roboczej.

Najpopularniejszą podkategorią robotów mobilnych są roboty przeznaczone do użytku w przestrzeniach zamkniętych. Stanowią rozwiązania łatwiejsze w implementacji, gdyż pozwalają na przyjęcie wielu założeń upraszczających projekt robota, m.in:

- ograniczony poślizg kół,
- brak przechyłu robota,
- analizę przeszkód jedynie dwóch wymiarach,
- wykorzystanie infrastruktury np. przygotowanych punktów orientacyjnych lub linii magnetycznych.

W przeciwieństwie do robotów mobilnych przeznaczonych do przestrzeni zamkniętych, roboty terenowe stawiają przed inżynierami dodatkowe wyzwania. Muszą podołać niebezpiecznym warunkom terenowym, zmiennym nachyleniom trenu i przeszkodom trójwymiarowym, co wymaga bardziej zaawansowanych systemów nawigacyjnych i lepszej adaptacji do zmiennych warunków otoczenia. Dzięki temu ich zastosowania mogą być znacznie szersze. Cechują się zdolnością do operowania w nieustrukturyzowanych środowiskach, takich jak krajobrazy naturalne i rolnicze oraz obszary po kataklizmach. Umożliwia to wykorzystanie ich w zadaniach jak m.in.:

- eksploracji trudno dostępnych i niebezpiecznych terenów np. obszarów skażonych,
- zadaniach inspekcyjnych infrastruktury np. energetycznej i gazowej,
- zastosowaniach militarnych np. patrolowych i rozbrajaniu ładunków wybuchowych,
- branży rolniczej np. monitorowania rozległych upraw.

Szczególną odmianą pierwszego z wymienionych zastosowań jest eksploracja kosmiczna Księżyca i Marsa, będąca pierwotnym zastosowaniem terenowych robotów mobilnych - łazików kosmicznych. Łaziki umożliwiają przeprowadzanie badań bez zagrożenia życia ludzkiego i konieczności transportu żywności, wody, czy tworzenia systemów podtrzymania życia.

Powyższe zastosowania wymagałyby od człowieka narażenia na niebezpieczeństwo lub wykonywania powtarzalnych czynności, są więc dobrze dostosowane do wykonywania przez robota w sposób autonomiczny. W przypadku aplikacji pozaziemskich, dodatkowym argumentem za automatyzacją jest znaczące opóźnienie w komunikacji - w przypadku transmisji pomiędzy Ziemią a Marsem wynosi ono od 5 do 20 minut[2], co znacząco utrudnia bezpośrednie sterowanie.

W niniejszej pracy omówiona zostanie problematyka związana z projektowaniem i implementacją autonomicznego systemu nawigacji dla robota terenowego. W tym kontekście analizowane będą różne aspekty, takie jak platforma mobilna, systemy sensoryczne, a także algorytmy nawigacyjne.

2. Cel pracy

Celem niniejszej pracy było stworzenie autonomicznego systemu dla robota mobilnego własnej konstrukcji, zdolnego do nawigacji w nieustrukturyzowanych środowiskach. Zastosowana platforma mobilna technicznie umożliwia poruszanie się po nieregularnym terenie. Opracowany system posiada zdolność lokalizacji robota, mapowania otoczenia, wyznaczania trasy do określonego celu, a następnie skutecznego jej pokonywania, uwzględniając omijanie przeszkód. W ramach projektu opracowano trójwymiarowy skaner LiDAR jako kluczowe narzędzie do percepcji otoczenia przez robota. Dodatkowo, przygotowano sterowniki niskopoziomowe, umożliwiające zbieranie danych oraz skuteczne sterowanie ruchem robota.

2.1. Założenia projektowe

Na podstawie analizy dziedziny i istniejących rozwiązań, które zostaną szczegółowo przeanalizowane w rozdziałach 3 i 4, przyjęte zostały następujące założenia projektowe:

- Nawigacja autonomiczna - system zdolny będzie do wykonywania zadania nawigacji autonomicznej zgodnie z definicją w sekcji 3.5.
- Dane wejściowe do systemu - współrzędne celu
- Wykorzystanie autorskiej terenowej platformy mobilnej, opisanej w sekcji 3.2. Oferuje ona bardzo dobrą mobilność i jest przystosowana do poruszania się po nieregularnym terenie, oraz pozwala na instalację szerokiego zakresu czujników.
- Skonstruowanie trójwymiarowego skanera LiDAR. Z przyczyn przytoczonych w sekcji 3.3 jest odpowiednim źródłem danych w robocie terenowym, jednocześnie mieszcząc się w budżecie autora. Ponadto, konstrukcja własnego czujnika pozwoli na zbadanie wpływu parametrów skanera na działanie systemu.
- Wykorzystanie mikrokontrolerów ATmega i języka C++ do sterowania niskopoziomową częścią systemu. W ramach niniejszej pracy oprogramowanie zostało poszerzone o obsługę dodatkowych sensorów oraz możliwość komunikacji z komputerem obsługującym funkcje wysokopoziomowe.
- Zastosowanie struktury ramowej ROS2 w wersji Humble [3] przeznaczonej na system Ubuntu 22.04 LTS do rozwijania oprogramowania wysokopoziomowego, z powodów przedstawionych w sekcji 3.6. Wybrana została najnowsza wersja z długim terminem wsparcia. Ponadto ROS w wersji 2 oferuje zwiększone bezpieczeństwo i stabilność. Do programowania pojedynczych węzłów użyto język Python, ze względu na szybkość tworzenia kodu i dostępności obszernej gamy bibliotek.
- Do tworzenia schematów blokowych SysML zgodnie z metamodelem MeROS [4] wykorzystano Visual Paradigm.
- Pozostałe wykorzystane oprogramowanie: IDE VS Code, MatLab R2021a do analizy danych, Solidworks 2021 do modelowania 3D, CURA 5.0 do przygotowania modeli do druku 3D.

2.2. Ograniczenia

Ponadto, przyjęte zostały następujące ograniczenia zakresu pracy:

- Podczas wykonywania zadania nawigacji ignorowane będą dynamiczne przeszkody. Robot z założenia poruszać ma się po terenach niezamieszkanych, ponadto jego niska prędkość ogranicza wpływ poruszających się z dużą prędkością obiektów na jego ścieżkę.
- Możliwości porównania wyników z innymi rozwiązaniami są ograniczone, ze względu na brak dostępu do standaryzowanych warunków testowych i duże różnice w konstrukcjach robotów.
- W czasie nawigacji ignorowane są zbiorniki wodne. Woda jest przezroczysta i odblaskowa, co uniemożliwia poprawne jej wykrycie przez czujnik LiDAR. Konieczne byłoby zastosowanie dodatkowych czujników np. ultradźwiękowych, których wiązka odbija się od wody.

2.3. Walidacja działania systemu

Skuteczność rozwiązania przetestowana została podczas testów terenowych całości systemu oraz testów działania odseparowanych podsystemów. Działanie skanera LiDAR zostało przetestowane w szerokim zakresie warunków otoczenia i porównane z komercyjnym sensorem. Sterowniki niskopoziomowe testowane były w odseparowaniu od warstwy wysokopoziomowej. Algorytmy autonomiczne przetestowano w pomieszczeniu w pobliżu stanowiska roboczego, w celu ułatwienia modyfikacji oprogramowania na bieżąco. Dalsze testy przeprowadzono w warunkach naturalnych, w nierównym terenie z przeszkodami, najczęściej drzewami lub dodatkowymi przeszkodami umieszczonymi ręcznie w środowisku. W celu walidacji działania kompletnego systemu, przeprowadzono testy w naturalnym, nieregularnym środowisku z przeszkodami. Zadano robotowi cel, następnie obserwowano realizację zadania. Przejazd zarejestrowano z wykorzystaniem zewnętrznej kamery, oraz zapisano wizualizację stanu systemu.

3. Narzędzie i technologie

3.1. Roboty mobilne

Robot mobilny jest to robot zdolny do zmiany położenia w przestrzeni, jeżdżąc, krocząc, latając lub pływając. Poniższa analiza ograniczona będzie do robotów lądowych. Podstawą robota jest napęd. W robotach terenowych najczęściej wykorzystywane są napędy:

- napęd różnicowy - podstawowo obejmujący dwa współosiowe koła o regulowanej prędkości. W takiej konfiguracji powszechny w robotach zaprojektowanych do poruszania się po płaskim terenie. W robotach terenowych na ogół poszerzony o dodatkowe nieskrętne koła, lub wyposażony w gąsienice zamiast kół. Dodatkowe koła lub gąsienice powodują poślizg podczas skręcania, ale ułatwiają jazdę po nierównym terenie.
- napęd holonomiczny (ang. omnidirectional) - robot wyposażony w wiele kół skrętnych, lub inne, niestandardowe rozwiązanie. Robot holonomiczny z definicji może mieć zadany dowolny wektor prędkości. Wadą takiego rozwiązania jest wysokie skomplikowanie konstrukcji.
- robot kroczący - ruch robota odbywa się poprzez poruszanie kończynami skonstruowanymi na wzór organizmów żywych. Wyposażone w dużą liczbę stopni swobody, skomplikowane mechaniczne i wymagające do sterowania skomplikowanych algorytmów.

Dany rodzaj napędu determinuje konstrukcję robota. Przy robotach kołowych zalecane jest wykorzystanie mechanizmu umożliwiającego utrzymanie kontaktu kół z nierównym podłożem, czyli zawieszenia. Zastosowanie takiego mechanizmu poprawia zdolność robota do pokonywania przeszkód, zwiększa stabilność robota, zmniejsza poślizg kół i zapewnia równomierne obciążenie napędów. Typ zawieszenia zależny jest od liczby kół, powszechnie wykorzystywane jest np. zawieszenie typu rocker-bogie [5], wykorzystane m.in. w łaziku NASA Perserverence [2] (rys. 3.1) lub zawieszenie wahaczowe.

Ponadto, aby robot mobilny był użyteczny, powinien dysponować autonomią energetyczną, co oznacza, że powinien posiadać zintegrowane źródło zasilania. Takie źródło może obejmować akumulatory, ogniwa paliwowe lub reaktor jądrowy (w przypadku łazików kosmicznych). Kluczowe jest również wyposażenie robota w układy niezbędne do sterowania napędami i innymi elementami mechanicznymi. Co więcej, w kontekście współczesnych systemów robotycznych, istotnym aspektem jest również brak konieczności wykorzystania przewodów do komunikacji. W przypadku robotów autonomicznych wystarczające jest otrzymanie komend z celem i przesyłanie telemetrii drogą radiową.

3.2. Platforma mobilna

W niniejszej pracy wykorzystano platformę mobilną własnej konstrukcji (rys. 3.2), zbudowaną we wcześniejszym czasie. Poniżej przedstawiona została jej specyfikacja.

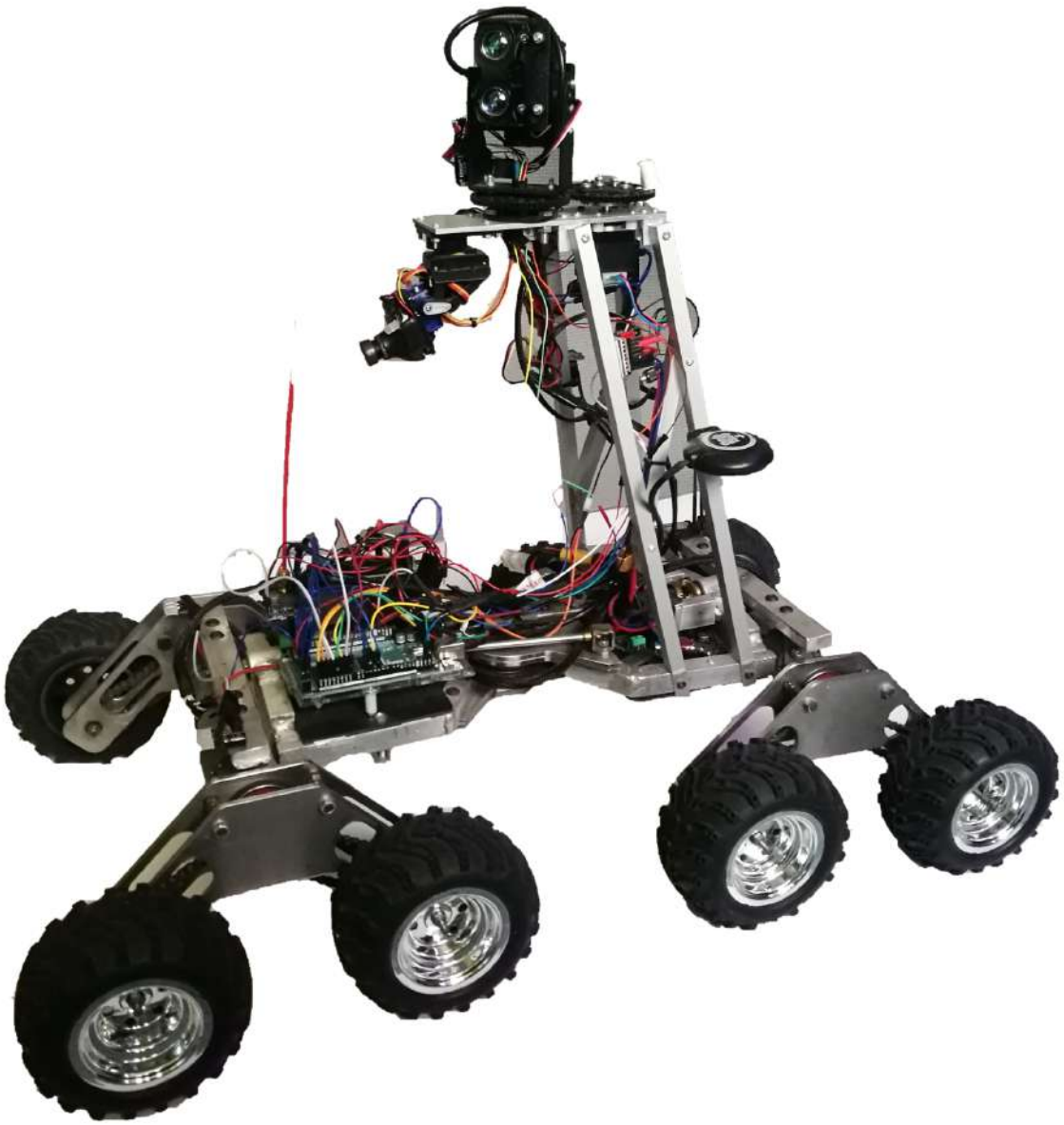
Rysunek 3.1. Zawieszenie łazika Perserverence [2]

Robot jest wyposażony w niestandardowy rodzaj napędu, składający się z 8 napędzanych kół oraz centralnie umieszczonego przegubu. Koła napędzane są silnikami prądu stałego (po 2 koła na pojedynczy silnik). Wykorzystane zostały silniki szczotkowe Igarashi 2738-125 [6] wraz z przekładnią 50:1. Do wychylenia przegubu wykorzystywany jest serwonapęd PowerHD LF-20MG [7] z systemem cięgien.

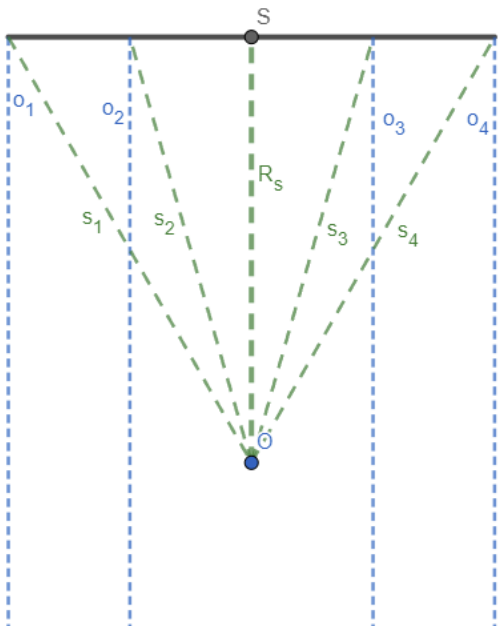
Napęd w istocie stanowi rozszerzenie napędu różnicowego - sterując jedynie prędkością kół z prawej i lewej strony sterowanie odbywałoby się na zasadzie standardowego napędu różnicowego. Dodany przegub poprawia właściwości jezdne platformy, redukując poślizg kół podczas skręcania. Poślizg (w obrocie robota wokół osi pionowej) nie zachodzi, gdy oś koła przechodzi przez środek rotacji robota. Przyjmując obrót wokół punktu O, z przegubem w punkcie S odpowiednie wychylenie przegubu powoduje zmniejszenie kąta odchylenia kierunku osi kół (o_1, o_2, o_3, o_4) od kierunku promienia skrętu (Promień skrętu robota R_s, promienie dla poszczególnych kół odpowiednio o_1, o_2, o_3, o_4). (Rys. 3.3, 3.4). Szczegółowe równania ruchu układu napędowego zostaną wyprowadzone w dalszej części pracy. Napęd ma taką samą swobodę ruchu jak napęd różnicowy. Nie jest holonomiczny, może realizować trajektorie o niezerowej prędkości liniowej w osi x oraz prędkości kątowej wokół osi z w układzie odniesienia robota [1].

Zastosowanie przegubu, poza ograniczeniem poślizgu skutkuje zwiększeniem precyzji pokonywania zakrętów i zmniejsza obciążenie silników podczas skręcania. Należy jednak zaznaczyć, że kąt wychylenia przegubu jest ograniczony. W przypadku obrotów o zbyt małym promieniu skrętu, na przykład podczas skręcania w miejscu, gdy środek obrotu znajduje się pomiędzy prawymi i lewymi kołami, przegub nie może być efektywnie

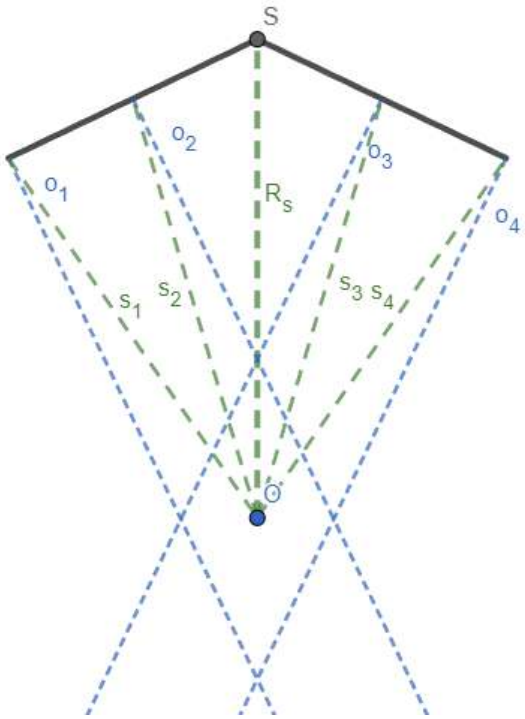
[1] Konwencja nazewnictwa osi w tej i kolejnych wzmiankach, to oś x skierowana w przód robota, oś z w górę



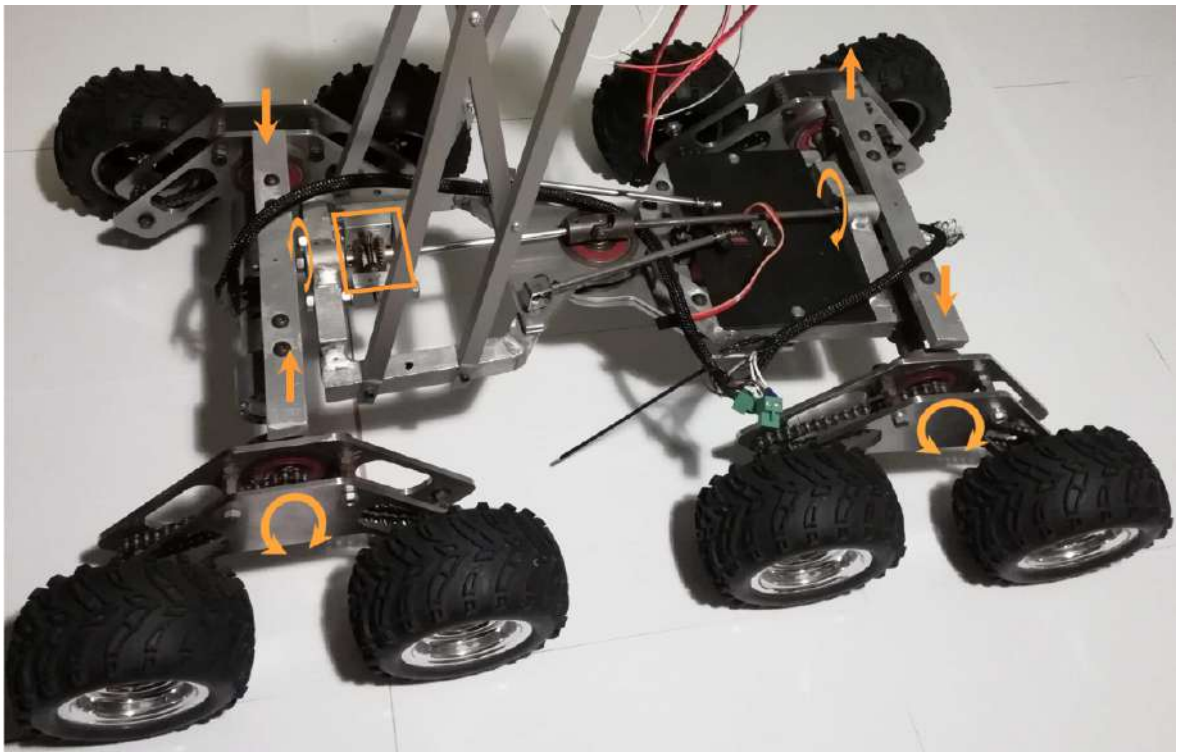
Rysunek 3.2. Platforma mobilna



Rysunek 3.3. Skręt w napędzie różnicowym



Rysunek 3.4. Skręt z dodatkowym przegubem



Rysunek 3.5. Zawieszenie platformy mobilnej

wykorzystywany. W takich sytuacjach robot traktowany jest jako platforma z napędem różnicowym.

Kolejnym istotnym komponentem jest zawieszenie łazika (Rys. 3.5). Składa się ono z dwóch poziomych kolebek, umieszczonych z przodu i z tyłu pojazdu. Na końcach każdej kolebki znajduje się wózek, który obraca się wokół osi kolebki. Każdy z tych wózków jest wyposażony w dwa koła, napędzane jednym silnikiem. Napęd ten jest przenoszony poprzez łożysko umieszczone między kolebką a wózkiem, a następnie za pomocą łańcucha wewnątrz wózka. Dodatkowo, przez środek łazika przechodzi wał, który łączy obie kolebki. Wewnątrz kadłuba znajduje się przekładnia odwracająca kierunek obrotu wału. W rezultacie, kąt wychylenia kadłuba jest średnią z kątów wychyleń obu kolebek. Ten złożony system zawieszenia umożliwia łazikowi utrzymanie stabilności na nierównym terenie, dostosowując kąt pochylenia kadłuba do warunków terenowych.

Robot zasilany jest dwiema bateriami litowo-polimerowymi połączonymi równolegle, o łącznej pojemności 3,6Ah i napięciu nominalnym 18,5V.

W rezultacie robot posiada następującą specyfikację:

- maksymalna prędkość liniowa: $0,230$ m/s
- maksymalna prędkość kątowa: $1,34$ rad/s
- maksymalne wychylenie przegubu: $20,1°$
- wymiary podwozia: 660mm długości x 410mm szerokości x 160mm wysokości (520mm z wieżą sensoryczno-komunikacyjną)

Ponadto, robot jest wyposażony w wieżę sensoryczno-komunikacyjną, która umożliwia

umieszczenie sensorów z dala od potencjalnych źródeł zakłóceń, zwiększa widoczność anten w przypadku nierównego terenu, oraz pozwala na montaż skanerów lub kamer na większej wysokości, co z kolei zwiększa ich efektywność i pole widzenia.

Funkcje niskopoziomowe platformy sterowane są za pomocą mikrokontrolera ATmega2560. Obejmują sterowanie silnikami, odczyt danych z czujników oraz zarządzanie komunikacją z urządzeniami sterującymi. Możliwości komunikacyjne samej platformy obejmują dwustronną łączność radiową w częstotliwości 433 MHz ze stacją bazową (obsługującą sterowanie i telemetrię), a także przesyłanie obrazu z kamery analogowej na głowicy typu Pan-Tilt (ang., obrót i pochylenie) w częstotliwości 5.8 GHz. Te funkcje umożliwiają operatorowi na ręczne sterowanie platformą, co było pierwotnym przeznaczeniem platformy.

3.3. Czujniki wizyjne

Aby możliwa była automatyzacja robota, kluczowa jest jego zdolność do efektywnego pozyskiwania informacji o otoczeniu. Porównane zostanie kilka popularnych technologii służących do pozyskiwania danych z otoczenia.

LiDAR

LiDAR (ang. Light Detection and Ranging) to technologia wykorzystywana do mierzenia odległości poprzez pomiar czasu, jaki wysłana przez emiter wiązka laserowa potrzebuje na powrót do sensora po odbiciu od obiektu. Proces ten umożliwia precyzyjne określenie odległości między czujnikiem a otaczającymi obiektami. LiDAR charakteryzuje się dużą szybkością pojedynczego odczytu, co sprawia, że jest efektywnym narzędziem do pomiarów w dynamicznych środowiskach. Dodatkowo, technologia ta cechuje się wysoką dokładnością, sięgającą poniżej 1 cm, oraz dużym zasięgiem działania do 300 metrów [8].

Skaner LiDAR 2D

Skaner LiDAR 2D składa się z pojedynczego sensora z mechanizmem obrotowym lub wielu sensorów (przykładowy sensor na rys. 3.6). Dostarcza informacji o odległości do obiektów znajdujących się w płaszczyźnie sensora, co sprawia, że jest dedykowany głównie do regularnych pomieszczeń, gdzie większość przeszkód ma niewielkie zmiany w pionie. Dodatkowo, w przypadku przechylenia robota względem płaszczyzny poziomej, pomiar może ulec zaburzeniu. W rezultacie, skaner LiDAR 2D nie sprawdza się dobrze w przypadku nieusystematyzowanego terenu.

Kamera głębi

Kamera głębi opiera się na technologii stereowizji, wykorezystując 2 kamery RGB (przykładowy sensor na rys. 3.7). Jej działanie jest wrażliwe na warunki oświetleniowe [10], co może stanowić utrudnienie w otwartych przestrzeniach. Dodatkowo, może napotykać trudności przy interpretacji nieregularnych, naturalnych obiektów takich jak roślinność czy skały, które znacząco zmieniają wygląd zależnie od padania światła i niewielkich przesunięć.

Rysunek 3.6. Skaner 2D RPLIDAR A3M1 [9]

Rysunek 3.7. Kamera głębi StereoLabs ZED 2 [11]

Skaner LiDAR 3D

Skaner LiDAR 3D składa się z pojedynczego lub wielu sensorów z mechanizmem obrotowym w jednej lub wielu osiach (przykładowy sensor na rys. 3.8). Skaner LiDAR 3D generuje chmurę punktów reprezentującą całe otoczenie, co czyni go odpowiednim do pozyskiwania danych z nieusystematyzowanego terenu. Wadą rozwiązania jest wysoki koszt, przekraczający 10 000 zł.

Rysunek 3.8. Skaner 3D Artec Ray II [12]

Własny skaner LiDAR 3D

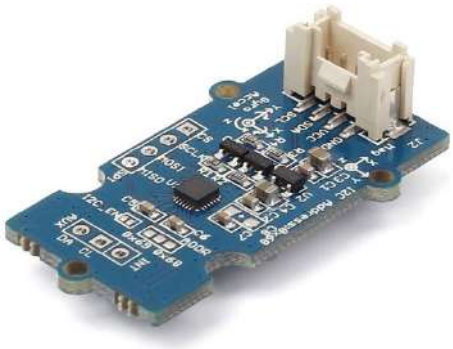
Alternatywą jest zbudowanie własnego skanera LiDAR 3D, stworzonego przy użyciu czujnika 1D. Rozwiązanie to posiada zalety charakterystyczne dla skanerów 3D, ale cechuje się znacznie niższym kosztem, który może być jednak skorelowany z potencjalnie niższą szybkością i rozdzielczością w porównaniu do urządzeń komercyjnych.

3.4. Pozostałe czujniki

W robotach mobilnych powszechnie wykorzystywane są również inne rodzaje czujników:

IMU

IMU (ang. Inertial measurement unit), czyli jednostka pomiarowa inercji, to moduł zawierający akcelerometr, żyroskop i magnetometr. Akcelerometr mierzy przyspieszenie liniowe, żyroskop rejestruje prędkość kątową, a magnetometr dostarcza informacje o wektorze pola magnetycznego. Uzyskiwany jest wektor 9 wartości. Dzięki tym danym IMU umożliwia śledzenie ruchu i orientacji obiektu. Przykładowy czujnik przedstawiono na rys. 3.9.



Rysunek 3.9. Czujnik Grove IMU 9DOF v2.0 MPU-9250 [13]

GNSS

GNSS (ang. Global Navigation Satellite System) to system obejmujący różne grupy satelitów umieszczonych na orbicie, takie jak najbardziej znany GPS, oraz GLONASS, Galileo czy BeiDou. Czujnik GNSS, wykorzystując sygnały z tych satelitów, dostarcza informacje o lokalizacji - współrzędne geograficzne (długość, szerokość, wysokość). Te informacje pozwalają na określenie globalnej lokalizacji. Przykładowy odbiornik przedstawiono na rys. 3.10.

Rysunek 3.10. Odbiornik GNSS Ublox NEO-M8N [14]

3.5. Autonomiczna nawigacja

Zadanie nawigacji autonomicznej odnosi się do zdolności do autonomii decyzyjnej robota. Obejmuje kompleksowy zestaw operacji, mających na celu umożliwienie robotowi skutecznego poruszania się w nieznanym środowisku. Kluczowymi aspektami tego zadania są:

- Lokalizacja robota: Określenie dokładnego położenia robota w przestrzeni, w pewnym globalnym układzie odniesienia.
- Mapowanie otoczenia: Tworzenie reprezentacji otoczenia robota poprzez zbieranie informacji o strukturze przestrzeni wokół niego. Mapowanie może obejmować zarówno elementy stałe, jak i zmienne, takie jak przeszkody, ściany, drzewa i skały.
- Wyznaczanie ścieżki do celu: Opracowanie optymalnej trasy prowadzącej z aktualnej lokalizacji robota do, określonego przez operatora lub algorytm, celu. Algorytmy planowania trasy uwzględniają trudność ścieżki i znajdujące się na niej przeszkody.
- Przejechanie ścieżki omijając przeszkody: Realizacja zaplanowanej trasy przez robota, przy jednoczesnym unikaniu przeszkód oraz dostosowywaniu się do zmieniającego się otoczenia.

Wszystkie te operacje mają być wykonane autonomicznie, bez konieczności ingerencji ze strony operatora. To oznacza, że robot musi być wyposażony w zaawansowane systemy percepcji oraz oprogramowanie zdolne do planowania i sterowania, umożliwiające mu samodzielne funkcjonowanie w różnorodnych warunkach środowiskowych.

3.6. Robot Operating System

Najpopularniejszym rozwiązaniem do tworzenia oprogramowania robotów jest ROS [3] (ang. Robot Operating System). ROS to oprogramowanie typu open source stanowące strukturę ramową dedykowaną tworzeniu systemów robotycznych. System ten oferuje zestaw narzędzi i bibliotek, które ułatwiają programowanie, kontrolę i interakcję między różnymi komponentami. Jedną z kluczowych cech ROS jest modularność, która umożliwia łatwe łączenie programów w jeden konfigurowalny system. Komunikacja między różnymi komponentami systemu opiera się na mechanizmach tematów i węzłów.

Węzły (Nodes): Węzły to autonomiczne jednostki wykonawcze w systemie ROS, które realizują konkretne zadania. Każdy węzeł może zawierać logikę programu, kontrolować konkretne elementy robota lub zbierać dane z sensorów. Węzły komunikują się ze sobą poprzez tematy. Różne węzły mogą być zaimplementowane w różnych językach programowania, co pozwala na wykorzystanie niskopoziomowych języków w węzłach krytycznych dla wydajności systemu.

Tematy (Topics): Tematy są jednym z podstawowych mechanizmów komunikacyjnych w ROS. Są to nazwane strumienie danych, które pozwalają jednemu węzłowi publikować informacje (dane sensoryczne, dane stanu, itp.), a inne węzły mogą się na nie subskrybować.

Tematy umożliwiają bezpośrednią komunikację między węzłami poprzez przekazywanie komunikatów w czasie rzeczywistym.

Mechanizm ten umożliwia elastyczną i zdecentralizowaną architekturę systemu, gdzie różne węzły, odpowiedzialne za różne aspekty funkcjonalne robota, mogą współpracować poprzez publikowanie i subskrybowanie danych.

ROS obsługuje szeroki zakres zastosowań w robotyce mobilnej, manipulacyjnej, sensorycznej i innych dziedzinach. Dzięki bogatej społeczności użytkowników i aktywności deweloperskiej, ROS stanowi preferowane narzędzie wśród badaczy i inżynierów.

3.7. Lokalizacja

Lokalizacja w kontekście robotyki często wykorzystuje dwa kluczowe układy odniesienia: układ odometrii (odom) i układ mapy (map). Lokalizację w układzie odom stanowi lokalizację lokalną, zaś w układzie map globalną.

Układ odometrii opiera się na pomiarach przemieszczenia robota, wykorzystując dane z ruchu kół, jednostki inercyjnej, lub odometrii wizualnej. Jest to lokalny układ odniesienia, który dostarcza informacji o ruchu w czasie rzeczywistym, w sposób ciągły. Jest jednak podatny na kumulację błędów.

Z drugiej strony, układ mapy reprezentuje globalne środowisko, w którym porusza się robot. Jest to reprezentacja przestrzeni zawierająca informacje o przeszkodach, kształcie terenu itp. Układ mapy umożliwia robotowi orientację w globalnej skali, co jest istotne dla nawigacji w dużych środowiskach. Układ globalny pozwala na wyeliminowanie kumulowanego błędu odometrii.

3.8. SLAM

SLAM (ang. Simultaneous Localization and Mapping) to algorytm, który jednocześnie lokalizuje obiekt (np. robota) w przestrzeni oraz aktualizuje na bieżąco mapę otoczenia. Algorytmy SLAM integrują dane pochodzące z sensorów wizyjnych, takich jak kamery i skanery, w czasie rzeczywistym. Algorytm może działać zarówno na mapach otoczenia i źródłach danych dwu- i trójwymiarowych. SLAM dopasowuje najnowszy skan lub obraz do przechowywanej w pamięci mapy - uzyskane położenie nowego skanu stanowi nową wyznaczoną lokalizację. Następnie nowy skan jest wykorzystywany do aktualizacji mapy. Dodatkowo, wraz z algorytmem SLAM często stosuje się algorytmy domykające pętlę (ang. loop closure), które korygują całość mapy, gdy wykryją, że znane już miejsce zostało ponownie odwiedzone. Zapewnia to zachowanie ciągłości mapy, i jednoznaczność odwzorowania rzeczywistych lokalizacji na mapie.

3.9. Filtr komplementarny

Filtr komplementarny działa na zasadzie łączenia informacji z sensorów w IMU. Bazuje na dokładnych danych z żyroskopu, eliminując przy tym dryf tego sensora za pomocą

informacji z akcelerometru i magnetometru. Te ostatnie, choć charakteryzują się mniejszą dokładnością, nie posiadają dryfu. Dzięki takiemu podejściu, filtr komplementarny eliminuje dryf pomiarowy i zapewnia stabilną orientację obiektu.

3.10. Filtr Kalmana

Filtr Kalmana to algorytm estymujący wektor stanu modelu na podstawie pomiarów wejścia i wyjścia. W robotyce jego głównym celem jest integracja informacji z różnych rodzajów sensorów, wykorzystując informacje o niepewności pomiarów dostarczane przez czujniki. W efekcie uzyskiwana jest najdokładniejsza możliwa ciągła estymata stanu systemu. Algorytm reprezentowany jest następującymi wzorami [15]:

Predykcja stanu:

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + Gu_k$$

Predykacja kowariancji:

$$P_{k|k-1} = FP_{k-1|k-1}F^T + Q$$

Wzmocnienie Kalmana:

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R_k)^{-1}$$

Aktualizacja stanu:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1})$$

Aktualizacja kowariancji:

$$P_{k|k} = (I - K_kH)P_{k|k-1}(I - K_kH)^T + K_kR_kK_k^T$$

Oznaczenia:

$\hat{x}_{k|k-1}$: Przewidywana wartość stanu w chwili k, obliczona w chwili $k-1$.

F : Macierz przejścia stanu.

$\hat{x}_{k-1|k-1}$: Estymowana wartość stanu w chwili $k-1$

G : Macierz sterowania.

u_k : Wejście (sterowanie) w chwili k.

$P_{k|k-1}$: Przewidywana kowariancja stanu w chwili k, obliczona w chwili $k-1$.

Q : Kowariancja szumu procesu.

K_k : Wzmocnienie Kalmana.

H : Macierz obserwacji.

R_k : Kowariancja szumu pomiarowego.

$\hat{x}_{k|k}$: Estymowana wartość stanu w chwili k

z_k : Wektor pomiarów w chwili k.

I : Macierz jednostkowa.

Istnieje również rozszerzony filtr Kalmana (ang. Extended Kalman Filter, EKF) - rozszerzony o przypadki, gdy model matematyczny systemu jest nieliniowy, umożliwiając skuteczną estymację stanu nieliniowych systemów dynamicznych.

3.11. MeROS

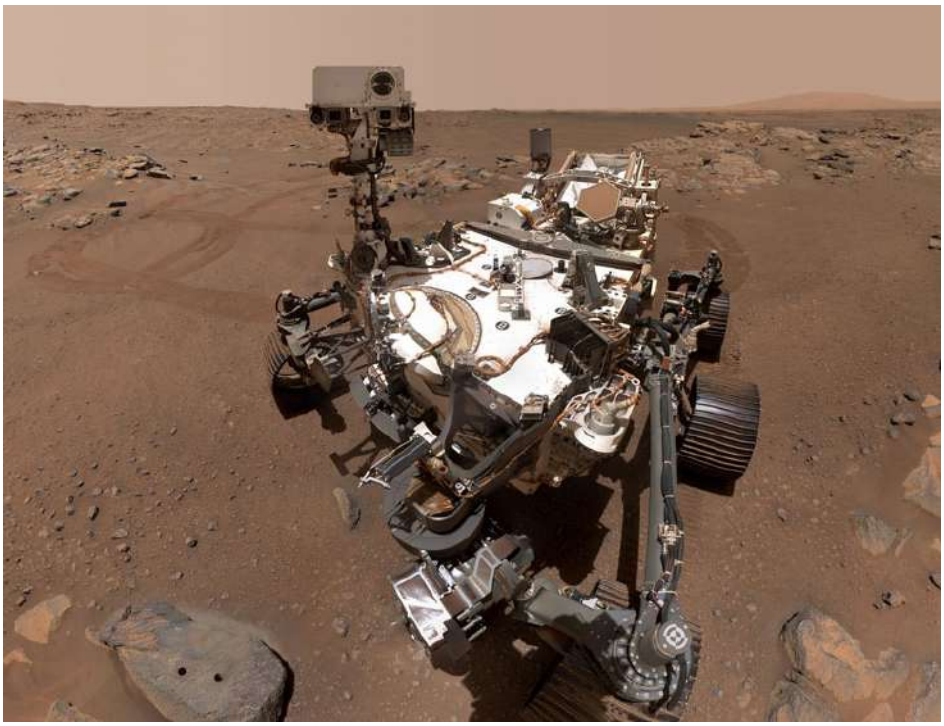
MeROS [4] to metamodel ułatwiający modelowanie systemów robotycznych bazujących na ROS1 i ROS2 z wykorzystaniem SysML (ang. Systems Modeling Language). Zapewnia ustandaryzowaną formę diagramów blokowych, ułatwiającą projektowanie i interpretację.

4. Przegląd rozwiązań

W niniejszym rozdziale przedstawiono przegląd różnych rozwiązań istniejących na rynku i prac naukowych, które konkurują lub stanowią alternatywę dla proponowanego systemu.

4.1. Łaziki marsjańskie

Pierwsze łaziki marsjańskie (Sojourner (1997), Spirit (2003), Opportunity (2003)) [16] nie wykorzystywały pełnej nawigacji autonomicznej. Roboty realizowały uproszczoną autonomię bazującą na wykonywaniu określonych zachowań, z reakcjami na zagrożenia. Stopniowo wprowadzano planowanie ruchów. Rozwiązania w pełni autonomicznej nawigacji zaczęto stosować w robotach Curiosity (2011) i Perseverance (2020) (rys. 4.1). Pozwoliło to na znaczące zmniejszenie czasu reakcji sterowania i zwiększyło bezpieczeństwo działania.



Rysunek 4.1. Łazik Perseverance [2]

4.2. Leo rover

Leo Rover [17] (rys. 4.2) to platforma mobilna dostępna na rynku, dedykowana zastosowaniom edukacyjnym. Stanowi gotowe rozwiązanie sprzętowe z możliwością samodzielnego programowania z wykorzystaniem ROS 1. Stanowi jednakże zamknięty produkt, z ograniczonymi możliwościami modyfikacji przez użytkownika, wykorzystuje określony zestaw sensorów (domyślnie wyposażony w kamerę). Ponadto cechuje się wysokim kosz-

Rysunek 4.2. Leo Rover [17]

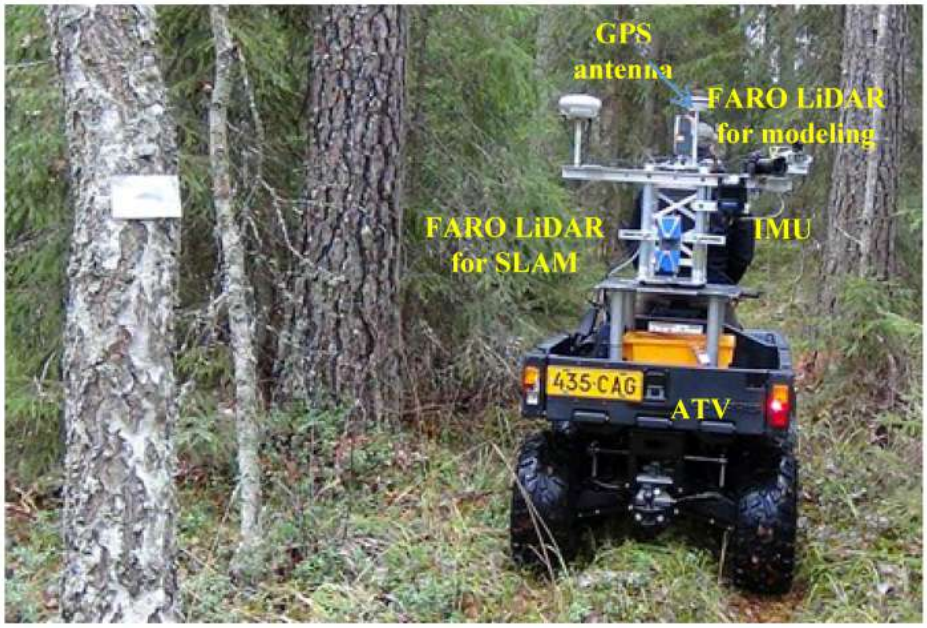
tem zakupu ok. 20000 zł. Stanowi pewną alternatywę dla proponowanego rozwiązania, w zakresie sprzętowym.

4.3. Mapowanie pni

W pracy "SLAM-Aided Stem Mapping for Forest Inventory with Small-Footprint Mobile LiDAR"[18] wykorzystano robota mobilnego (rys. 4.3) wyposażonego w LiDAR. System jest dostosowany do nawigacji w terenach leśnych. Wykorzystuje kształt pni do mapowania i lokalizacji - przy założeniu że pnie są pionowe, możliwe jest zwiększenie precyzji lokalizacji dzięki wykorzystaniu trójwymiarowego skanera. Jednakże, choć rozwiązanie jest efektywne w specyficznych warunkach leśnych, jego zastosowanie może być ograniczone ze względu na niewielką uniwersalność. Celem autora niniejszej pracy jest natomiast utworzenie rozwiązania do nawigacja w dowolnym terenie.

4.4. Nawigacja w nierównym terenie

W pracy "Autonomous mobile robot navigation in uneven and unstructured indoor environments"[19] wykorzystano robota mobilnego (rys. 4.4) wyposażonego w LiDAR o niskiej rozdzielczości. Robot nawiguje w nieusystematyzowanym środowisku, jest ono jednak uproszczone względem naturalnego - reprezentowane przez rampy i bryły. Pewnym ograniczeniem jest tu zastosowanie robota nie terenowego.



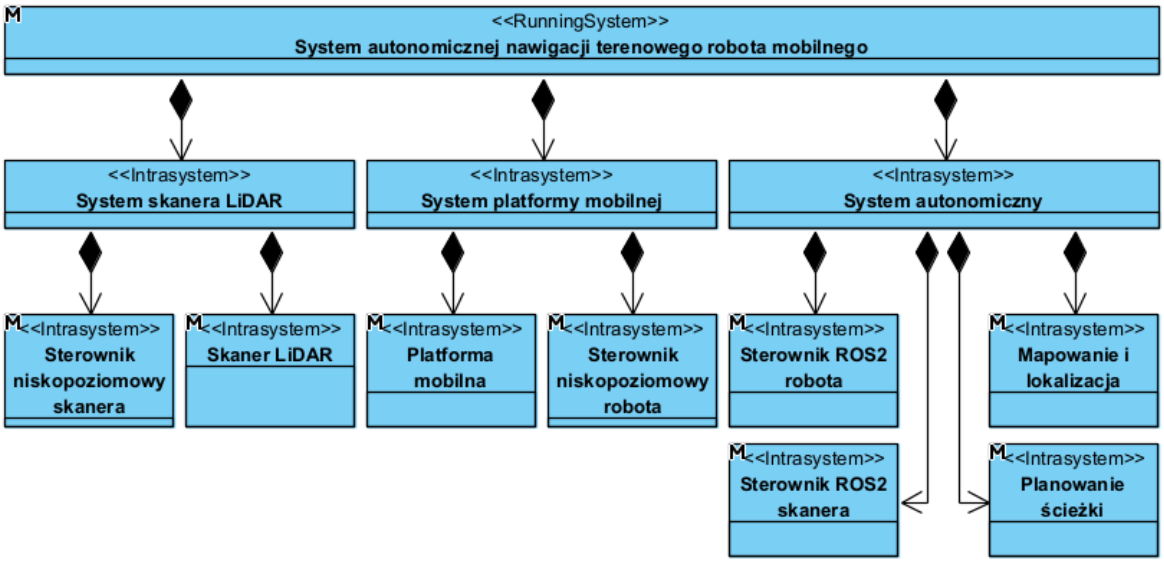
Rysunek 4.3. Robot wykorzystany do mapowania pni [18]

Rysunek 4.4. Robot wykorzystany do nawigacji w nierównym terenie [19]

5. Projekt systemu

5.1. Struktura systemu

Zaprojektowany system składa się z kilku kluczowych modułów, reprezentujących zarówno fizyczne urządzenia, jak i fragmenty oprogramowania. Strukturę systemu przedstawiono na rys. 5.1. Komunikację pomiędzy poszczególnymi modułami przedstawiono na rys. 5.2. Ponadto, system można podzielić na warstwę sprzętową, niskopoziomową i wysokopoziomową.



Rysunek 5.1. Proponowana kompozycja systemu

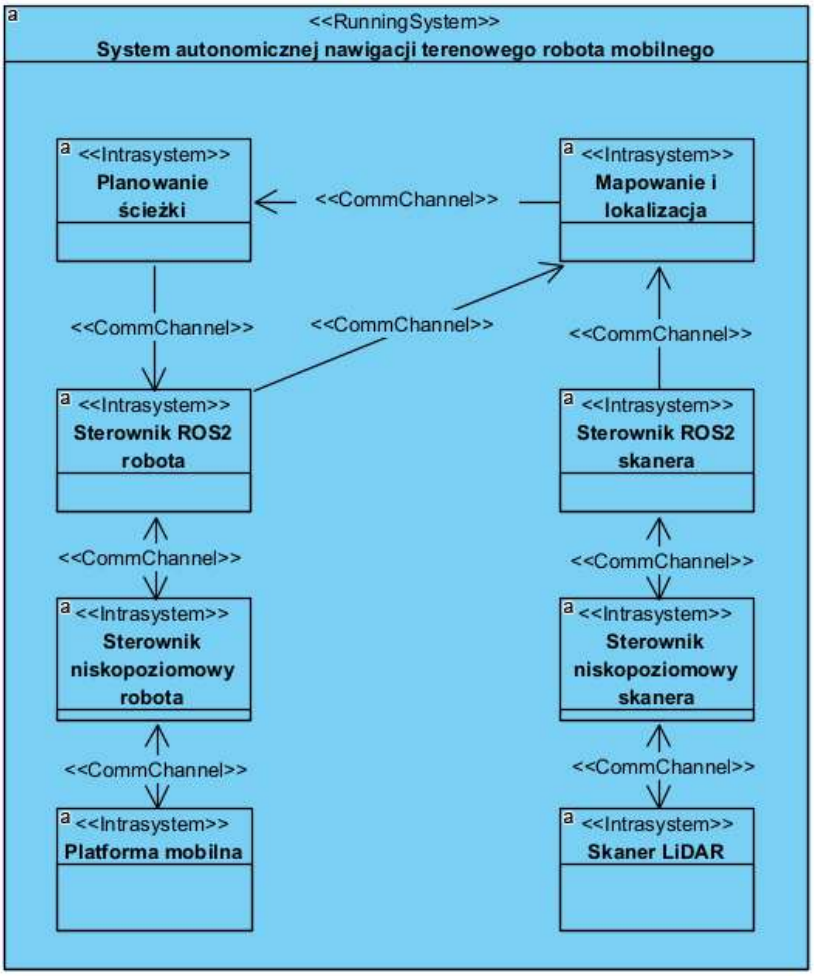
Warstwa sprzętowa to fizyczna część systemu, obejmująca wszystkie urządzenia, komponenty elektroniczne, czujniki, efektory. Składają się na nią:

- Fizyczna platforma mobilna, z efektorami (silniki napędowe) i zainstalowanymi czujnikami.
- Moduł skanera LiDAR, z efektorami (silniki poruszające mechanizmem) i czujnikiem odległości.

Warstwa niskopoziomowa to element systemu, który zajmuje się bezpośrednim sterowaniem sprzętem i realizacją podstawowych funkcji, często w kontekście interakcji z fizycznym środowiskiem. Składają się na nią:

- Sterownik robota, odpowiadający za kontrolę silników powodujących ruch robota i przesyłanie do warstwy wysokopoziomowej danych z czujników.
- Sterownik skanera, komponent zarządzający działaniem skanera LiDAR, kontrolujący ruch głowicy skanującej i przesyłający pomiary.

Warstwa wysokopoziomowa to fragment systemu, który obejmuje bardziej złożone funkcje i algorytmy, działające na wyższym poziomie abstrakcji. Jest odpowiedzialna za nawigację autonomiczną, gdzie jedyną informacją wymaganą od użytkownika są współ-



Rysunek 5.2. Proponowana struktura komunikacji

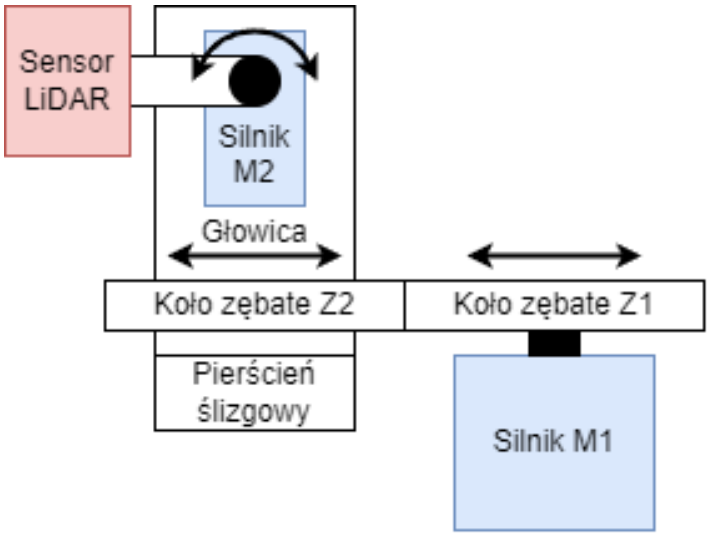
rzędne celu robota. W oparciu o te współrzędne oraz dane otrzymywane z niższych warstw systemu, warstwa wysokopoziomowa przeprowadza autonomiczną nawigację, wykonując zadania opisane w sekcji 3.5.

Podział systemu na warstwy i moduły pozwala na oddzielenie różnych funkcjonalności i możliwość ich niezależnej rozbudowy i testowania, pod warunkiem zachowania określonych interfejsów komunikacji. Ponadto wykonywanie funkcjonalności niskopoziomowych i wysokopoziomowych na oddzielnych urządzeniach, zapewnia ciągłość działania. W kolejnych rozdziałach opisany zostanie projekt, realizacja i działanie wymienionych powyżej fragmentów systemu.

5.2. Skaner LiDAR

5.2.1. Projekt skanera

Zadaniem skanera LiDAR jest wykonanie skanu otoczenia, poprzez pomiar odległości pomiędzy czujnikiem a elementami środowiska, w wielu kierunkach, o określonej rozdziel-



Rysunek 5.3. Szkic struktury mechanicznej skanera

czości. Ponieważ w pracy wykorzystany ma być jeden czujnik jednowymiarowy, konieczne jest zaprojektowanie mechanizmu obracającego czujnikiem w dwóch osiach.

Czujnik powinien cechować się jak największą częstotliwością odczytów aby zmaksymalizować również częstotliwość pełnych skanów. Zasięg sensora powinien być adekwatny do użytku w otwartych przestrzeniach i przekraczać 10m.

Przeszkody oraz obiekty charakterystyczne znajdują się w płaszczyźnie robota, na różnych wysokościach. Optymalne jest więc wykonanie skanu we wszystkich kierunkach w poziomie (obrót wokół osi z), oraz w różnych odchyleniach od płaszczyzny (obrót wokół osi y czujnika). Możliwe byłoby również skanowanie jedynie wycinka przestrzeni (np. z przodu robota), byłoby to jednak niepraktyczne - zmiana kierunku ruchu głowicy na skrajach wycinka obciążałaby silnik bezwładnością głowicy i uniemożliwiałaby uzyskanie wysokich prędkości. Przy ciągłym obrocie, bezwładność głowicy jest utrzymywana.

Skoro więc szybka zmiana kierunku ruchu jest kosztowna, optymalnym sposobem skanowania jest wykonywanie szybkich obrotów wokół osi z, zmieniając za każdym razem nachylenie o pewien kąt. Wówczas ruch którego zwrot musi być zmieniany będzie na tyle wolny, że nie obciąży znacząco silnika i sterowanie nim nie będzie zaburzone przez bezwładność.

Wybrany sposób poruszania wymaga dwóch silników. Silnik M1 obracający głowicą wokół osi z i silnik M2 nachylający czujnik wokół osi y. Schemat opracowanej struktury przedstawiono na rys. 5.3. Obrót wokół osi z jest szybszy, wymaga więc silnika o większej mocy i masie. W łańcuchu kinematycznym umieszczony więc został on jako pierwszy, zamontowany do platformy mobilnej będącej podstawą. Porusza on głowicą, do której zamontowany jest silnik M2, zmieniający nachylenie sensora przegubem obrotowym.

Silnik M1 musi umożliwiać precyzyjne sterowanie pozycją - w każdej chwili wykonywania pomiaru kierunek musi być zgodny z zadanym, aby ten był poprawny. Ponadto, pomiary powinny następować w równych odstępach kątowych. W powyższe wymagania

doskonale wpisuje się silnik krokowy. Wykonanie kroku następuje między odczytami i skutkuje przemieszczeniem głowicy do pozycji następnego pomiaru. Precyzja jest zachowywana w przypadku gdy kroki wykonywane są asynchronicznie (co może nastąpić, ponieważ czas wykonywania pomiaru przez LiDAR nie jest stały). Silnik krokowy wymaga dedykowanego sterownika, aby możliwe było sterowanie nim za pomocą mikrokontrolera.

Wymagania wobec silnika M2 są mniejsze, ponieważ może poruszać się on wolniej. Priorytetem jest natomiast minimalna masa silnika, gdyż umieszczony jest na ruchomej głowicy. Sprawdzi się więc w tym przypadku serwomechanizm modelarski. Umożliwia on zadanie i utrzymanie pozycji.

Ciągły obrót głowicy uniemożliwia wykorzystanie przewodów do zasilania i komunikacji z czujnikiem i silnikiem M2. Zastosowany został więc pierścień ślizgowy, umożliwiający transmisję bez skręcania przewodów poprzez szczotki i pierścienie do elementu obracającego się. Pierścień uniemożliwia napędzanie obrotu współosiowo, dodana została więc przekładnia z dwóch kół zębatych (Z1 przyczepione do wału silnika krokowego, Z2 do obrotowej części pierścienia i głowicy).

Aby wykonać skan poprawnie, konieczna jest znajomość centralnej pozycji skanera, ponieważ silnik krokowy nie daje możliwości pomiaru absolutnej pozycji. Aby ją wykryć wykorzystany jest czujnik magnetyczny typu kontaktron.

Skaner ma obracać się z dużą prędkością, konieczne jest więc żeby jego środek ciężkości znajdował się jak najbliżej osi obrotu jak to możliwe, aby zapewnić równomierne obciążenie mechanizmu i ograniczyć wibracje.

5.2.2. Sterownik niskopoziomowy skanera

System sterowania skanerem oparty jest na mikrokontrolerze, oddzielnym od mikrokontrolera obsługującego niskopoziomowe funkcje łazika. Pozwala to ograniczyć potencjalne opóźnienia (które mogłyby wynikać z równoległej obsługi różnych sensorów), oraz ułatwić oddzielne testowanie skanera. Zadaniem układu jest akwizycja danych z sensora, sterowanie silnikami i przesyłanie danych.

Mikrokontroler obsługuje następujące urządzenia:

- LiDAR, wykorzystując interfejs I2C.
- Sterownik silnika krokowego do kontroli obrotu wokół osi z, połączony z silnikiem krokowym za pomocą 4 przewodów fazowych, z wyprowadzeniami do sygnałów kroku i kierunku kroku podłączonymi do mikrokontrolera. Kalibracja prądu sterownika i mikrokroków odbywa się poprzez zwory.
- Serwonapęd do sterowania obrotem wokół osi y, sterowany sygnałem PWM.
- Kontaktron do centrowania.

Układ komunikuje się z głównym komputerem za pośrednictwem protokołu UART przez kabel USB.

Oprogramowanie na mikrokontrolerze pozwala na:

- Zlecanie i odczyt pomiaru z czujnika LiDAR za pomocą odpowiednich wiadomości przesyłanych protokołem I2C.
- Sterowanie silnikami poruszającymi głowicą.
- Otrzymywanie poleceń (start, stop) i ustawień przez UART.
- Przesyłanie pomiarów przez UART.
- Kalibrację i centrowanie skanera z wykorzystaniem kontaktronu. Głowica obraca się wokół osi z do momentu rozłączenia kontaktronu (po wcześniejszym załączeniu). Jako pozycja centralna wykorzystywana jest średnia pozycji rozłączenia i załączenia.
- Rozpędzanie głowicy po rozpoczęciu skanowania - natychmiastowe zadanie pełnej prędkości silnikowi krokowemu doprowadziłoby do pomijania kroków i utraty pozycji.

Proces sterowania silnikami i przesyłania danych realizowany jest w czasie, gdy sensor dokonuje pomiaru, a operacje te są przerwane w chwili, gdy nadejdzie czas na odczyt.

Pomiary dokonywane są od najniższej linii do najwyższej. Aby przeskoki między liniami były jak najmniejsze, po dotarciu do najwyższej linii kierunek ruchu jest odwracany.

Z zewnątrz przez odpowiednią wiadomość przesłaną przez UART (gdy nie jest wykonywany skan) możliwa jest zmiana ustawień wymienionych w tab. 5.1.

Wartość	Jednostka	Opis
Rozdzielczość mikrokroków	Liczba (1, 2, 4, 8)	Zmienia rozdzielczość poziomą skanu (odpowiednio 200, 400, 800, 1600).
Rozdzielczość pionowa	Liczba całkowita	Zmienia rozdzielczość pionową skanu (liczba różnych nachyleń).
Minimalny kąt nachylenia	Wartość w stopniach	Określa nachylenie najniższej linii skanu. Wartość 90 stopni odpowiada skanom w płaszczyźnie skanera.
Krok skanu w pionie	Wartość w stopniach	Określa różnicę w kącie pomiędzy pomiarami w pionie.
Maksymalny kąt nachylenia	n.d. (wyliczany automatycznie)	Obliczany z wzoru (Minimalny kąt + Krok w pionie · Rozdzielczość pionowa)

Tabela 5.1. Ustawienia skanera

Pomiary przesyłane są pojedynczo protokołem UART, jako odległości w centymetrach, bez zbędnego przetwarzania. Kierunek danego odczytu określany jest w wysokopoziomowym sterowniku na podstawie kolejności napływających danych. Aby umożliwić wykrycie błędu w transmisji np. pominięcia odczytu, na koniec każdej linii i skanu przesyłane są dodatkowe znaki informacyjne.

5.3. Dostosowanie platformy mobilnej

Aby platforma mobilna mogła operować autonomicznie, konieczne jest rozszerzenie jej funkcjonalności. Musi być możliwe sterowanie robotem poprzez komunikację z komputerem autonomii (w odróżnieniu od domyślnego sposobu sterowania drogą radiową)

oraz platforma musi być wyposażona w dodatkowe czujniki dostarczające informacji o położeniu i orientacji robota.

5.3.1. Czujniki

Poza skanerem LiDAR opisanym w rozdziale 5.2, platforma została wyposażona w IMU i odbiornik GNSS (opisane w sekcji 3.4).

Z IMU pozyskiwane są dane z akcelerometru, żyroskopu i magnetometru. Pomiar kierunku wektora przyspieszenia ziemskiego (z akcelerometru), oraz całkowanie wartości prędkości kątowej (z żyroskopu) pozwala na precyzyjne obliczenie orientacji robota. Całkowanie prędkości kątowej narażone jest na dryf w czasie (jeżeli rejestrowana jest niezerowa prędkość gdy sensor się nie obraca). Wykorzystanie kierunku wektora przyspieszenia ziemskiego pozwala skorygować ten błąd. W niniejszej pracy wykorzystane będzie bardziej kompleksowy sposób obliczania orientacji, opisany w sekcji 5.4.3. Dane z magnetometru pozwalają na określenie kierunku północy, co zapewnia bezwzględną wartość obrotu robota wokół osi z. Dane z akcelerometru, po dwukrotnym całkowaniu, stanowią potencjalne źródło informacji do wyznaczenia położenia robota. Dwukrotne całkowanie powoduje jednak kumulację błędów, co prowadzi do powstania znacznego dryfu w krótkim czasie i ogranicza praktyczną użyteczność tej informacji.

Odbiornik GNSS pozwala na uzyskanie informacji o lokalizacji globalnej. Komunikacja z satelitami GNSS jest możliwa w dowolnym miejscu, poza zamkniętymi przestrzeniami. Warto jednak podkreślić, że dokładność pomiarów GNSS może być podatna na zakłócenia, przez co precyzja pomiarów jest ograniczona do około ± 1 m. Ponadto lokalizacja otrzymywana przez GNSS zmienia się w sposób skokowy, gdy otrzymywane są nowe dane.

5.3.2. Model ruchu

Robot jest kompatybilny z modelem ruchu napędu różnicowego, ponieważ jego model ruchu stanowi rozszerzenie napędu różnicowego (omówione w sekcji 3.2). Standardowo robotowi zadawana jest prędkość liniowa wzdłuż osi x (V_x) oraz prędkość kątowa wokół osi z (ω_z). Wówczas wzory na prędkość kątową kół prezentują się następująco [20]:

$$\dot{\theta}_L = \frac{V_x}{R} - \frac{L \cdot \omega_z}{2R}$$
$$\dot{\theta}_R = \frac{V_x}{R} + \frac{L \cdot \omega_z}{2R}$$

Gdzie:

$$V_x = \text{prędkość liniowa}$$
$$\omega_z = \text{prędkość kątowa}$$
$$R = \text{promień koła}$$
$$L = \text{odległość między kołami}$$
$$\dot{\theta}_L = \text{prędkość kątowa lewego koła}$$
$$\dot{\theta}_R = \text{prędkość kątowa prawego koła}$$

Wyznaczony został wzór na kąt wychylenia przegubu zależnie od zadanych prędkości:

$$\alpha = 2\arctan(\frac{D\omega_z}{V_x})$$

Gdzie:

$$D = \text{odległość między przegubem i osią wózka napędowego}$$
$$\alpha = \text{kąt skrętu}$$

Wartość kąta skrętu jest ograniczona mechanicznie do $90° \pm 20,1°$. Gdy wymagane jest większe wychylenie robot może być sterowany jako standardowy napęd różnicowy (z zablokowanym przegubem w środkowej pozycji).

Wartości używanych we wzorach parametrów zostały zmierzone i wynoszą:

$$R = 59,3mm$$
$$L = 344mm$$
$$D = 192mm$$

5.3.3. Sterownik niskopoziomowy

Robot pierwotnie wyposażony był w sterownik niskopoziomowy bazujący na mikrokontrolerze ATmega2560 w Arduino Mega 2560. Do mikrokontrolera podłączono dodatkowo:

- IMU, wykorzystując interfejs I2C.
- GPS, wykorzystując interfejs UART.

Do oprogramowania zainstalowanego na mikrokontrolerze dodano obsługę oraz pozyskiwanie danych z wymienionych czujników. Dane z modułu GNSS nie wymagają wstępnego przetwarzania. Konieczna natomiast była wstępna konfiguracja modułu. Czujnik IMU natomiast wymaga kalibracji. Proces kalibracji został opisany w sekcji 6.2.2.

Ponadto dodano możliwość sterowania silnikami platformy poprzez interfejs UART.

Otrzymywane są wiadomości ze sterowaniem, zgodne z opracowanym interfejsem przedstawionym w tab. 5.2.

Wartość	Jednostka	Opis
Prędkość lewa	Liczba (0 - 510)	Prędkość zadana kół z lewej strony. 255 odpowiada wyłączeniu, wartości > 255 jeździe w przód, < 255 jeździe w tył
Prędkość prawa	Liczba (0 - 510)	Prędkość zadana kół z prawej strony. 255 odpowiada wyłączeniu, wartości > 255 jeździe w przód, < 255 jeździe w tył
Wychylenie skrętu	Liczba (0 - 180)	Skręt centralnego przegubu. 90 odpowiada jeździe prosto, wartości > 90 skręcaniu w prawo, < 90 w lewo

Tabela 5.2. Interfejs sterowania robotem

Sterowanie jest realizowane w pętli otwartej ze względu na brak enkoderów w wyposażeniu platformy mobilnej. Napędy korzystają z przekładni o bardzo niskim przełożeniu, co skutkuje niewielkim obciążeniem silników. Gdy silnik jest obciążony małym momentem obrotowym, dana zmiana procentowa momentu (np. wynikająca ze zmiany nachylenia terenu), powoduje proporcjonalnie małą zmianę prędkości, ponieważ zmiana prędkości silnika jest proporcjonalna do wartości zmiany obciążającego go momentu obrotowego [21]:

$$\omega = \frac{U}{k} - \frac{R}{k^2} \cdot M$$
$$d\omega = -\frac{R}{k^2} \cdot dM$$

Gdzie:

U = Napięcie zasilania

R = Rezystancja silnika

k = Współczynnik zależny od projektu silnika

M = Moment obrotowy

ω = Prędkość kątowa silnika

Taka konfiguracja bezpośrednio przyczynia się do zwiększenia precyzji sterowania w pętli otwartej. Obliczanie współczynników do przeliczania prędkości zadanej na prędkość silników odbywa się w sekcji 5.4.2 zgodnie z wzorami wyznaczonymi w 5.3.2. Gdy wymagany kąt skrętu jest większy od maksymalnego wychylenia przegubu, ale poniżej określonego progu, kąt wychylenia przegubu utrzymywany jest na maksymalnej wartości. Powyżej tego progu przegub blokowany jest w pozycji środkowej.

System sterowania i komunikacji został zabezpieczony przed błędami. Po otrzymaniu zadanej wartości prędkości, jest ona utrzymywana do momentu nastąpienia jednego z następujących wydarzeń:

- otrzymania nowej wiadomości z zadaną prędkością (np. w szczególności prędkości 255 w celu zatrzymania),
- upłynięcia określonego limitu czasu od ostatniej wiadomości z prędkością (ustawione na 0.5s co odpowiada przejechaniu przez robota maksymalnie 12 cm),
- utraty komunikacji radiowej ze stacją bazową (wykorzystywanej standardowo do sterowania ręcznego robotem).

Ponadto, platforma mobilna przesyła dane z czujników, również za pośrednictwem interfejsu UART. Przesyłane są bezpośrednio, skalibrowane dane, pozyskane z sensorów.

5.4. Obsługa sprzętu w systemie ROS2

Aby umożliwić integrację sprzętu w systemie ROS, konieczne jest stworzenie dedykowanych sterowników zapewniających skuteczną komunikację z urządzeniami. Te sterowniki powinny również wspierać standardy komunikacji wewnątrz ROS2 poprzez subskrybowanie i publikowanie wiadomości odpowiedniego typu, które zawierają dane z sensorów oraz polecenia sterujące dla robota. Taka implementacja umożliwi efektywną interakcję z innymi węzłami w systemie ROS2 oraz umożliwi wizualizację danych przy użyciu zintegrowanej aplikacji RVIZ.

5.4.1. Obsługa skanera

Utworzono węzeł ROS dedykowany do obsługi skanera. Po uruchomieniu węzła przesyłane są do skanera ustawienia rozdzielczości (ustawiane jako parametry ROS) i polecenie rozpoczęcia skanowania. Po kalibracji i rozpędzeniu głowicy przesyłane są pomiary.

Głównym zadaniem węzła jest przetworzenie otrzymywanych pomiarów w chmurę punktów. W ROS2 chmura punktów standardowo przesyłana jest jako wiadomość typu sensor_msgs/PointCloud2 [22] na temacie /lidar_scan. Ma ona zawierać listę współrzędnych punktów. Na podstawie kolejności otrzymywanych punktów, określany jest kierunek w jakim skierowany był sensor podczas wykonywania danego pomiaru i obliczane są współrzędne punktu w przestrzeni odpowiadające pomiarowi, zgodnie z wzorami:

$$x = d \cdot \cos\phi \cdot \sin\theta$$
$$y = d \cdot \cos\phi \cdot \cos\theta$$
$$z = d \cdot \sin\phi$$

Gdzie:

ϕ = Kąt pionowy
θ = Kąt poziomy
d = Odległość z pomiarów
x = Współrzędna x w przestrzeni
y = Współrzędna y w przestrzeni
z = Współrzędna z w przestrzeni

Następnie po otrzymaniu wszystkich punktów z danego skanu, publikowana jest wiadomość PointCloud2, z odpowiednią sygnaturą czasową i układem współrzędnych.

5.4.2. Obsługa robota

Utworzono węzeł ROS dedykowany do obsługi platformy mobilnej. Węzeł odbiera od sterownika platformy mobilnej dane z IMU i odbiornika GNSS (3.4), oraz przesyła do robota informacje o prędkości.

Węzeł subskrybuje w ROS temat /cmd_vel, na którym publikowana jest zadana prędkość. Gdy otrzymana jest nowa wiadomość, jest ona przeliczana na prędkości kół oraz nachylenie przegubu, zgodnie z wzorami w sekcji 5.3.2. Następnie jest przeliczana do zakresu wartości [0, 510] do sterowania w pętli otwartej, tak aby maksymalne prędkości w obu kierunkach ruchu odpowiadały wartościom na skraju przedziału. Skręt przegubu jest przeliczany do zakresu [0, 180]. Prędkość jest ponadto ograniczana aby nie przekroczyła wartości maksymalnej.

Gdy otrzymane są dane z IMU przez interfejs UART. Jednostki zostają przekonwertowane do standardowych dla ROS, aby dane mogły być poprawnie interpretowane. Następnie zostają opublikowane na tematach /imu/data_raw (przyspieszenie, prędkość kątowa) i /imu/mag (wektor pola magnetycznego). Dane muszą być podzielone na 2 tematy ze względu na typ danych wiadomości w systemie ROS - wiadomość typu sensor_msgs/Imu [23] powinna zawierać orientację robota, nie wektor pola magnetycznego. Konieczne będzie więc dokonanie dalszego przetwarzania.

Dane z GNSS po otrzymaniu zostają publikowane na temacie /gps_rover w wiadomości typu sensor_msgs/NavSatFix [24]. Na podstawie uzyskanych wartości błędu generowana jest macierz kowariancji błędu sensora.

Ze względu na brak enkoderów na kołach robota, utworzony został dodatkowy węzeł, generujący informacje o odometrii kół na podstawie zadanej prędkości. Nie zastępuje rzeczywistych danych o prędkości pojazdu, ale może być wykorzystana jako dodatkowe źródło informacji o ograniczonej dokładności.

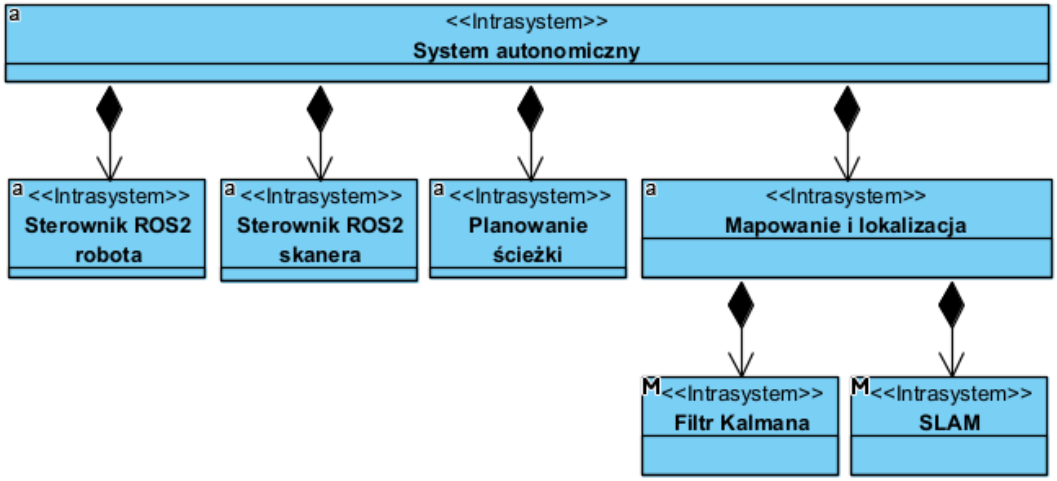
5.4.3. Przetwarzanie IMU

Aby uzyskać dane o orientacji i utworzyć pełną wiadomość w systemie ROS, konieczne jest dokonanie dodatkowego przetwarzania na danych w temacie /imu/data_raw. W tym celu został wykorzystany filtr komplementarny (3.9). Na podstawie danych z żyroskopu i akcelerometru, filtr komplementarny eliminuje dryf pomiarowy i zapewnia stabilną orientację obiektu. Węzeł publikuje wiadomość ze skomponowanymi danymi z IMU (generowany jest kwaternion orientacji) na temacie /imu/data.

5.5. System autonomiczny

5.5.1. Struktura oprogramowania wysokopoziomowego

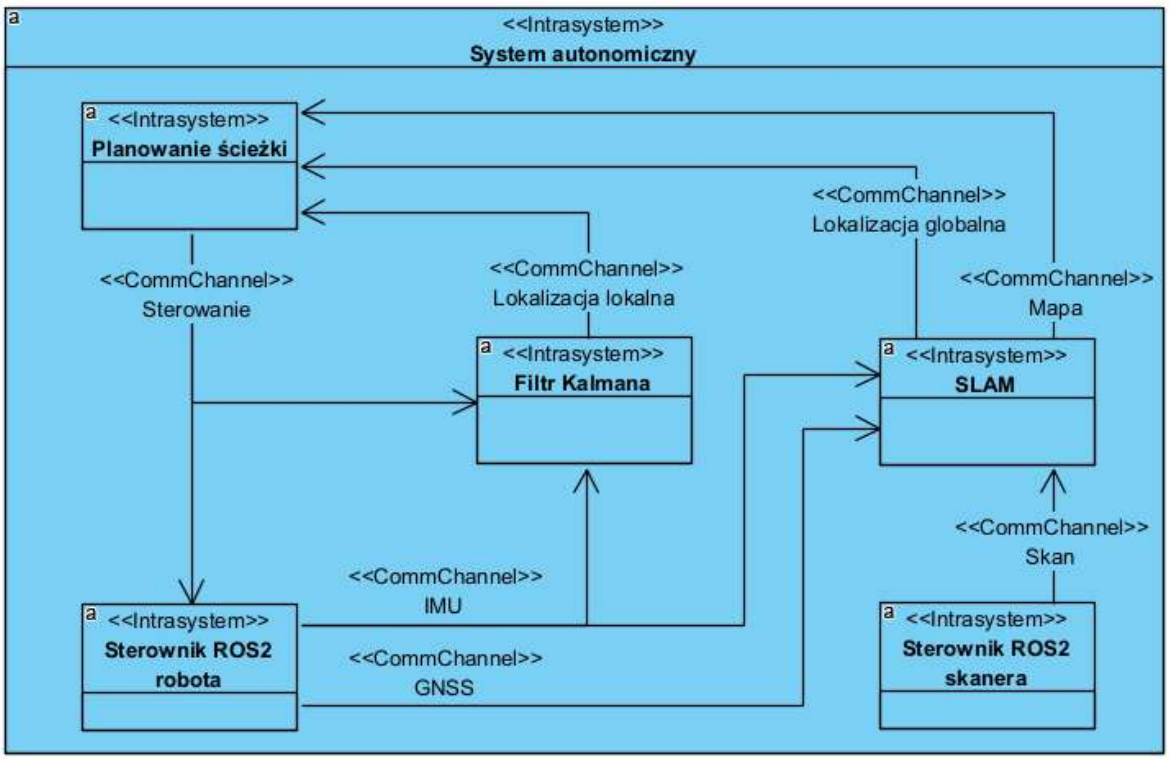
Elementy opisane w sekcji 5.4 dostarczają danych, umożliwiających utworzenie funkcjonalnego oprogramowania zarządzajacego nawigacją autonomiczną. Oprogramowanie musi umożliwiać lokalizację robota, mapowanie, planowanie ścieżki i jej wykonanie. Opracowano strukturę wykonującą te zadania, przedstawioną na rys. 5.4. Komunikację pomiędzy modułami przedstawiono na rys. 5.5



Rysunek 5.4. Proponowana struktura oprogramowania

Lokalizacja, zgodnie z konwencją opisaną w sekcji 3.7 została podzielona na lokalizację lokalną i globalną. Lokalizacja lokalna realizowana jest z wykorzystaniem filtru kalmana (3.10), na podstawie danych z IMU i zadanej prędkości kół. Dostarcza informację o lokalizacji robota w układzie odniesienia odometrii. Lokalizacja globalna natomiast, generowana jest z wykorzystaniem algorytmu SLAM (3.8), na podstawie skanów z LiDARA a także danych z czujników IMU i GPS. Algorytm tworzy również mapę otoczenia. W efekcie uzyskane jest przesunięcie układu odom względem układu map (z lokalizacji globalnej) i przesunięcie robota względem układu odom (z odometrii).

Dysponując lokalizacją robota, mapą otoczenia i zadanymi współrzędnymi celu, zadaniem planera jest wygenerowanie ścieżki po której ma poruszać się robot. Ścieżka ta podobnie do lokalizacji jest podzielona na plan globalny, aktualizowany z niską częstotli-



Rysunek 5.5. Proponowana struktura oprogramowania

wością i uwzględniający topologię terenu oraz plan lokalny, biorący pod uwagę pobliskie przeszkody. Następnie ścieżka jest wykonywana, poprzez zadanie odpowiedniej prędkości liniowej i kątowej robota.

5.5.2. Lokalizacja

Filtr Kalmana używany do lokalizacji lokalnej wykorzystuje jedynie informacje o orientacji z IMU. Wykorzystanie informacji o prędkości kątowej doprowadziłoby do dwukrotnego wykorzystania tych samych danych - orientacja jest obliczana na podstawie danych z żyroskopu przez filtr komplementarny. Wykorzystanie przyspieszenia mogłoby pozwolić na obliczanie przesunięcia robota, ale doprowadziłoby do dużego dryfu lokalizacji spowodowanego przez dwukrotne całkowanie przyspieszenia. Ponadto filtr wykorzystuje zadaną prędkość liniową. Zadana prędkość kątowa została pominięta, ponieważ IMU jest znacznie precyzyjniejszym źródłem orientacji. Prędkość zadana zastępuje bezpośrednie dane o prędkości kół, gdyż platforma nie jest wyposażona w enkodery.

Do lokalizacji globalnej wykorzystany został algorytm SLAM w wersji trójwymiarowej. Jego zadaniem jest dopasowywanie chmur punktów ze skanów do przechowywanej w pamięci mapy otoczenia. Zastosowanie trójwymiarowej wersji algorytmu jest kluczowe przy nawigacji w nieusystematyzowanym terenie - robot może znajdować się pod różnymi nachyleniami, oraz zmienia położenie również w osi pionowej. Dane o orientacji otrzymywane z IMU są wykorzystywane przez algorytm do ograniczenia możliwych przekształceń pomiędzy mapą a nowymi skanami. Ponadto dane otrzymane przez SLAM są łączone

ze współrzędnymi pochodzącymi z odbiornika GNSS. Współrzędne GNSS cechują się zdecydowanie mniejszą dokładnością niż dopasowania skanów, ale pomagają wyeliminować dryf lokalizacji podczas długich przejazdów, oraz porównać mapę globalną do rzeczywistych współrzędnych geograficznych.

5.5.3. Mapowanie

Trójwymiarowa mapa otoczenia generowana jest również przez algorytm SLAM. Jednakże planowanie trajektorii robota w przestrzeni byłoby problematyczne - robot, jako pojazd lądowy, zdolny jest jedynie to sterowania prędkością w płaszczyźnie. Wobec tego generowane są na bieżąco dwuwymiarowe mapy kosztu otoczenia (ang. costmap), lokalna i globalna, przeznaczone do wyznaczania trajektorii. Koszt na mapie reprezentuje trudność przejechania danego fragmentu przez robota. Mapa lokalna odnosi się do niewielkiego obszaru w bezpośrednim sąsiedztwie robota i jest aktualizowana na podstawie najnowszych skanów otoczenia. Mapa globalna obejmuje cały obszar znanego środowiska, w którym porusza się robot. Aktualizowana jest również ze skanów LiDAR i wykorzystuje lokalizację wyznaczoną przez algorytm SLAM. W mapach stosowana jest tzw. inflacja - wykryte przeszkody są powiększane, aby zmniejszyć prawdopodobieństwo kolizji robota.

5.5.4. Planowanie

Na podstawie dwuwymiarowej mapy globalnej, i celu zdanego przez operatora opracowywana jest ścieżka prowadząca od lokalizacji robota do docelowych współrzędnych. Ścieżka omija znane przeszkody znajdujące się na mapie, biorąc również pod uwagę ich inflację. Ścieżka globalna nie jest jednakże realizowana w czasie rzeczywistym - to bezpośredniego sterowania robotem w czasie rzeczywistym wykorzystywana jest mapa lokalna i umieszczane na niej dynamicznie wykrywane przeszkody. Kierunek ścieżki globalnej jest celem dla planera lokalnego. Planer lokalny bezpośrednio oblicza zadaną prędkość dla robota, biorąc pod uwagę mobilność platformy.

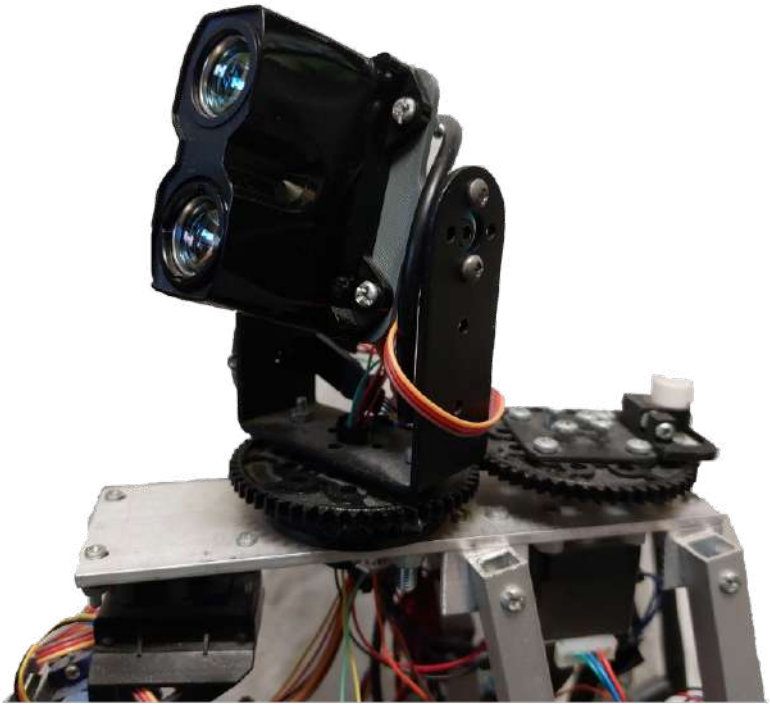
6. Realizacja projektu

Utworzone zasoby są udostępnione w repozytorium w serwisie github [25] .

6.1. Realizacja skanera

6.1.1. Budowa skanera

Zrealizowane zostały dwie wersje skanera. Opisana zostanie implementacja ostatecznej wersji, a następnie przedstawione będą problemy pierwotnej wersji oraz sposób ich naprawy. Ostateczną wersję przedstawiono na rys. 6.1.



Rysunek 6.1. Skaner LiDAR

Jako punkt montażu skanera wybrano najwyższy punkt platformy mobilnej, na szczycie masztu sensoryczno-komunikacyjnego. Pozwala to zminimalizować przysłanianie otoczenia przez robota, oraz zapewnić większe pole widzenia, dalej od podłoża. Na szczycie masztu zamontowano silnik krokowy, pierścień ślizgowy i przekładnię zębatą z przełożeniem 1:1. W efekcie na napędzanym kole zębatym uzyskano obrotowy punkt montażowy dla głowicy, z doprowadzonymi przewodami. Ponadto zainstalowany został kontaktron (detektor na szczycie platformy i magnes w określonej pozycji na kole zębatym).

Wykorzystano silnik krokowy JK28HS32-0674 [26] w standardzie NEMA 11. Zapewnia on wystarczający moment obrotowy i prędkość, zachowując niską jak na silnik krokowy masę. Ma on rozdzielczość wynoszącą 200 kroków na obrót, co odpowiada kątowi 1,8 stopnia na krok.

Dodany został mechanizm kalibracji kontaktronu – zaprojektowany został mechanizm umożliwiający przesuwanie magnesu prostopadle do promienia zębatki, co przekłada się

Rysunek 6.2. Mechanizm kalibracji kontaktronu

na kalibracje kąta uznawanego za środkową pozycję LiDARu (rys. 6.2). Wykonany został w technologii druku 3D.

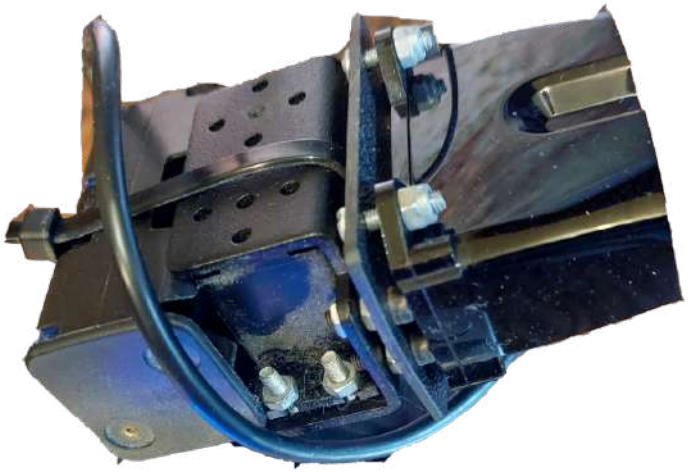
Do koła zębatego zamontowana została aluminiowa rama głowicy, sztywno połączona z wałem silnika M2. Wybrany został serwomechanizm TowerPro SG-90 [27] o masie 9g. Obudowa silnika i czujnik odległości połączone zostały za pomocą dedykowanego zaprojektowanego elementu wykonanego technologią druku 3D. W efekcie, ruch wału silnika powoduje zmianę nachylenia sensora.

Do pomiaru odległości wybrany został moduł firmy Garmin, Lidar Lite V3 HP [28]. Moduł pozwala na dokonywanie odczytów z częstotliwością do 1500 Hz na odległość do 40m. Możliwa jest komunikacja z nim za pośrednictwem interfejsu I2C.

Czujnik waży 38 g i nie jest umieszczony na osi obrotu głowicy. W celu osiągnięcia równowagi, konieczne było zainstalowanie przeciwwagi symetrycznie względem sensora. Przeciwwaga została dobrana tak, aby środek ciężkości całego zespołu sensora (czujnik, element drukowany, silnik M2 i przeciwwaga) znajdował się na obu osiach obrotu skanera. Dzięki temu konstrukcja jest zbalansowana niezależnie od orientacji sensora. Rozmieszczenie podzespołów zostało starannie dobrane, aby zminimalizować moment bezwładności modułu. Wybrano modele o jak najmniejszej masie, umieszczone jak najbliżej osi obrotu (minimalne ramię momentu).

Silnik M1 sterowany jest za pomocą sterownika silników krokowych Pololu 2968 [29] bazującego na układzie MP6500. Pozwala on na poruszanie silnikiem w trybie mikrokroków (do 1/8 kroku), co zwiększa rozdzielczość obrotu silnika na 1600 mikrokroków/obrót. Sterownik zasilany jest prądem 1A i napięciem 32V (bliskim maksymalnemu), aby uzyskać jak najwyższy moment obrotowy przy wysokich prędkościach [30]. Wyższe napięcie powoduje szybsze narastanie zboczy sygnału na fazach, przy tym samym prądzie ustawionym w sterowniku.

Jako sterownik skanera, wybrany został mikrokontroler ATMega 328 w Arduino Nano



Rysunek 6.3. Stara wersja zespołu czujnika

z uwagi na wystarczające możliwości obliczeniowe, dostępną liczbę wyprowadzeń i korzystną cenę. Sterownik widoczny jest na rys. 6.7.

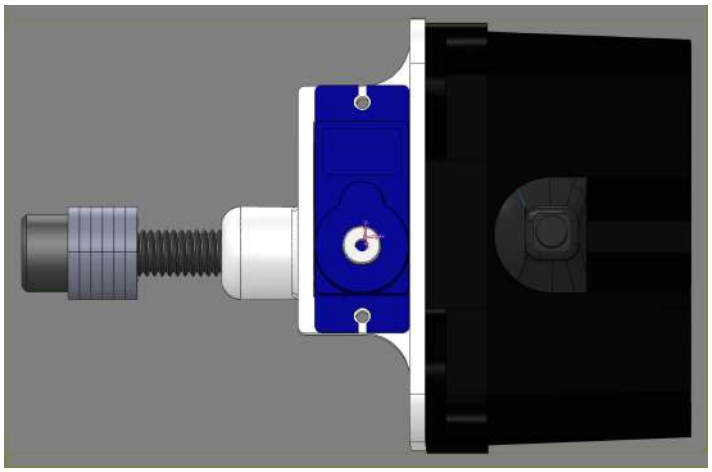
Do obsługi czujnika LiDAR wykorzystano dedykowaną bibliotekę LIDARLite_v3HP [31]. W konfiguracji biblioteki dodana została własna konfiguracja czujnika, pozwalająca na wykonywanie szybszych odczytów poprzez ograniczenie liczby powtórzeń odczytów (SIG_COUNT_VAL, REF_COUNT_VAL) oraz zmianę trybu szybkiego zakończenia odczytu (ACQ_CONFIG_REG). Parametry są opisane w dokumentacji sensora [28].

Aby zwiększyć stabilność czasu pomiaru, wprowadzono dodatkowe opóźnienia, równoważne czasowi zmiany nachylenia skanera, gdy w danym pomiarze nie zachodzi jego zmiana. Wprowadzono również opóźnienia stabilizujące czas przesyłania wyniku skanu, bez względu na ilość przesyłanych znaków (liczbę cyfr wyniku w centymetrach). Inne podejścia mogę wydawać się skuteczniejsze, na przykład zastosowanie przerwań od wystąpienia zdarzenia uzyskania nowego wyniku, czy wykorzystanie timerów. Pierwsze podejście nie było jednak możliwe - czujnik nie udostępnia przerwań. Drugie podejście nie pozwalałoby wykorzystać pewnych różnic w czasach pomiarów, więc wszystkie skany musiałyby być wykonywane z równą, mniejszą częstotliwością. Opóźnienia dobrano mierząc czas wykonywania równoważonych operacji w mikrosekundach na mikrokontrolerze.

6.1.2. Porównanie wersji skanera

Najpierw została zbudowana wstępna wersja skanera, nie stosująca elementów wykonanych w druku 3D i wykorzystująca większy i cięższy serwonapęd (rys. 6.3. W efekcie waga zespołu czujnika wynosiła 167g (w porównaniu do 80g w najnowszej wersji).

Stara wersja borykała się z dużą bezwładnością, powodującą niekiedy pomijanie kroków przez silnik, co doprowadzało do skorumpowania skanu. Redukcja wagi wyeliminowała to zjawisko poprzez zmniejszenie obciążenia silnika. Ponadto, stara wersja cechowała się niezbalansowaniem, co doprowadzało do obserwowalnych oscylacji wieży



Rysunek 6.4. Symulacja środka ciężkości zespołu czujnika

komunikacyjno-sensorycznej, zaburzających stabilność pomiaru. W nowej wersji więc, przygotowany został model 3D i położenie oraz masa przeciwwagi została dobrana w symulacji w programie Solidworks, tak, aby środek ciężkości zespołu sensora znajdował się na osiach obrotu sensora (rys. 6.4). Ponadto, dzięki wykorzystaniu śruby jako przeciwwagi możliwe było dostrojenie środka ciężkości na fizycznym obiekcie. Waga przeciwwagi została dostosowana poprzez obserwację oscylacji masztu. Nagranie filmu w zwolnionym tempie pozwoliło na obserwację środka ciężkości głowicy. Zastosowano również dodatkową przeciwwagę na głowicy, aby zbalansować ją wzdłuż osi y.

6.2. Dostosowanie platformy mobilnej

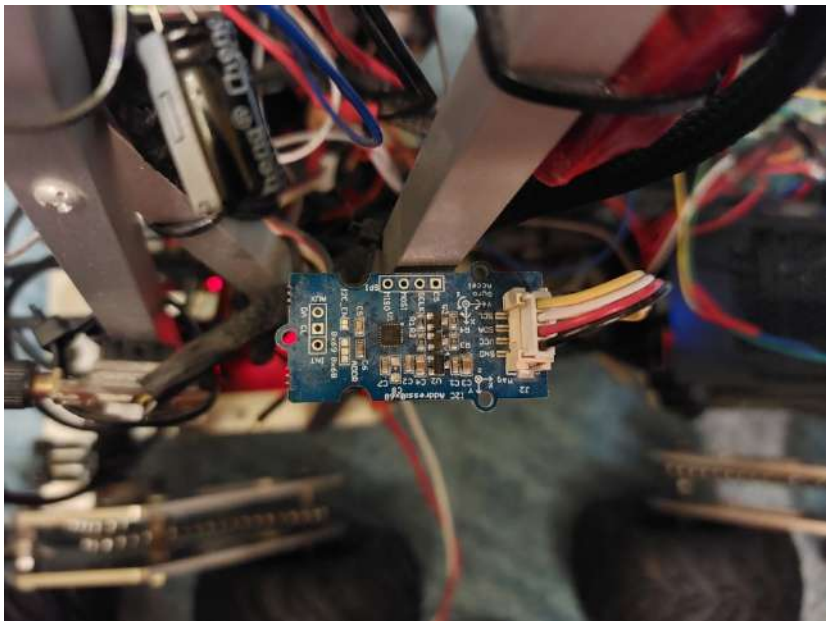
6.2.1. Czujniki

Jako sensor IMU w projekcie zastosowany został czujnik Grove IMU 9DOF v2.0 [13], bazujący na układzie MPU-9250. Pozwala na odczyt wartości z dokładnością do 16 bit i komunikację przez magistralę I2C.

Istotnym aspektem było umieszczenie czujnika na robocie. Sensor został zamontowany w poziomie, zgodnie z orientacjami osi. Magnetometr jest wrażliwy na zakłócenia pola magnetycznego, które mogą być powodowane przez silniki, przewody prądu, lub duże elementy wykonane z ferromagnetyków. Czujnik został więc umieszczony na maszcie robota, z dala od innych komponentów (rys. 6.5). Pole magnetyczne w tym miejscu zostało wcześniej zmierzone ($43\mu T$), i nie odbiegało od typowych wartości (średnio $45\mu T$ w pomieszczeniu). W głównej części robota pole ma wartość ok. $100\mu T$. Do montażu zaprojektowany został dedykowany element, wytworzony metodą druku 3D.

Jako odbiornik GNSS wykorzystany został Ublox NEO-M8N [32]. Jest on kompatybilny ze wspomnianymi systemami satelit i jest wyposażony w zintegrowaną antenę. Komunikacja z modułem odbywa się przez protokół UART.

Odbiornik został również umieszczony na maszcie platformy mobilnej, przesunięty w bok aby ograniczyć przysłanianie satelit przez resztę masztu. Antena została skierowana



Rysunek 6.5. Czujnik Grove IMU 9DOF v2.0 zamontowany na robocie

pionowo w górę by zapewnić widoczność satelit na całym niebie (rys. 6.6). Do montażu zaprojektowany został dedykowany element, wytworzony metodą druku 3D.

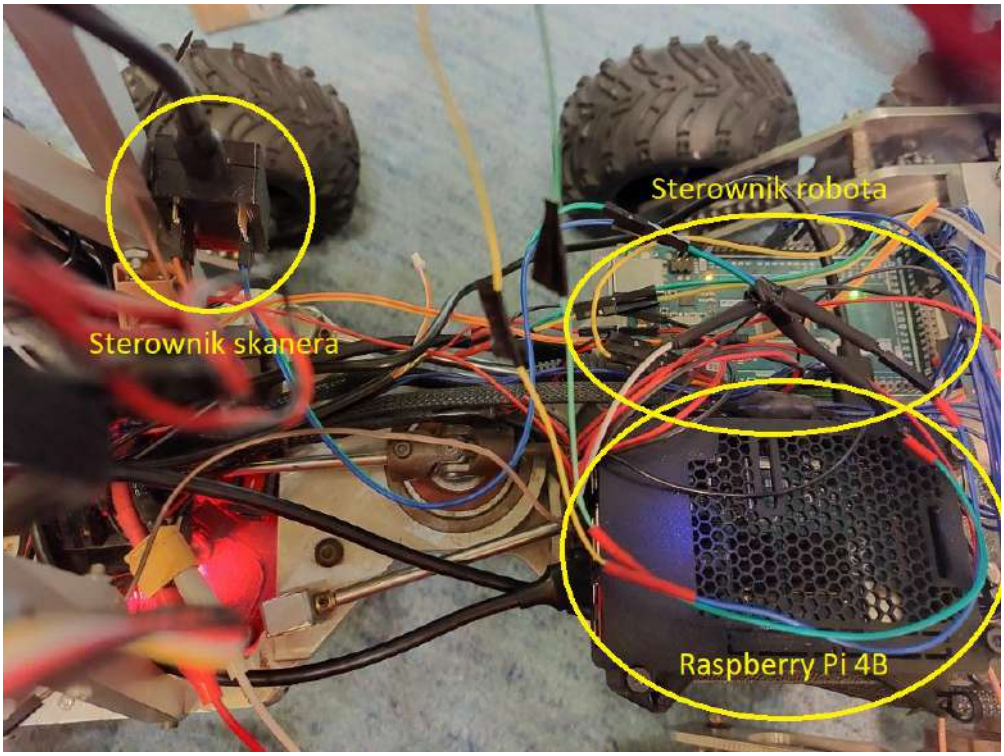
6.2.2. Sterownik niskopoziomowy

Sterownik niskopoziomowy stanowi rozwinięcie pierwotnego sterownika robota, wykorzystującego mikrokontroler ATmega2560. Sterownik widoczny jest na rys. 6.7.

Do obsługi IMU wykorzystano bibliotekę MPU9250[33]. Proces kalibracji magnetometru odbywa się za pomocą wbudowanych funkcji dostępnych w bibliotekę. Kalibracja polega na zbieraniu danych w różnych orientacjach. Natomiast kalibracja żyroskopu przeprowadzana została ręcznie przez autora. Przeprowadzono akwizycję danych, gdy czujnik był w spoczynku. Odczyty są przesuwane o średnią wartość zarejestrowanych pomiarów, aby podczas działania czujnika nie następował dryf orientacji. W przypadku akcelerometru każda oś została przeskalowana tak, aby wartość odpowiadała przyspieszeniu grawitacyjnemu, gdy czujnik był umieszczony zgodnie ze zwrotem danej osi w pionie.

Moduł GNSS obsługiwany jest przez bibliotekę NeoGPS[34]. Moduł został skonfigurowany aby uzyskać wybrane wiadomości w standardzie NMEA [35] przy maksymalnej częstotliwości odczytu, poprzez przesłanie odpowiednich ramek z danymi. Wybrane zostały wiadomości w standardzie NMEA: RMC, GGA, GST - zapewniające współrzędne geograficzne z wysokością oraz dokładność pomiaru (odchylenie standardowe pozycji). Zmieniona została częstotliwość odczytu na maksymalną możliwą przy wybranych wiadomościach, wynoszącą 4Hz.

Rysunek 6.6. odbiornik GNSS Ublox NEO-M8N zamontowany na robocie



Rysunek 6.7. Zainstalowane sterowniki niskopoziomowe i komputer

6.3. Obsługa sprzętu w systemie ROS2

Główny komputer wykorzystywany w sekcji autonomicznej to Raspberry Pi 4B. Jest to kompaktowy komputer jednopłytkowy. Czterordzeniowy procesor ARM Cortex-A72, oraz 4GB pamięci RAM w wybranym wariancie zapewniają wystarczającą wydajność do obsługi wysokopoziomowej części projektu. Niewielkie wymiary i pobór mocy poniżej 15W są adekwatne do wielkości robota i dostępnej pojemności baterii. Ponadto dzięki wystarczającej mocy obliczeniowej, komputer nie ogranicza użytkownika pod względem wyboru systemu operacyjnego, pozwalając wykorzystać standardowe dystrybucje. Wybrano systemem Ubuntu Server 22.04 LTS, pozbawiony interfejsu graficznego i zbędnych dodatków oraz kompatybilny z wybraną wersją ROS2. Zainstalowany komputer widoczny jest na rys. 6.7.

Zarówno robot jak i skaner podłączone zostały do głównego komputera przez UART, przewodami USB. Konieczna jest więc obsługa portu szeregowego w systemie. Aby ujednolicić działanie po podłączeniu i odłączeniu urządzeń (aby zachować stałe identyfikatory portów), dodano zasady udev mapujące urządzenia do własnych portów na podstawie identyfikatorów producenta i modelu, odpowiednio do ttyRVR dla łazika i ttyLID dla skanera.

Węzły ROS do obsługi skanera (lidar controller) i robota (rover controller) zostały zaimplementowane w języku Python. Do obsługi komunikacji szeregowej wykorzystano bibliotekę pySerial [36]. Skonfigurowano baud rate na maksymalną obsługiwaną wartość 115200 bps. Istotne było skonfigurowanie limitów czasu dla komunikacji szeregowej, aby zapewnić płynną komunikację bez zbędnego oczekiwania. Realizacja przesyłania danych wykorzystuje mechanizm odwołań zwrotnych (ang. callback) aby zapewnić płynną obsługę różnych funkcji. Reakcja na wiadomości wewnątrz systemu ROS odbywa się poprzez bezpośrednie odwołania, natomiast odbieranie danych od robota odbywa się poprzez timer.

Obsługa sprzętu została zabezpieczona na wypadek wyłączenia. W przypadku wyłączenia węzła sterującego skanerem, przesyłana jest komenda kończąca skanowanie i zatrzymująca głowicę skanującą, poprzez obsługę sygnału przerwania SIGINT. Dzięki temu głowica skanująca nie porusza się gdy jest to zbędne i oszczędzana jest energia poprzez odcięcie zasilania silnika krokowego. W przypadku wyłączenia węzła odpowiedzialnego za komunikację z robotem, wysyłana jest komenda sterująca z zerową prędkością.

Utworzono również węzeł odom cmd, zastępujący enkodery robota. Subskrybuje on wiadomości na temacie /cmd_vel i na ich podstawie generuje wiadomości na temacie /odom_cmd. Gdy nie jest zadawana prędkość, węzeł nadaje wiadomości z zerową prędkością. Ponadto, wygenerowano macierz kowariancji symulowanego czujnika - odchylenie standardowe prędkości określono na 10% prędkości maksymalnej, na podstawie pomiarów prędkości pojazdu na różnych nawierzchniach.

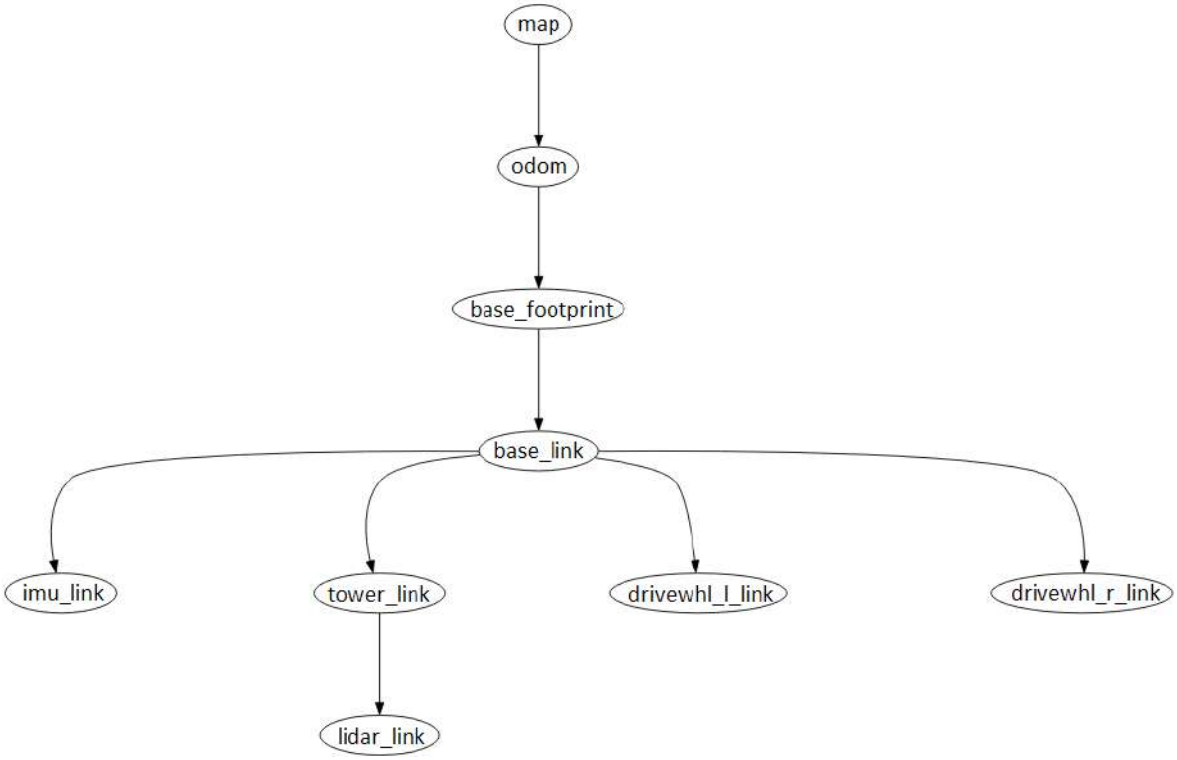
Do generowania orientacji został wykorzystany filtr komplementarny w implementacji

z biblioteki imu_tools: imu_complementary_filter [37]. Filtr został skonfigurowany, aby wykorzystywać dane z magnetometru.

6.4. System autonomiczny

6.4.1. Wstępne wymagania

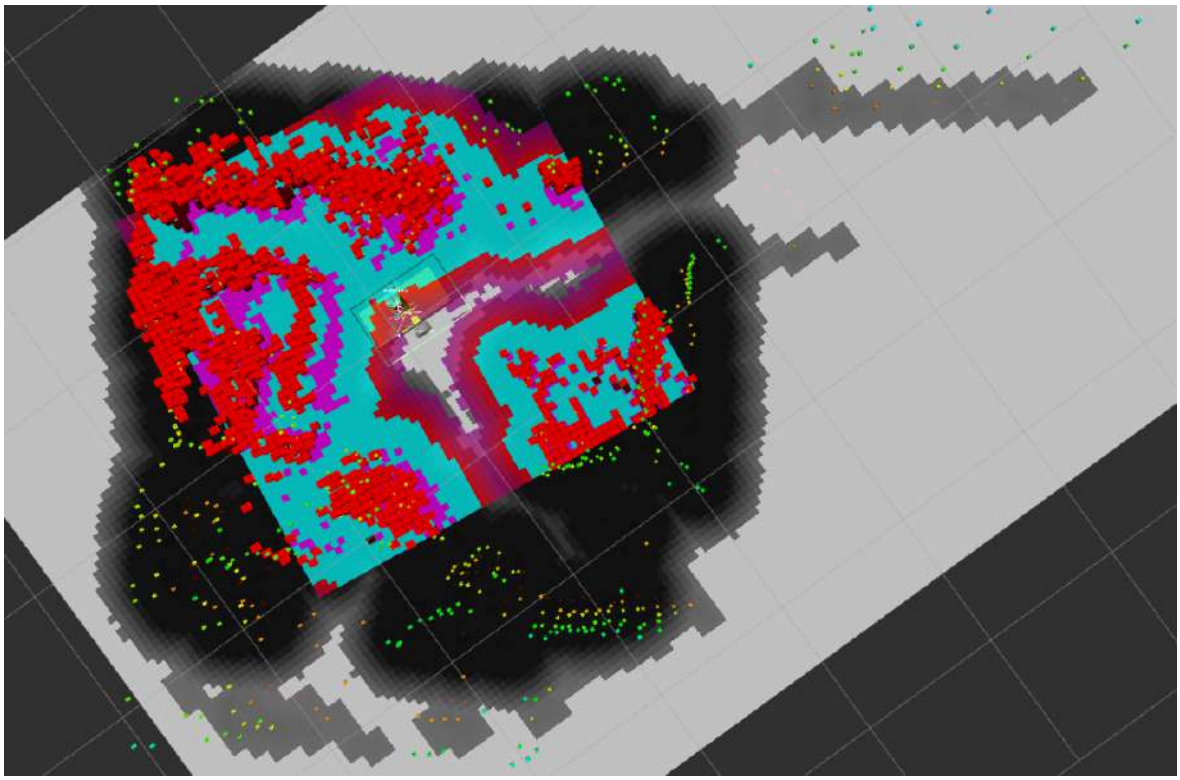
Aby stworzyć system robotyczny z wykorzystaniem ROS2, niezbędne jest przeprowadzenie początkowej konfiguracji. W tym celu utworzono przestrzeń roboczą projektu, a także przygotowano uproszczoną reprezentację robota w formie pliku URDF. Konfiguracja URDF opisuje transformacje między układami współrzędnymi obecnymi w robocie, jednocześnie stanowiąc wizualną reprezentację, którą można wykorzystać do wizualizacji. Schemat układów współrzędnych przedstawiono na rys. 6.8.



Rysunek 6.8. Układy współrzędnych w robocie

Następnie przygotowano plik uruchamiający system w języku python, z wykorzystaniem biblioteki launch będącej częścią ROS2. Uruchamia on następujące programy:

- robot state publisher
- joint state publisher
- rviz2
- rover controller
- lidar controller
- odom cmd
- imu filter



Rysunek 6.9. Wizualizacja działającego systemu w RVIZ 2

- robot localiztion
- cartographer occupancy grid
- cartographer
- nav2
- nav2 costmap 2d markers

Robot state publisher i joint state publisher to standardowe narzędzia ROS, odpowiedzialne za publikowanie przekształceń pomiędzy układami współrzędnych, i stanu stawów pomiędzy nimi.

Węzeł rviz2 odpowiedzialny jest za wyświetlanie wizualizacji działania systemu (rys. 6.9). Uruchamiany jest jedynie w konfiguracji z aktywnym interfejsem graficznym. W wizualizacji zastosowano następujące oznaczenia, zgodnie ze standardem ROS:

- Mapa globalna - skala szarości, proporcjonalna do kosztu
- Mapa lokalna - koszt malejący w kolejności magenta -> cyjan -> czerwony
- Robot - żółty
- Najnowszy skan LiDAR - punkty, wysokość punktu w osi z rosnąca w kolejności czerwony -> zielony
- Lokalne przeszkody - czerwone voxele

Węzły rover controller, lidar controller, odom cmd i imu filter to węzły których tworzenie opisano w sekcji 6.3. Pozostałe programy i ich role zostaną opisane kolejno w dalszej części pracy.

6.4.2. Odometria

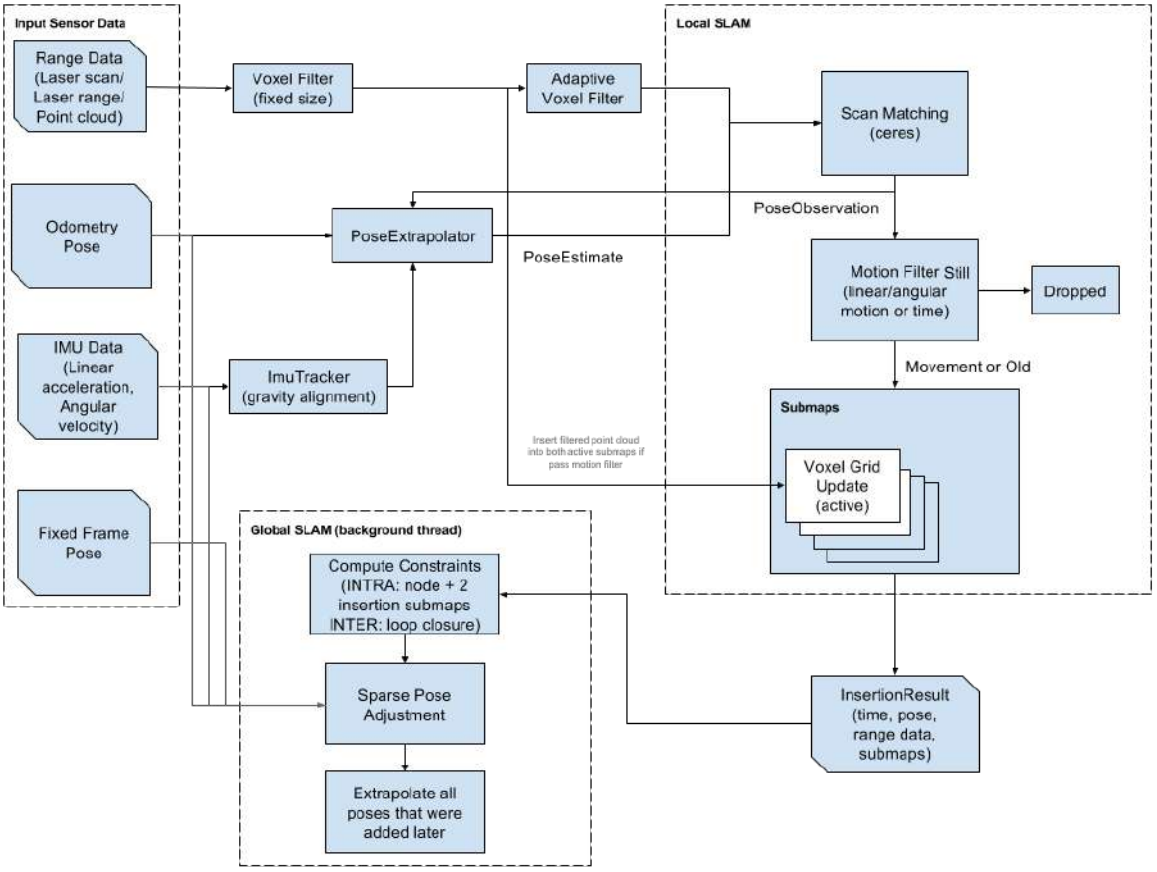
Do lokalizacji lokalnej robota wykorzystano węzeł robot localization. Węzeł wykorzystuje ekf_filter_node z pakietu robot_localization [38], stanowiący standardowe rozwiązanie w systemie ROS. Węzeł bazujące na rozszerzonym filtrze Kalmana i publikuje przekształcenie z układu odom do układu robota. Został skonfigurowany, aby wykorzystywać dane o orientacji z imu zgodnie z planem projektowym. Z węzła odom cmd, do estymacji stanu wykorzystywana jest jedynie prędkość liniowa. Zadana prędkość kątowa i rzeczywista są zbyt różne ze względu na poślizgi podczas skręcania, by dane były użyteczne. Zostało to potwierdzone eksperymentami, dołączenie prędkości kątowej pogarszało odwzorowanie. Przetestowano również działanie odometrii z wykorzystaniem przyspieszenia z akclerometru. Mimo poprawnego działania po uruchomieniu, błędy szybko się kumulowały co doprowadzało do szybkiego dryfu lokalizacji lokalnej. Ponadto węzeł wykorzystuje otrzymywane kowariancje sensorów. Określono również ograniczenia przyspieszenia robota, na podstawie obserwacji czasu uzyskania maksymalnej prędkości przez platformę (0,5s dla przyspieszania i 0,25s dla zatrzymania).

6.4.3. SLAM

Zadania lokalizacji globalnej i mapowania otoczenia wykonywane są przez algorytm SLAM. Większość dostępnych implementacji algorytmu, w tym te wbudowane w system ROS2, działa dla środowisk dwuwymiarowych i jest dostosowana do lokalizacji w pomieszczeniach z wykorzystaniem skanera 2D, co stanowi najpopularniejszą konfigurację robota mobilnego. W niniejszej pracy zastosowano trójwymiarowy algorytm SLAM, kompatybilny ze skanerami 3D. Taką funkcjonalność dostarcza implementacja w bibliotece cartographer ros [39], integrującej z ROS rozwiązanie firmy Google, Cartographer [40]. Rozwiązanie zostało uruchomione w węźle cartographer. Węzeł publikuje przekształcenie z układu map do odom, oraz generowaną mapę.

Biblioteka umożliwia zastosowanie szerokiej gamy sensorów, w tym skanerów 2D i 3D. Schemat jej działania przedstawiono na rys. 6.10 i opiera się na dwóch głównych komponentach:

- SLAM lokalny, wykorzystujący algorytm Ceres Scan Matcher. Dopasowuje on kolejne skany, tworząc tzw. submapy. Wykorzystuje solwer Ceres [41] do zadania optymalizacji. Dopasowanie wspierane jest informacjami o względnym przesunięciu skanów, pozyskiwanymi z IMU i odometrii. Mechanizm został skonfigurowany przez autora podczas testów terenowych. Platforma poruszała się z zadaną prędkością i obserwowano jakość lokalizacji. Zwiększono wagi przyporządkowane karom za przesunięcie i rotację skanu, do momentu wyeliminowania sytuacji gdy pozycja robota zmieniała się w sposób skokowy do innego, błędnego dopasowania. Zachowano przy tym ciągłość trafnego odwzorowania zmian pozycji robota. Szczególnie zwiększona została kara za rotację, ponieważ rotacja jest odwzorowana z dużą dokładnością w transformacji odom -> robot.



Rysunek 6.10. Schemat działania systemu cartographer [39]

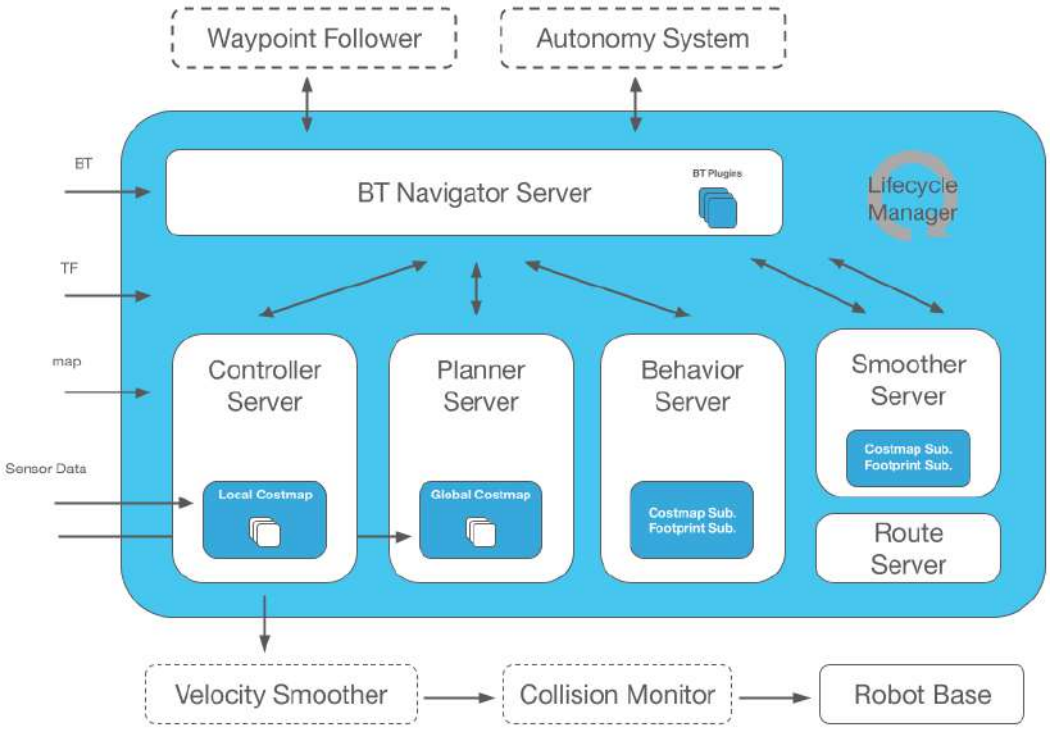
- SLAM globalny, którego zadaniem jest optymalizacja wzajemnego ułożenia submap. Jest to wolniejszy mechanizm wykonywany w tle, rozwiązujący również problem domykania pętli. Dzięki niemu uzyskiwana jest spójna mapa globalna. Ze względu na działanie na wysokim poziomie abstrakcji, nie było konieczna zmiana konfiguracji SLAM globalnego.

Ponadto wykorzystano węzeł cartographer occupancy grid, budujący na podstawie dopasowanych submap, pojedynczą globalną mapę kompatybilną z systemem ROS, zgodnie z dopasowaniami wygenerowanymi przez SLAM globalany.

Przetestowano również działanie algorytmu SLAM w wersji dwuwymiarowej, po przetworzeniu chmury punktów do dwuwymiarowego skanu. W takiej konfiguracji nie udało się uzyskać poprawnego działania lokalizacji, ze względu na zbyt niską częstotliwość i rozdzielczość skanów.

6.4.4. System nawigacji

Do zrealizowania zadania nawigacji w zakresie planowania ścieżki i jej wykonania, wykorzystano kompleksowe rozwiązanie, jakim jest zestaw oprogramowania NAV 2 [42]. Jest to rozbudowany zbiór bibliotek, umożliwiający pełną konfigurację mechanizmów sterowania robota w ramach systemu ROS2. Ogólny schemat systemu przedstawiono na rys. 6.11.



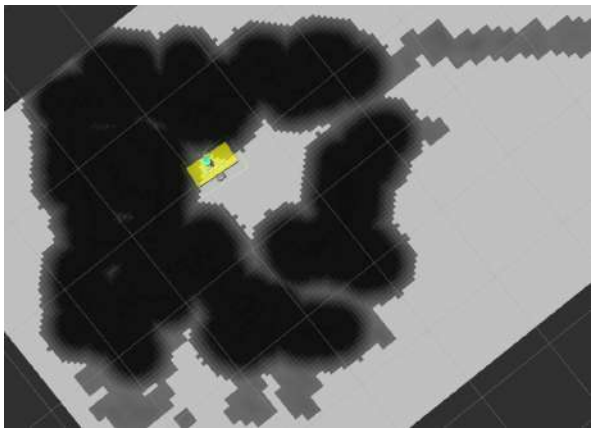
Rysunek 6.11. Schemat działania systemu NAV 2 [42]

W skład systemu nawigacji NAV2 wchodzą następujące narzędzia:

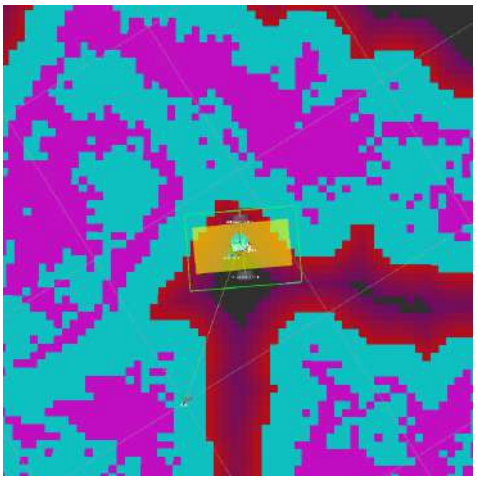
- Globalna mapa kosztu - system NAV 2 wykorzystuje dwuwymiarową mapę otoczenia. Skonfigurowana mapa składa się z trzech warstw:
  - — static layer - aktualizowana na podstawie mapy publikowanej przez Cartographer.
  - — obstacle layer - aktualizowana an podstawie przeszkód wykrywanych przez chmurę punktów, rzutowana do dwóch wymiarów
  - — inflation layer - warstwa powiększająca przeszkody by uniknąć kolizji

  W mapie skonfigurowano m.in. ślad robota - jako prostokąt opisujący platformę mobilną bez względu na wychylenie skrętu. Dostosowano również parametry odpowiedzialne za ograniczenie dystansu i wielkości przeszkód - przeszkody niższe niż 10cm traktowane są jako przejezdne. Przykładową zarejestrowaną mapę globalną przedstawiono na rys. 6.12
- Lokalna mapa kosztu - analogiczna do globalnej, jednak reprezentująca jedynie obszar dookoła robota. Składa się z dwóch warstw:
  - — voxel layer - trójwymiarowa, aktualizowana an podstawie przeszkód wykrywanych przez skaner. Reprezentuje zajętość wycinków widocznej przestrzeni w kształcie sześcianów.
  - — inflation layer - analogiczna do warstwy inflacji w mapie globalnej.

  Parametry mapy zostały skonfigurowane, podobnie do mapy globalnej. Przykładową zarejestrowaną mapę lokalną przedstawiono na rys. 6.13
- Planer - odpowiedzialny za planowanie globalnej ścieżki robota. Wykorzystuje infor-



Rysunek 6.12. Mapa globalna. System uruchomiony w przykładowym pomieszczeniu.



Rysunek 6.13. Mapa loklana. System uruchomiony w przykładowym pomieszczeniu.

macje z mapy globalnej. Zastosowany został niestandardowy planer SmacPlanner-Lattice [42]. Jest on znacznie lepiej przystosowany do sposobu lokomocji robota niż domyślny planer NavFn [42]. Wybrany planer dostosowuje trasę do kształtu robota - jest to istotne ze względu na podłużny kształt platformy mobilnej. Dostosowane zostały również wagi odpowiadające karom za wykonywanie ostrych zakrętów - dzięki temu generowana jest ścieżka o łagodnych zakrętach, odpowiednich dla układu kinematycznego robota - wówczas może być wykorzystany centralny przegub, co zapewnia większą precyzję wykonania ścieżki. Ograniczone do minimum jest również planowanie zakrętów w punkcie - odpowiednich dla zwykłych robotów różnicowych, ale obciążających zastosowaną platformę mobilną.

- Kontroler - odpowiedzialny za wyznaczenie ścieżki w skali lokalnej (planer loklany) i bezpośrednie wyznaczenie prędkości robota. Wykorzystuje wtyczkę DWBLocalPlanner [42]. Dostrojony został szereg parametrów planera, aby zapewnić jego kompatybilność z platformą mobilną. Między innymi dostosowano maksymalną prędkość do możliwości robota oraz ograniczono maksymalne przyspieszenia by zwiększyć płynność ruchu.
- Węzeł wygładzający prędkość (ang. velocity smoother) - odpowiedzialny za ograniczenie nagłych zmian prędkości, które mogłyby spowodować nagłe ruchy robota. Dostosowano do momentu uzyskania płynnych zmian prędkości.
- Drzewo zachowań (ang. Behaviour tree) - umożliwia sterowanie funkcjami robota na najwyższym poziomie abstrakcji, między innymi poprzez zadawanie celów i ścieżek, a także bardziej skomplikowanych zadań robotycznych. W niniejszym projekcie jest stosowane w ograniczonym zakresie, logika drzewa zachowań może być jednakże wykorzystana do rozbudowy i wykorzystania praktycznego całego systemu. Drzewo zachowań zostało wykorzystane jedynie do wyznaczania celu, a także obsługi zachować awaryjnych (rescue).

Ponadto wykorzystano węzeł nav2 costmap 2d markers, publikujący markery w przestrzeni oznaczające przeszkody, oznaczone względem mapy lokalnej. Markery wykorzystywane są jedynie do wizualizacji.

7. Wyniki i działanie

Testy pojedynczych modułów przeprowadzany były na bieżąco, w trakcie realizacji i po ich ukończeniu. Testy całości systemu przeprowadzono po ukończeniu pierwotnej fazy realizacji. Testy spowodowały pewne zmiany w podmodułach, które zostały uwzględnione w opisach ich realizacji.

7.1. Działanie skanera

Przetestowane została zarówno jakość skanów, jak i działanie mechanizmu skanera i samego czujnika.

Uzyskano skaner o pełnej rozdzielczości poziomej 1600 punktów na obrót, co odpowiada zmianie kąta o 0,225 stopnia pomiędzy pomiarami. Ponieważ do ruchu w pionie wykorzystany jest serwomechanizm przyjmujący impulsy PWM, rozdzielczość pionowa jest zależna od luzów w przekładni oraz minimalnej długości impulsu. Przetestowana została wartość rozdzielczości pionowej 91 punktów i kroku 1 stopnia. Wówczas każda zmiana o 1 stopień wywołuje widoczne przemieszczenie czujnika, możliwe jest jednak dalsze zwiększanie rozdzielczości.

Użyteczna rozdzielczość jest jednak powiązana z szybkością skanowania. Ze względu na szybkość czujnika, dla rozdzielczości 1600x91 czas trwania pojedynczego skanu wynosi około 2 minut. Aby czujnik był praktyczniejszy, można zredukować rozdzielczość. Wówczas potencjalna rozdzielczość praktyczna, uwzględniająca ograniczony kąt widzenia w pionie, wynosi 200 punktów w poziomie na 6 punktów w pionie przy częstotliwości 1.0 Hz, co odpowiada kątowi 1.8 stopnia na 6 stopni. Rozdzielczość pozioma jest większa, ponieważ przy omijaniu naturalnych przeszkód, jak np. drzewa jest ona istotniejsza. Przeszkody szerokie i niskie nie są tak powszechne.

Zakres skanowanych nachyleń określono w tym przypadku na od 70 stopni do 100 stopni. Kąt 70 stopni jest wystarczający, ponieważ mniejsze wartości powodują skanowanie podłoża blisko robota. (dla 70 stopni, odległość 137 cm w poziomie). Niesie to mało informacji o otoczeniu kosztem czasu pracy skanera. Podobnie zbędne jest skanowanie wysokich obiektów, które nie są przeszkodami dla robota. Skierowanie skanera ku górze pod dużym nachyleniem przy większych odległościach powoduje, że skanowane są punkty znajdujące się bardzo wysoko. Np. przy odległości w poziomie 15m i nachyleniu 110 stopni, skanowane będą punkty na wysokości 5,5m.

Stabilność skanera można uznać za zadowalającą. Wibracje podczas pracy z pełną prędkością są niewielkie (zmierzona amplituda drgań na szczycie wieży wyniosła 0.6 mm). Sprawdzono również poprawność działania skanera podczas występowania wibracji zewnętrznych (np. spowodowanych przez poruszanie się łazika po przeszkodach). Nie doprowadziły one do destabilizacji skanera. Podczas rzeczywistej jazdy wibracje są ograniczone ze względu na niską prędkość robota.

Testy w różnych warunkach oświetleniowych i na szerokim zakresie odległości pozwa-

lają ocenić, jak urządzenie może sprawować się w rzeczywistych warunkach. Badania zostały przeprowadzone bezpośrednio na samym sensorze, ponieważ oświetlenie nie ma wpływu na pozostałe elementy skanera.

Wyniki testów wykazały, że czas wykonywania skanu wydłuża się wraz ze wzrostem odległości. Gdy skaner nie napotyka przeszkody (np. gdy jest skierowany w pustkę), czas pomiaru wydłuża się jeszcze bardziej, ponieważ oprogramowanie układowe powtarza pomiary. Pewne nieregularności w czasie wykonywania skanu są akceptowalne, dzięki asynchronicznej naturze sterowania silnikiem krokowym. Czas pomiaru został ustabilizowany poprzez:

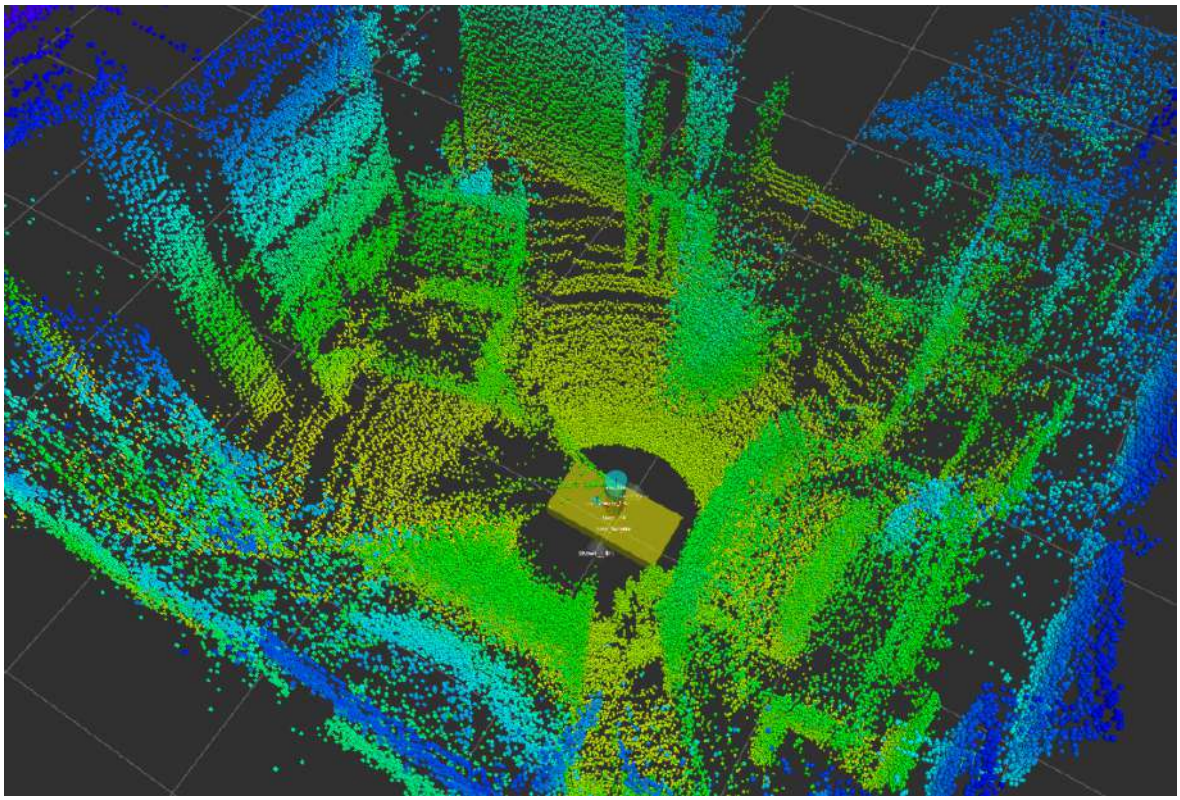
- Ograniczenie czasu maksymalnego przez zmniejszenie maksymalnej liczby powtórzeń przez autora w konfiguracji czujnika.
- Wydłużenie czasu minimalnego (gdy przeszkoda jest bliska i dobrze widoczna) przez wprowadzenie minimalnego czasu oczekiwania na pomiar w programie.
- Zapewnienie dodatkowych opóźnień stabilizujących czasy wykonywania pozostałych operacji

Dodatkowo, dokonano weryfikacji częstotliwości odczytu poprzez analizę widma amplitudowego dźwięku generowanego przez silnik krokowy. Największa amplituda rejestrowana była dla wartości odpowiadającej częstotliwości pomiarów (i jej wielokrotności). Stabilność tej częstotliwości oceniano natomiast na podstawie szerokości pasa odpowiadającego wysokim amplitudom. Szerszy pas sygnalizował mniejszą stabilność częstotliwości.

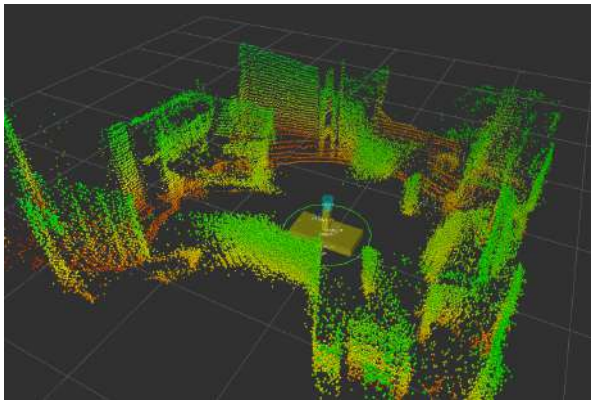
Zaobserwowano pewne zakłócenia przy skanowaniu błyszczących powierzchni, które odbijają wiązkę skanującą, oraz jasnych obiektów. Typowe błyszczące powierzchnie nie występują w środowiskach docelowych. Najpowszechniejsze błyszczące naturalne obiekty, zbiorniki wodne, są ignorowane w założeniach projektu. Przykładem bardzo jasnego obiektu jest słońce. Skierowanie czujnika w słońce lub powierzchnię odbijającą jego światło, powoduje pojedyncze błędne odczyty - mogą one być później odfiltrowane.

Następnie przeprowadzone zostały testy wyników skanowania. Poniżej zostały przedstawione wyniki skanowania dla różnych rozdzielczości (Rys 7.1 - pełny zakres nachyleń, 7.2 - 7.5 - ograniczony zakres), zwizualizowane za pomocą programu RVIZ2 zintegrowanego w systemie ROS2.

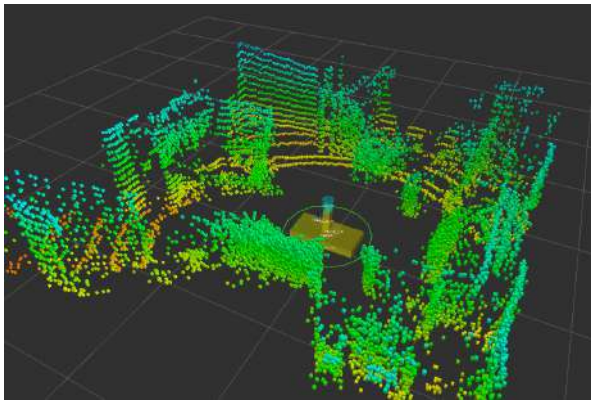
Wyniki odpowiadają topologii otoczenia. Skany w pełnej rozdzielczości zapewniają najwięcej szczegółów, jednakże są zbyt wolne, niezbędne jest więc ograniczenie rozdzielczości. Przy wybranej rozdzielczości 200 x 6 uzyskana częstotliwość 1 Hz odpowiada przejechaniu pomiędzy skanami przez robota, poruszając się z maksymalną prędkością, odległości 23 cm. O ile więc częstotliwość skanów jest wolna w stosunku do rozwiązań dostępnych na rynku, biorąc pod uwagę prędkość robota powinna być wystarczająca. W przeciwnym wypadku, możliwe jest dalsze ograniczenie prędkości robota lub zmniejszenie rozdzielczości. Jeżeli chodzi o rozdzielczość skanów, powinna być ona wystarczająca. W artykule "Autonomous Person Following with 3D LIDAR in Outdoor Environments"[19] wykorzystano skaner o zbliżonej rozdzielczości 59 x 29.



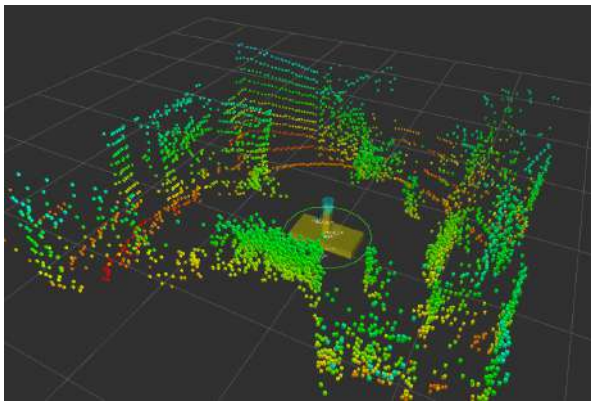
Rysunek 7.1. Test przy pełnej rozdzielczości (1600x91), nachylenia od 45 do 135 stopni



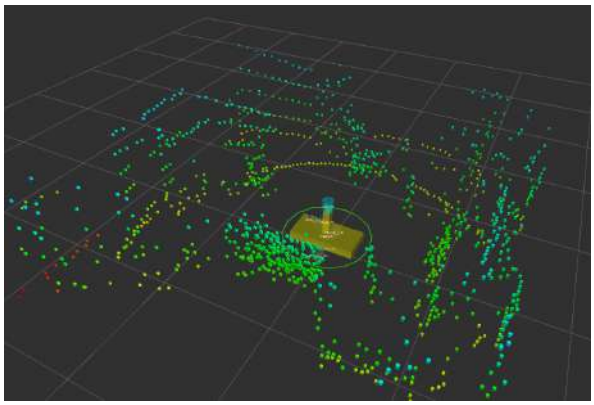
Rysunek 7.2. Test przy rozdzielczości (1600x31), nachylenia od 70 do 100 stopni



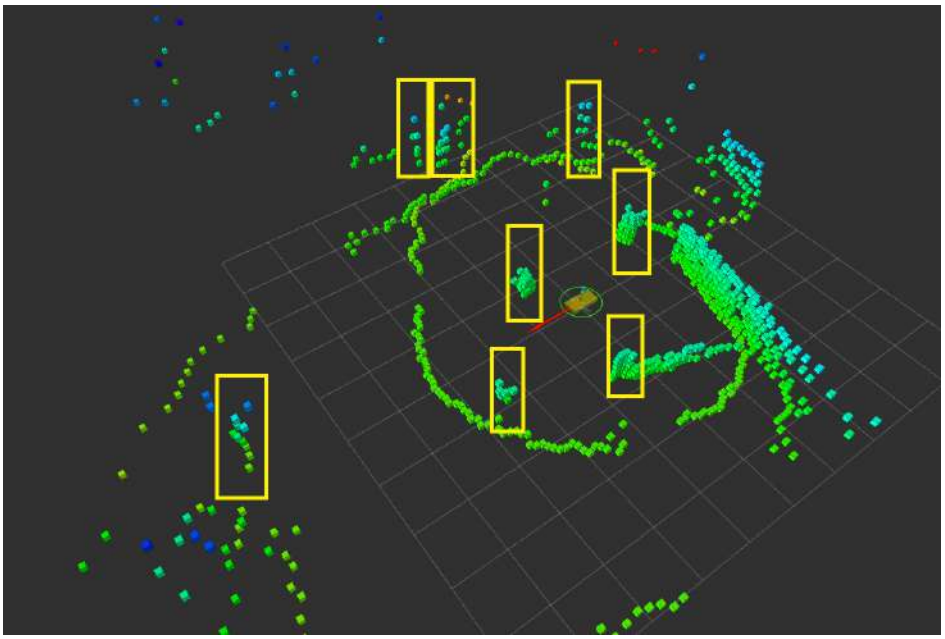
Rysunek 7.3. Test przy rozdzielczości (800x16), nachylenia od 70 do 100 stopni



Rysunek 7.4. Test przy rozdzielczości (400x11), nachylenia od 70 do 100 stopni



Rysunek 7.5. Test przy rozdzielczości (200x6), nachylenia od 70 do 100 stopni



Rysunek 7.6. Test terenowy skanera; oznaczone drzewa

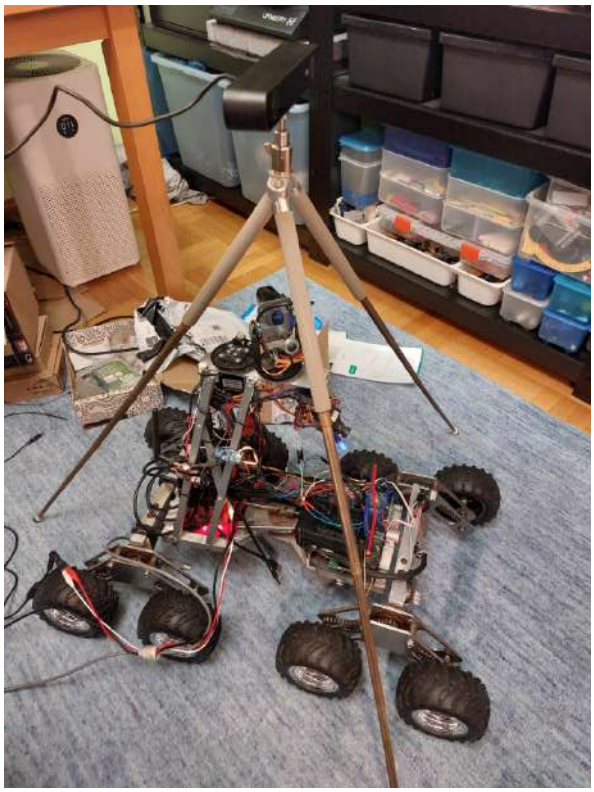
Zaobserwowano pewne odchylenia przy skanowaniu płaskich powierzchni - punkty w wizualizacji układają się w płaszczyznę, ale widoczne są niedokładności, zależne od m.in. koloru powierzchni i kąta nachylenia wiązki względem powierzchni. Niedokładności bezwzględne oszacowano na podstawie wyników na poniżej ± 5 cm.

Ponadto przetestowano działanie skanera w terenie (rys. 7.6). Rozdzielczość pionowa była wystarczająca do wykrycia wszelkich przeszkód. Rozdzielczość pozioma wystarczała do jednoznacznego zarejestrowania przeszkód w postaci drzew do odległości około 6m (oznaczone punkty odpowiadające pionowym przeszkodom). Duże odległości, nie występujące we wcześniejszych testach, nie zaburzyły procesu skanowania.

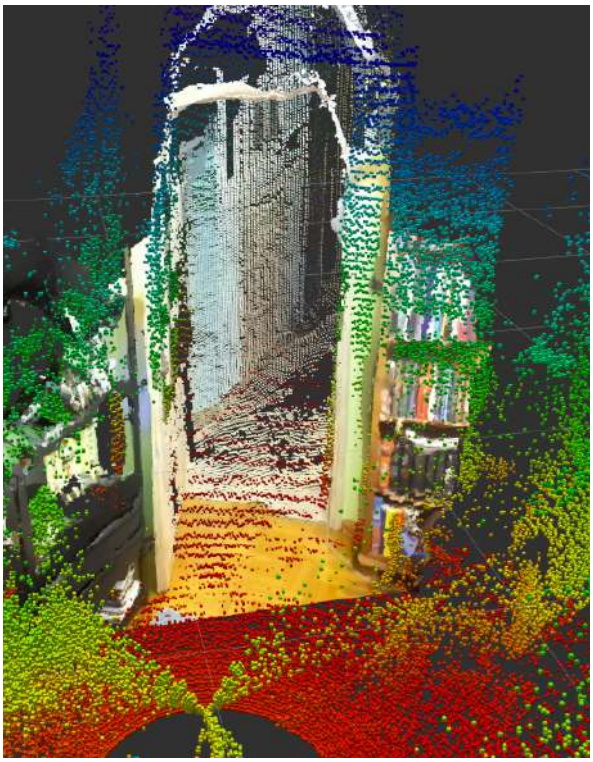
Ostatecznym testem było porównanie działanie skanera do zewnętrznego, referencyjnego urządzenia. Wybrano kamerę głębi StereoLabs ZED 2 [11]. Kamera została umieszczona w osi skanera, na statywie, bez względnej rotacji. Oba urządzenia podłączono do tej samej wizualizacji. Względne przesunięcie układów zostało uwzględnione. Konfigurację testową przedstawiono na rys. 7.7. Wyniki skanu (rys. 7.8 - 7.12)zostały porównane do kamery w dwóch konfiguracjach: pełnej rozdzielczości 1600x91 oraz wykorzystywanej rozdzielczości 200x6. Chmura punktów ze skanera wyświetlona została w skali kolorów w osi z. Wyniki z kamery głębi są reprezentowane kolorami obserwowanymi przez kamerę.

Podstawową różnicą między sensorami są rozdzielczość (znacznie wyższa w kamerze) i kąt widzenia (dookólny dla skanera). Istotniejsza w powyższym teście jest jednak dokładność pomiaru. We wszystkich testowanych widokach punkty rejestrowane pokrywają się z wynikami chmury punktów bez widocznych odchyleń.

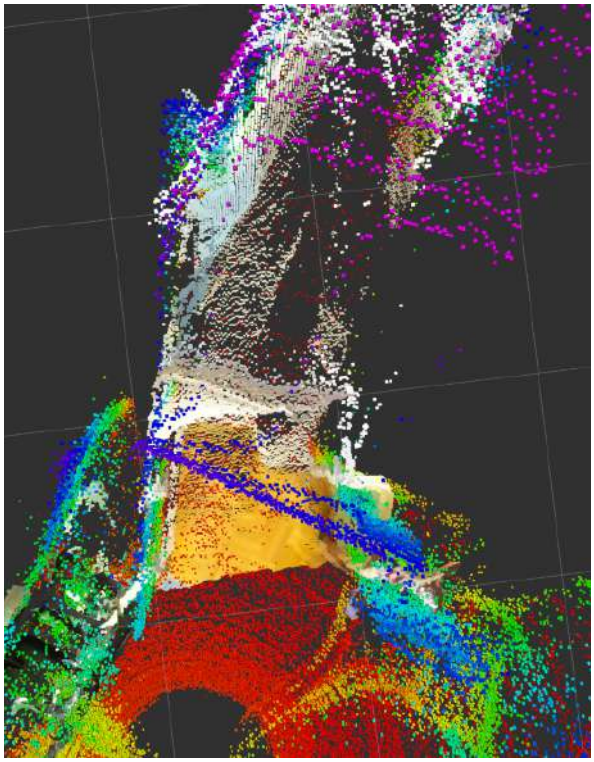
Największa różnica może być zaobserwowana w teście 7.10 - podłoga powinna być pozioma, jednakże przez kamerę rejestrowana jest pod nachyleniem - jest to rezultat małej



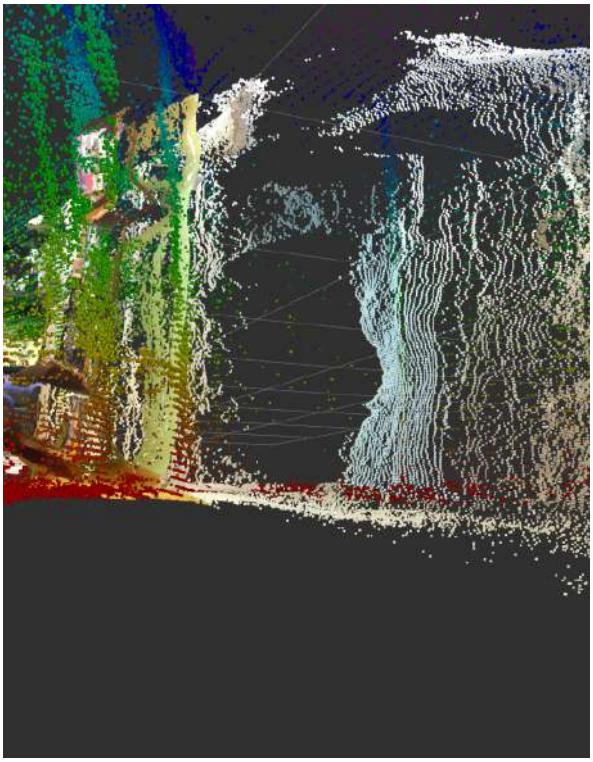
Rysunek 7.7. Konfiguracja testu porównania sensorów



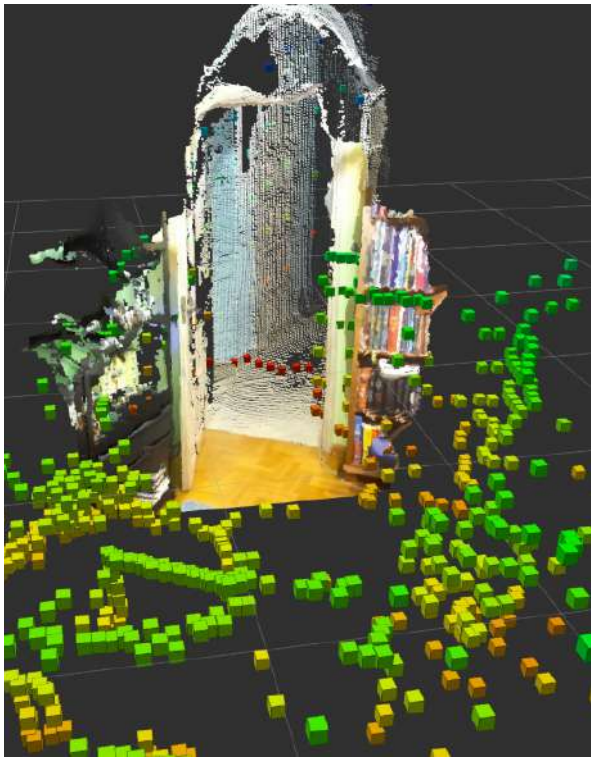
Rysunek 7.8. Test przy pełnej rozdzielczości, widok perspektywiczny



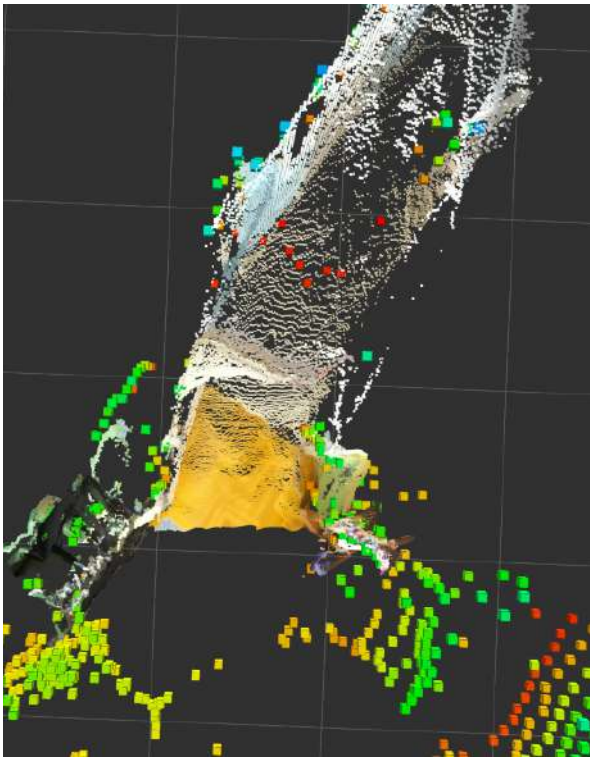
Rysunek 7.9. Test przy pełnej rozdzielczości, widok z góry



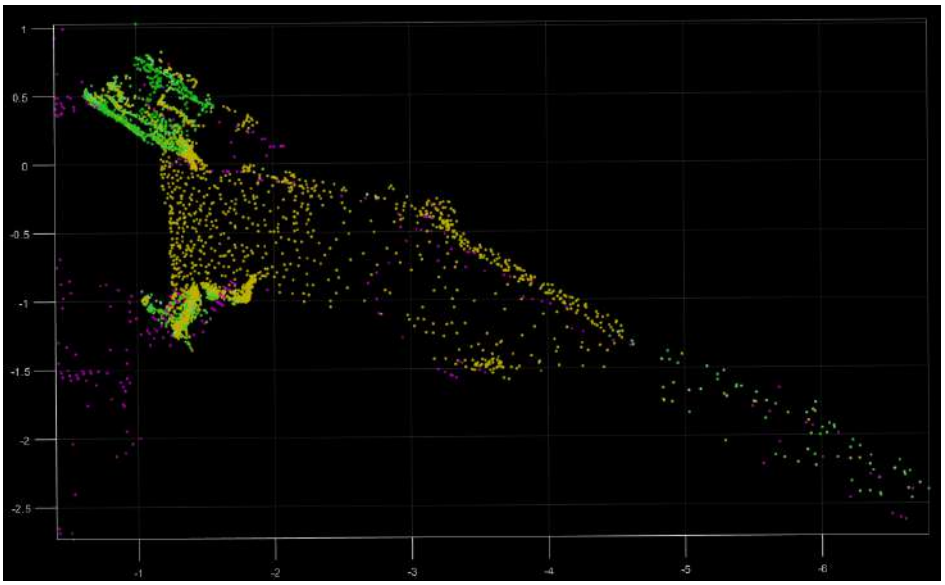
Rysunek 7.10. Test przy pełnej rozdzielczości, widok z boku



Rysunek 7.11. Test przy ograniczonej rozdzielczości, widok perspektywiczny



Rysunek 7.12. Test przy ograniczonej rozdzielczości, widok z góry



Rysunek 7.13. Porównanie skanów w MatLab

ilości detali widocznych dla kamery, zaburzający działanie algorytmów kamery. LiDAR działa tu poprawnie.

Aby potwierdzić jakość skanera numerycznie, dopasowano chmury wygenerowane przez skaner i kamerę głębi w programie MatLab za pomocą algorytmu Iterative Closest Point (pcregistericp [43]). Algorytm podczas dopasowania zwrócił wartość błędu średniokwadratowego dopasowania - odpowiadającą jakości dopasowania, co jest powiązane z podobieństwem skanów z obu urządzeń. Warto zaznaczyć, że wartość błędu wzrasta bez względu na to które z urządzeń myli się z rzeczywistością. Uzyskano wartość błędu RMSE = 0.25m.

Podsumowując, dokładność skanera jest satysfakcjonująca i nie odbiega od standardów rozwiązań komercyjnych. W zależności od charakterystyki obserwowanych powierzchni, możliwe jest osiągnięcie jeszcze wyższej dokładności. Warto zauważyć, że zarówno rozdzielczość, jak i częstotliwość odczytów są znacznie niższe w porównaniu do kamery ZED2 (rozdzielczość 2208x1242 przy częstotliwości 15 Hz). Różnica ta wynika z fundamentalnie innej zasady działania sensora skanera, gdzie sensor odpowiada pojedynczemu pikselowi kamery.

Ze względu na niską prędkość platformy mobilnej (maksymalna prędkość 0.23 m/s), niższa częstotliwość skanera nie stanowi ograniczenia dla całego systemu. Dodatkowo, umożliwia to przeprowadzanie obliczeń przy ograniczonej mocy obliczeniowej. W algorytmie SLAM przy sensorach wysokiej rozdzielczości przeprowadzany jest subsampling, więc wszystkie informacje nie byłyby wykorzystane.

Co więcej, skaner wykazuje się większą dokładnością przy obserwacji obiektów ruchomych i oświetlanych zmiennie, takich jak roślinność czy śnieg. To sprawia, że jest on bardziej efektywny w naturalnych środowiskach. LiDAR nie wymaga również oświetlenia do działania - wykorzystuje jedynie emitowaną przez skaner wiązkę. Poprawność i dokładność wskazań kamery są natomiast bezpośrednio powiązane z kierunkiem i jasnością oświetlenia. W ciemności kamera nie funkcjonuje.

7.2. Wydajność platformy mobilnej

Zostały przeprowadzone testy działania pojedynczych czujników, oraz oceniono możliwości platformy mobilnej.

Przy domyślnej konfiguracji, uzyskano częstotliwość odczytu 100 Hz z czujnika IMU. Wartość ta jest w pełni wystarczająca dla funkcjonowania systemu. Po kalibracji, zwracane wartości są zgodne z oczekiwaniami - bliskie zera wartości prędkości kątowej gdy czujnik jest statyczny, przyspieszenia zgodne z ziemskim. Wskazania kierunku północnego zostały porównane z innym kompasem cyfrowym i nie różniły się znacząco.

Największą wadą wykorzystanego odbiornika GNSS jest czas uzyskania ustalonej pozycji po włączeniu, wynoszący około 30s. Ponadto nie możliwe jest uzyskanie pozycji we wnętrzach, nie jest to jednak problem przy terenowym zastosowaniu robota. W nominalnych warunkach odbiornik działa poprawnie. Częstotliwość odczytu wynosi 4 Hz, co

Rysunek 7.14. Test nachylenia

Rysunek 7.15. Test na śniegu

jest w pełni wystarczające biorąc pod uwagę dokładność systemu GNSS. Przy dobrych warunkach odchylenie standardowe pozycji, odczytywane przez moduł GPS, wynosi 1-2m w otwartym terenie.

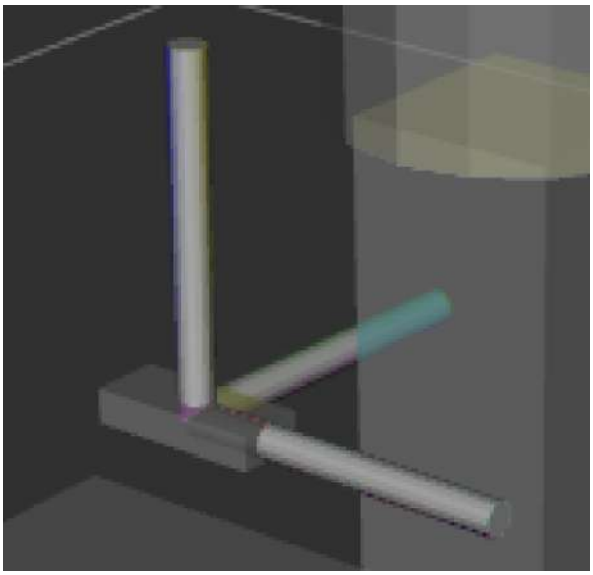
Opracowane sterowniki umożliwiają stabilną komunikację z warstwą niskopoziomową, oraz dostarczają standaryzowane interfejsy wewnątrz systemu ROS, pozwalające na rozbudowę warstwy wysokopoziomowej.

Możliwości terenowe samej platformy zostały ocenione poprzez serię testów przeprowadzonych w różnych warunkach terenowych. Przetestowano nachylenia terenu do 45°, różne nawierzchnie, takie jak ziemia, żwir i śnieg, oraz przeszkody o wysokości równej średnicy koła (do 10 cm). Platformę w warunkach testowych przedstawiono na rys. 7.14, 7.15. Mechanicznie platforma z powodzeniem sprostała tym wyzwaniom. Ze względu na wysoki moment obrotowy napędów, prędkość kół przy tej samej zadanej wartości nie odbiegała w sposób znaczący od prędkości podczas jazdy po płaskim terenie.

Brak enkoderów jest pewnym ograniczeniem platformy. Wartość danych z enkoderów podczas jazdy po nieregularnym terenie byłaby jednak niewielka - często występują poślizgi kół oraz dystans przejechany po przeszkodach jest zauważalnie większy niż odpowiadający mu dystans przebyty w poziomie. Konieczne natomiast będzie wyznaczenie współczynnika określającego zależność prędkości rzeczywistej od zadanej.

7.3. Testy odometrii

Precyzję wyznaczania samej orientacji przetestowano przemieszczając rzeczywistego robota i obserwując wizualizację orientacji wyznaczonej przez filtr komplementarny. Proces ten umożliwił ocenę zdolności filtru do skutecznego śledzenia zmian położenia i orientacji robota w czasie rzeczywistym. Nie zaobserwowano dryfu orientacji w spoczynku, i po przeprowadzeniu testów (Rys. 7.16). Nie zaobserwowano opóźnień w odwzorowaniu pozycji. Zaaranżowana w rzeczywistości orientacja odpowiadała wizualizowanej.

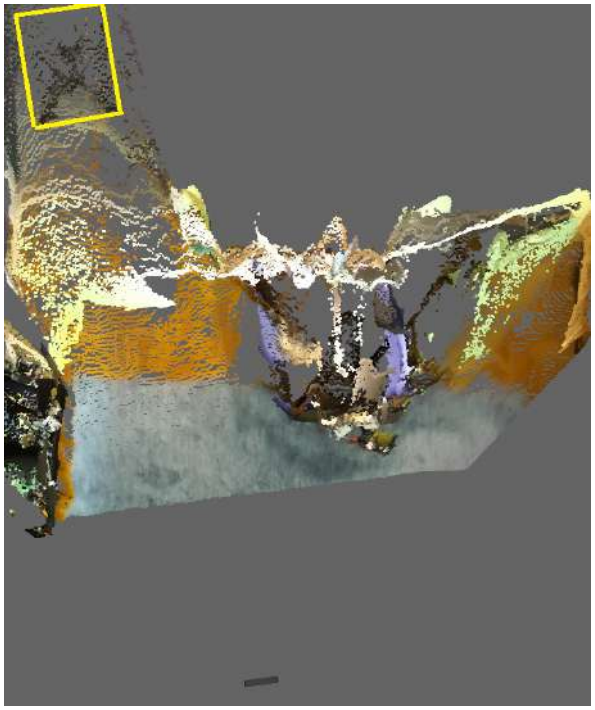


Rysunek 7.16. (niebieski, zielony, czerwony) - orientacja przed testem; (żółty, magenta, cyjan) - orientacja po teście; biały - pokrycie

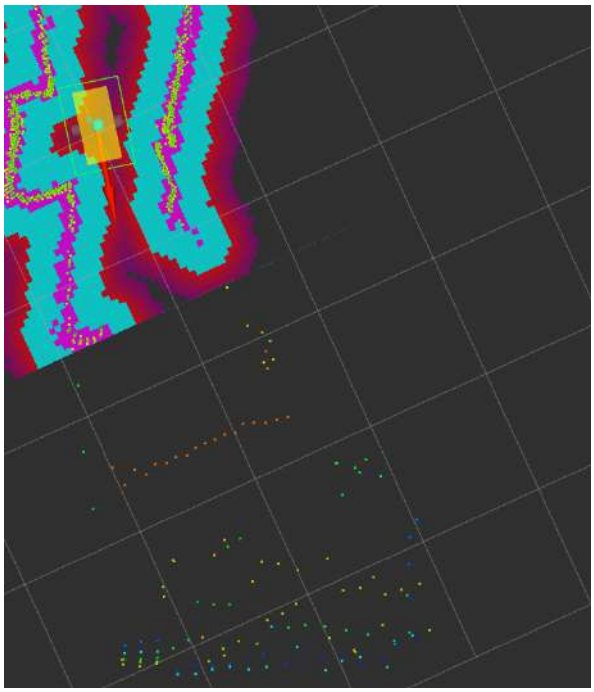
Gdy uruchomiony był jednocześnie skaner LiDAR, rejestrowane były oscylacje robota, zgodne z rzeczywistymi. Dowodzi to dobremu odwzorowaniu orientacji przy zmianach o wysokiej częstotliwości.

W przypadku odometrii należy przetestować dokładność odwzorowania lokalizacji robota przy rzeczywistym ruchu. W ramach testu zostały zadane kolejno prędkości, jako wiadomości na temacie /cmd_vel. Porównano zmianę współrzędnych, zmierzoną w rzeczywistości, ze zmianą współrzędnych w wizualizacji. Układ odom ustawiono jako statyczny, aby wykluczyć wpływ lokalizacji globalnej. Do referencyjnego pomiaru wykorzystano obraz z kamery głębi zed 2. Oba widoki wyświetlano w widoku rzutu ortogonalnego, ze statyczną kamerą. Wynik testu przedstawiono na rys. 7.17 - 7.20. Żółtą ramką oznaczono robota na kamerze. Czerwone strzałki oznaczają trajektorię robota w wizualizacji.

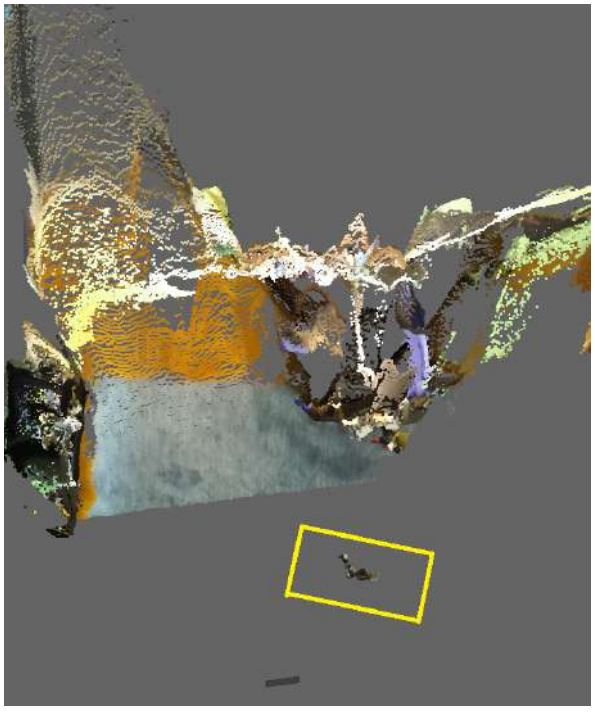
Zmiana położenia zmierzona w rzeczywistości wyniosła 150cm w poziomie i 290cm w pionie (w układzie widoku). W odometrii przebyta odległość wyniosła 115cm w poziomie i 236cm w pionie. Oznacza to błąd względny odometrii na poziomie 24%. Przejechana odległość była większa niż zmierzona. Błąd orientacji nie wystąpił (znalazł się poniżej dokładności pomiarowej). Przy jeździe terenowej mogą nastąpić dodatkowe poślizgi - wówczas może wystąpić odwrotna niedokładność - przejechana odległość będzie mniejsza niż zmierzona. Eksperyment wykazał, że precyzja jest ograniczona, ze względu na brak enkoderów i występowanie poślizgu kół przy skręcaniu. Uzyskana dokładność jest jednak akceptowalna, gdyż lokalizacja w systemie jest korygowana na bieżąco przez algorytm SLAM.



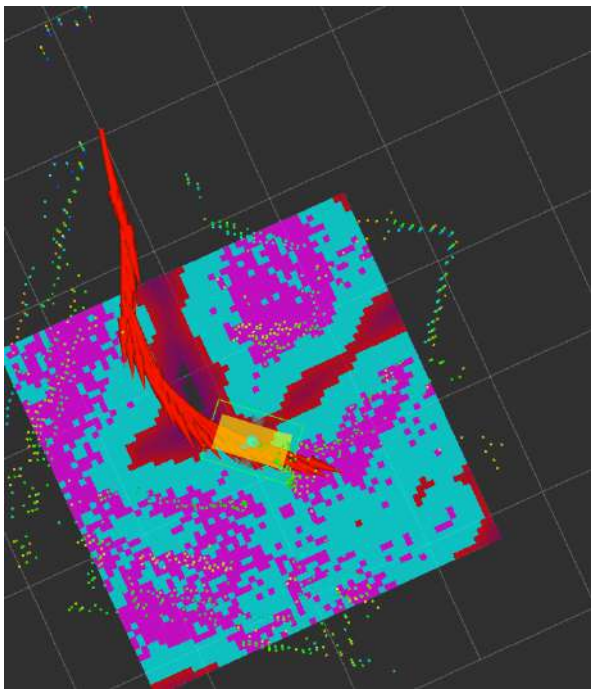
Rysunek 7.17. Widok z kamery przed testem



Rysunek 7.18. Wizualizacja przed testem



Rysunek 7.19. Widok z kamery po teście



Rysunek 7.20. Wizualizacja po teście

7.4. Testy SLAM

Weryfikacja działania algorytmu SLAM została przeprowadzona z wciąż włączoną odometrią, ponieważ SLAM został skonfigurowany z założeniem ograniczonej rotacji pomiędzy skanami - orientacja jest określana precyzyjnie przez lokalizację lokalną. Wykorzystanie samego SLAM skutkowałoby więc licznymi błędami. Przeprowadzono przejazd robotem przy zewnętrznym sterowaniu prędkością. Ponownie zastosowano procedurę porównawczą z kamerą głębi. Do testu wyłączono odbiornik GNSS, ponieważ został przeprowadzony w pomieszczeniu. Wynik testu przedstawiono na rys. 7.21 - 7.23. Na rys. 7.22 przedstawiono mapę nałożoną na rzut środowiska. Kolory mapy zostały dostosowane w celu poprawy czytelności.

W wyniku przeprowadzonego testu uzyskano mapę, która zadowalająco odzwierciedlała rzeczywistość. W trakcie tego testu nie stwierdzono błędów polegających na wykrywaniu nieistniejących przeszkód lub ignorowaniu istniejących przeszkód. W ramach możliwości porównania mapy do widoku z kamery głębi, odwzorowanie okazało się precyzyjne - elementy środowiska znajdujące się zarówno na mapie jak i widoczne w porównawczej kamerze pokrywają się. Widoczne są natomiast pewne błędy w obrazie generowanym przez kamerę - ściana w prawej części mapy nie została zarejestrowana poprawnie i ma kształt łuku.

7.5. Testy planowania

Test planowania przeprowadzono wyznaczając w zmapowanym środowisku cel dla robota, następnie obserwując wygenerowaną trajektorię (rys. 7.24).

Uzyskana trajektoria omija przeszkody - dzięki zastosowaniu inflacji mapy, trajektoria jest odsuwana na bezpieczny dystans od przeszkód, gdy jest to możliwe. Ponadto trajektoria jest wygładzona i nie zawiera ostrych zakrętów lub zakrętów w punkcie, co zwiększa precyzję pokonania jej przez robota.

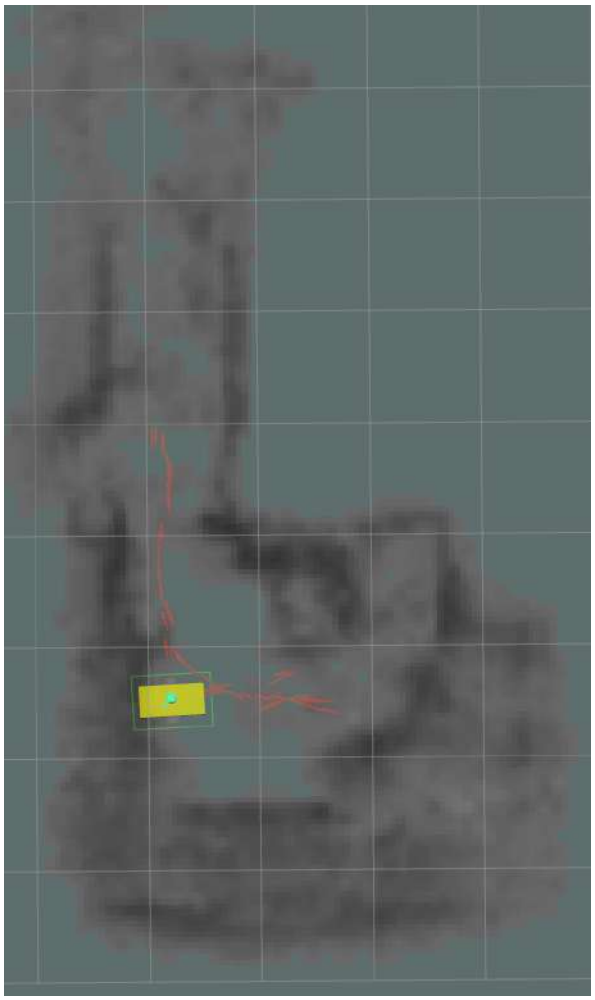
Przeprowadzenie wydzielonych testów kontrolera (planera loklanego) jest problematyczne, gdyż wymaga działającej lokalizacji, planera, i przekazania sterowania kontrolerowi. Testy te zostały więc przeprowadzone w ramach testu całości systemu.

7.6. Testy kompletnego systemu

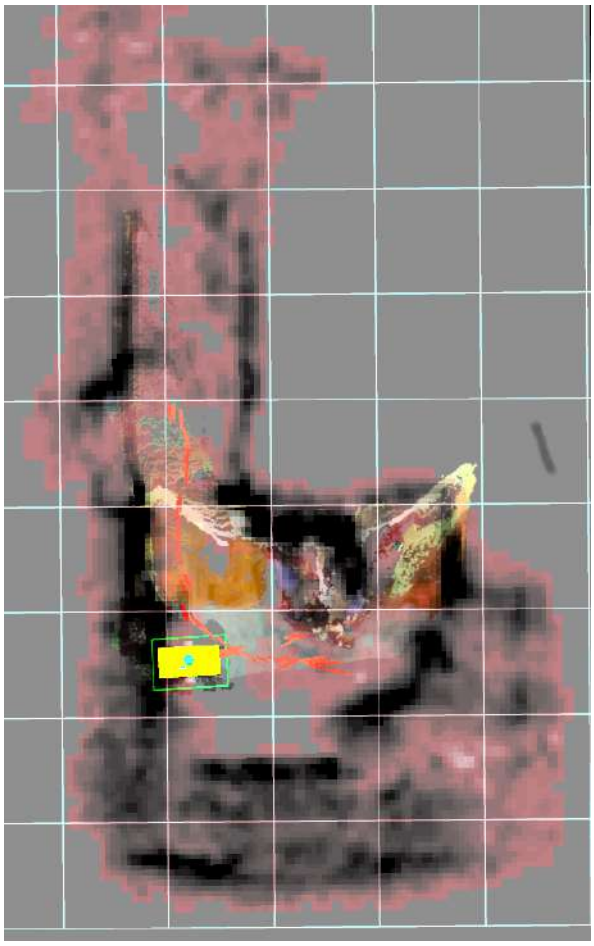
7.6.1. Środowisko i procedura testowa

Testy zostały przeprowadzone w naturalnym środowisku, reprezentującym docelowe miejsce pracy robota. Teren testowy przedstawiono na rys. 7.25 i 7.26. Współrzędne obszaru to 52,583136N; 21,442931E. W środowisku występują następujące elementy:

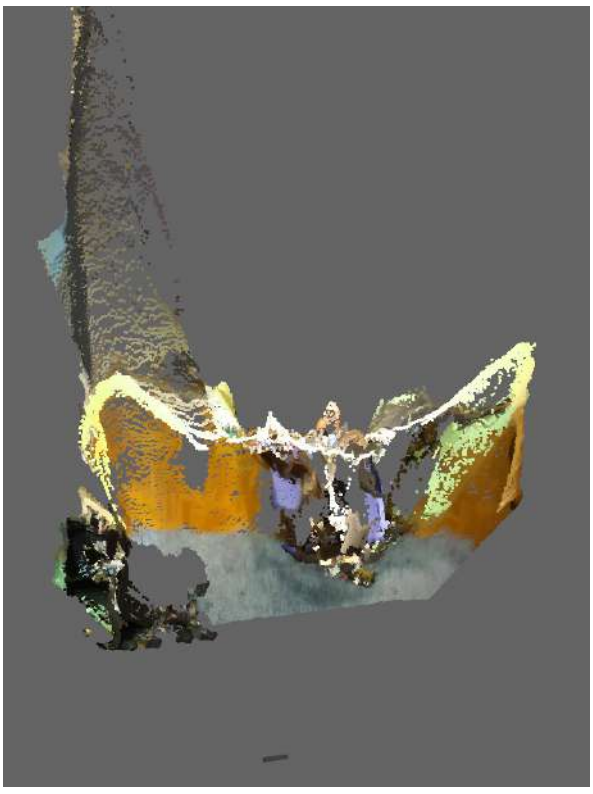
- Nierówny teren o zmiennym nachyleniu
- Trudna nawierzchnia (w czasie testów występowała pokrywa śnieżna)
- Sztuczne przeszkody (płot, postawione przeszkody)
- Naturalne przeszkody (zarośla, drzewa)



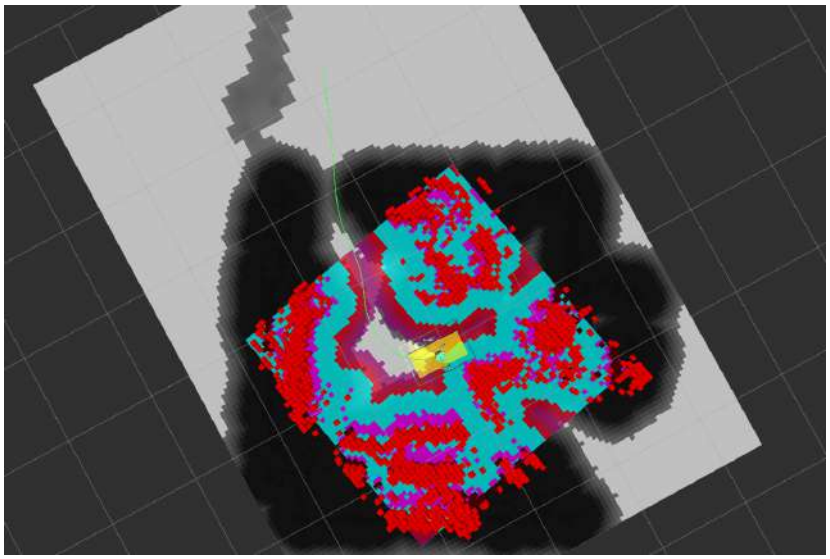
Rysunek 7.21. Wygenerowana mapa środowiska z widoczną trajektorią robota podczas mapowania



Rysunek 7.22. Porównanie mapy ze środowiskiem



Rysunek 7.23. Mapowane środowisko (widok z kamery głębi)



Rysunek 7.24. Planowana trajektoria (zielona linia)

Rysunek 7.25. Środowisko testu terenowego

- Naturalne oświetlenie dzienne

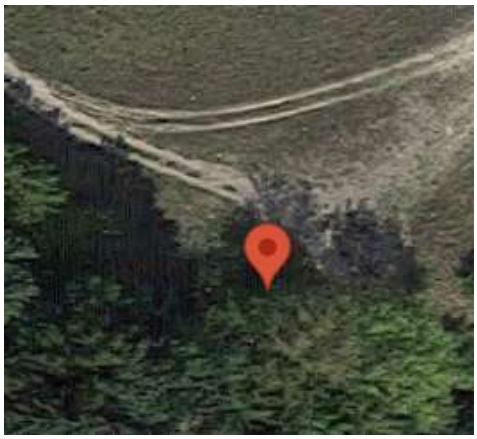
Powyższe elementy składają się na odpowiednie środowisko do testów robota terenowego. Do wykonania zadania niezbędne jest prawidłowe funkcjonowanie wszystkich systemów.

Przyjęta została następująca procedura testowa:

- Ustawienie robota na pozycji startowej, przed przeszkodami.
- Wybranie pozycji docelowej, znajdującej się za przeszkodami, uniemożliwiającej pokonanie trasy prostą ścieżką.
- Zadanie pozycji docelowej w programie RVIZ za pomocą funkcji 2D nav goal.
- Do czasu pokonania trasy przez robota, nie są przesyłane żadne komendy.

Przejazd został zarejestrowany zewnętrzną kamerą. Zarejestrowano również wizualizację systemu w RVIZ, w celu porównania na bieżąco stanu systemu z rzeczywistością. W wizualizacji zastosowano następujące oznaczenia, zgodnie ze standardem ROS:

- Mapa globalna - skala szarości, proporcjonalna do kosztu



Rysunek 7.26. Mapa satelitarna terenu testowego

- Mapa lokalna - koszt malejący w kolejności magenta -> cyjan -> czerwony
- Planowana trajektoria - zielona linia
- Robot - żółty

7.6.2. Wyniki

Test przeprowadzono zgodnie z ustaloną procedurą na wyznaczonym obszarze testowym. Pełny zarejestrowany przejazd dostępny jest jako materiał wideo [44]. W niniejszej pracy, materiał został przedstawiony w sposób poklatkowy na rys. 7.27.

Test wykazał zdolność robota do autonomicznej nawigacji w nieregularnym terenie, zgodnie z celem pracy. Wszystkie przeszkody zostały ominięte, poprawnie zastosowano planowanie. W trakcie testu wygenerowana została mapa terenu, odzwierciedlająca występujące przeszkody. Nie zaobserwowano dryfu lokalizacji robota ani deformacji mapy w trakcie przejazdu. Wyniki były zgodne z oczekiwanymi po przeprowadzeniu cząstkowych testów podsystemów.

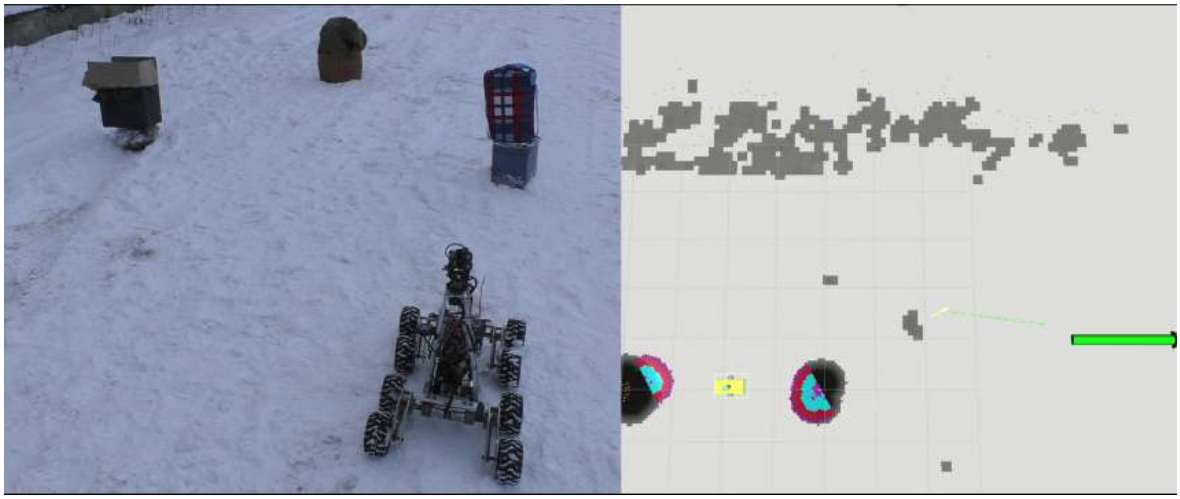
Przeprowadzono również testy systemu w innych środowiskach.

Przejazd przez las (rys. 7.28, 7.29) polegał na pokonaniu terenu bez sztucznych przeszkód, w pobliżu drzew. Mimo braku wyraźnych, bliskich przeszkód, robot lokalizował się poprawnie. Na mapie widoczne są pojedyncze wykryte pnie drzew.

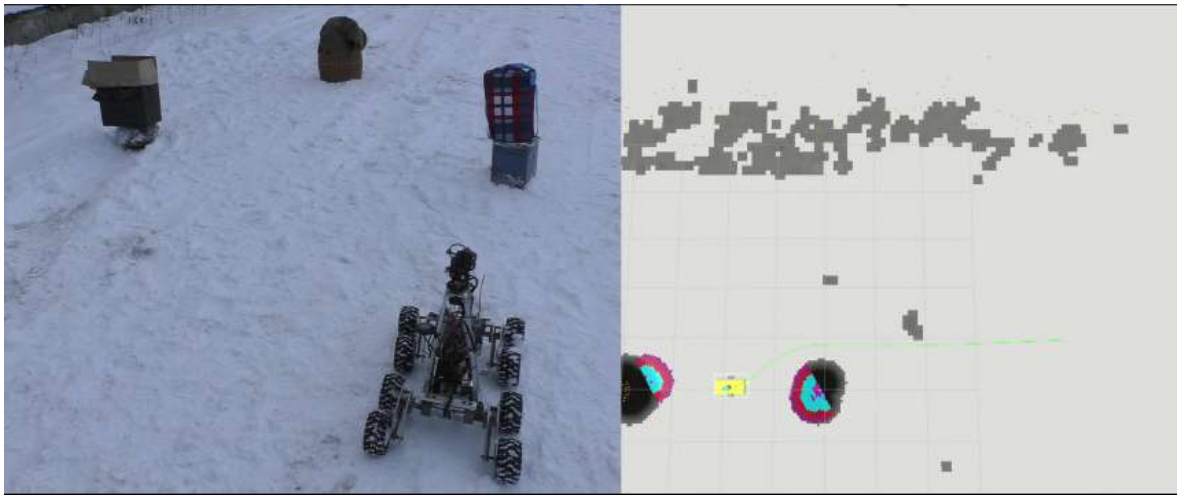
Test we wnętrzu (wykonanie trajektorii przedstawionej na rys. 7.24) obejmował przejazd wąskim przejściem, blisko przeszkód. Jego celem było wyznaczenie minimalnej szerokości przesmyku dla robota. Ustrukturyzowane środowisko wewnętrznego sprzyjało wysokiej dokładności lokalizacji. W takich warunkach wymagana jest jednak wysoka zwrotność od robota. Robot pokonał wyznaczoną trasę, z najwęższym przejściem o szerokości 80cm. Planer lokalny nie wyznaczył ścieżki przez przejście o szerokości 70cm, uznając je za zablokowane. Szerokość robota z kołami wynosi 41cm, więc pokonanie takiej przeszkody wymagałoby dużej precyzji i może być uznane za ryzykowne.

W ogólności, algorytm SLAM może napotkać problemy z działaniem, gdy środowisko jest pozbawione szczegółów - na płaskim terenie brak punktów charakterystycznych

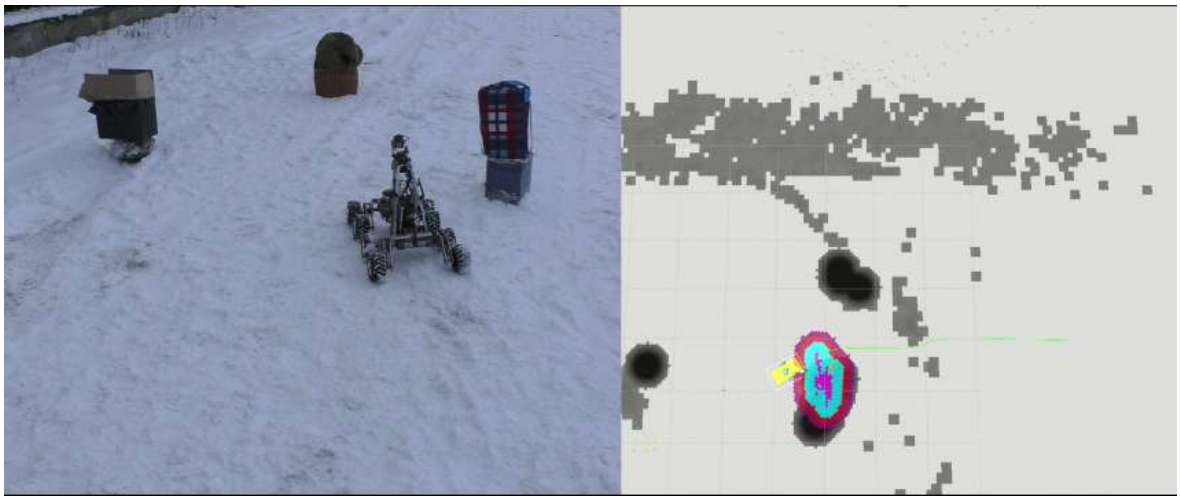
Rysunek 7.27. Test nawigacji terenowej



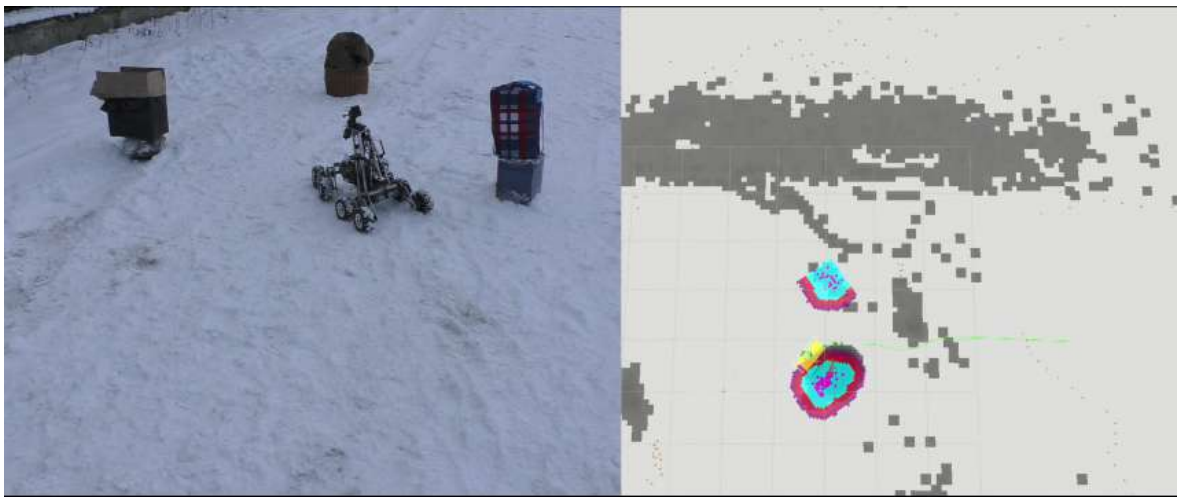
(a) Uruchomienie systemu. Zadany cel (zielona strzałka). Przeszkody są widoczne na wygenerowanej mapie globalnej



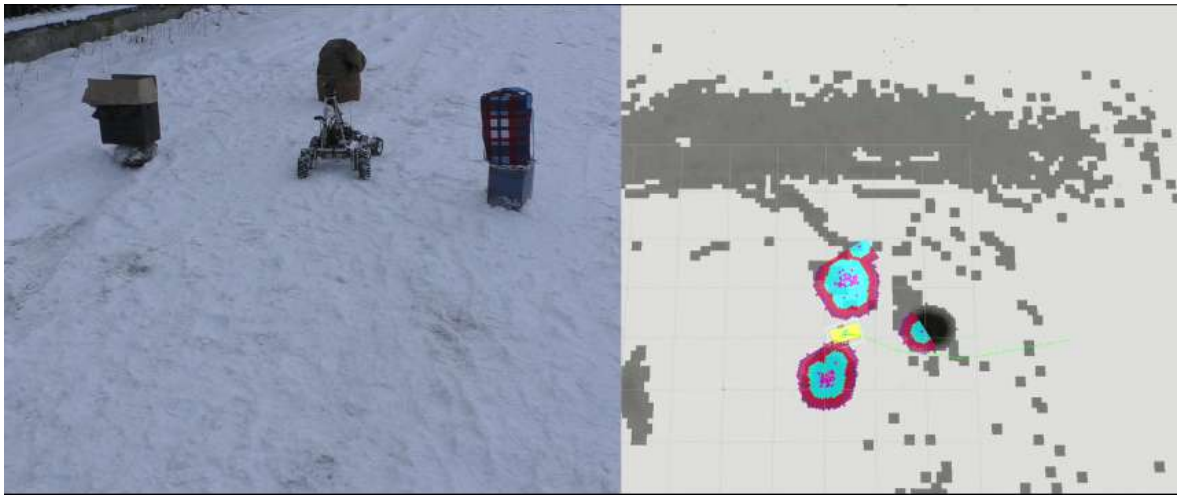
(b) Planer wyznaczył trajektorię (zielona linia) omijającą pierwszą przeszkodę.



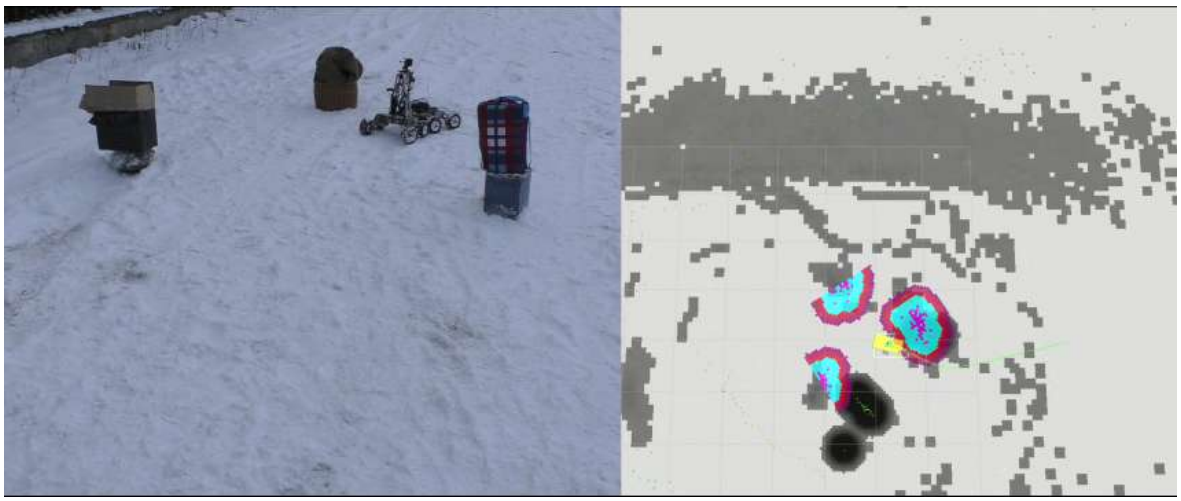
(c) Dotarcie do pierwszej przeszkody



(d) Ominięcie pierwszej przeszkody. Druga przeszkoda widoczna na mapie lokalnej.



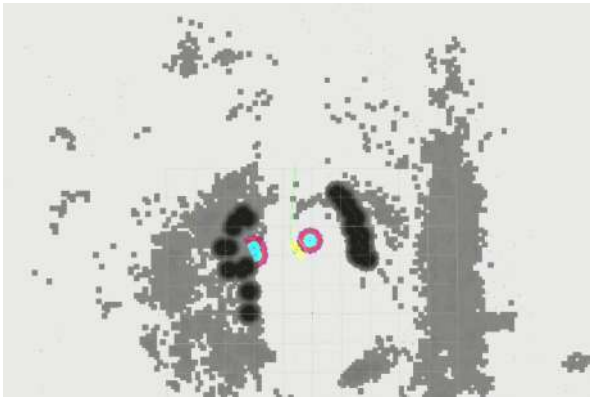
(e) Trzecia przeszkoda widoczna na mapie lokalnej. Dostosowanie trasy do ominięcia jej z zachowaniem bezpiecznego odstępu (inflacja)



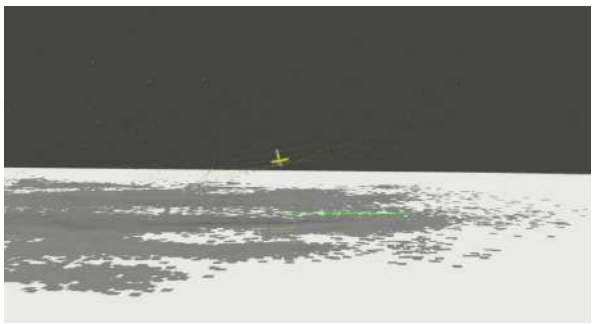
(f) Ominięcie trzeciej przeszkody

(g) Dotarcie do zadanego celu

Rysunek 7.28. Przejazd przez las



Rysunek 7.29. Wizualizacja przejazdu przez las



Rysunek 7.30. Przejazd po nachylonym terenie

uniemożliwia poprawne dopasowanie. Wówczas dane o lokalizacji globalnej mogą być dostarczone przez moduł GNSS - w efekcie robot może poruszać się po dowolnym terenie. Dokładność GNSS jest jednak mniejsza niż algorytmu SLAM.

Teren o zmiennym nachyleniu jest poprawnie rejestrowany przez robota (rys. 7.30). Odometria rejestruje zmianę współrzędnej z gdy robot przemieszcza się, znajdując się pod nachyleniem względem płaszczyzny poziomej. Algorytm SLAM również dokonuje korekty wysokości podczas dopasowywania skanów. W programie RVIZ jest jednak możliwe zwizualizowanie jedynie rzutu mapy globalnej na płaszczyznę poziomą. Planowanie globalne również odbywa się w dwóch wymiarach. Planowanie lokalne odbywa się natomiast w płaszczyźnie robota. W efekcie system działa poprawnie przy nachylonym terenie, jednakże nie jest możliwe zadanie celu o wybranej współrzędnej z, co czyni robota niezdolnym do nawigacji w trójwymiarowych strukturach jak np. jaskinie.

7.6.3. Obciążenie systemu

Zmierzono obciążenie Raspberry Pi 4 podczas obsługi robota. Uzyskano obciążenie wszystkich 4 rdzeni procesora na poziomie 85% oraz zużycie pamięci ram poniżej 500MB. Wydajność jest więc wystarczająca, jednakże przy dodaniu dodatkowych funkcjonalności procesor mógłby być zbyt wolny.

Wykorzystana w robocie bateria wystarcza na około 6h pracy w spoczynku, lub 1h jazdy. Możliwa jest instalacja 2 baterii w celu wydłużenia czasu pracy.

8. Podsumowanie

Podsumowując, główny cel utworzenia autonomicznego systemu dla robota mobilnego własnej konstrukcji, zdolnego do nawigacji w nieustrukturyzowanych środowiskach został osiągnięty. Poszczególne etapy projektu zakończyły się sukcesem:

- Uzyskano precyzyjny skaner LiDAR
- Poprawnie zaimplementowano dodatkowe czujniki na platformie mobilnej
- Utworzono sterowniki umożliwiające sterowanie robotem z wykorzystaniem ROS2
- Robot jest zdolny do wyznaczania lokalizacji lokalnej w sposób ciągły z zadowalającą precyzją
- Robot jest zdolny do wyznaczania lokalizacji globalnej i tworzenia stabilnej mapy
- System umożliwia zaplanowanie trasy omijającej przeszkody do wybranego celu i jej wykonanie

Działanie systemu zostało przebadane w warunkach naturalnych, reprezentujących rzeczywiste środowisko pracy robota.

8.1. Problemy

W obecnej implementacji, na systemie ciążą pewne ograniczenia. Rozdzielczość i częstotliwość skanera są stosunkowo niskie. O ile system działa poprawnie, skaner wyższej rozdzielczości pozwoliłby na wykrywanie dalszych i mniejszych przeszkód - obecnie mogą one znaleźć się pomiędzy wiązkami skanera i zostać wykryte dopiero po zbliżeniu się robota. Wykorzystanie szybszego skanera spowodowałoby osiągnięcie kolejnego ograniczenia, mocy obliczeniowej Raspberry Pi 4. Wówczas konieczne byłoby zastosowanie szybszego komputera lub wykonywanie części obliczeń (nie wymagającej niskich opóźnień, np planowanie lub SLAM globalny) na zewnętrznej jednostce - takie rozwiązanie jest wspierane przez ROS2.

Samo planowanie trasy nie jest pozbawione wad - ponieważ robot ignoruje niskie przeszkody, system nie będzie preferować trasy bez przeszkód nad trasę z małymi przeszkodami. W rzeczywistości, choć obie trasy są przejezdne, jedna z nich będzie szybsza, obciąży robota mniej i nie niesie ryzyka błędów. Aby jednak system mógł rozróżniać takie sytuacje konieczne byłoby skanowanie podłoża, co z kolei wymagałoby ponownie zastosowania skanera o bardzo wysokiej rozdzielczości.

Ze względu na sposób powstania projektu i ograniczony czas realizacji, system mogą występować błędy, np. wynikające z błędnych odczytów sensora, powodujące zaburzenie działania systemu. Zlokalizowanie i naprawa takich błędów wymagałyby dalszych testów w różnych warunkach, prac nad projektem i ostatecznie instalacji np. dodatkowych czujników. Jednym z takich błędów jest utrata pozycji przez silnik krokowy skanera, powodująca błędy w skanach. Jest to sytuacja bardzo rzadka, ale obecnie nie ma sposobu na jej wykrycie. Wymagałoby to zastosowania enkodera i monitorowania pozycji głowicy.

Brak wykorzystania enkoderów w kołach powoduje zmniejszenie precyzji odometrii.

Zdaniem autora jednak w jeździe terenowej dodanie enkoderów nie doprowadziłoby do znaczącej poprawy ze względu na występowanie poślizgu, szczególnie, że testy wykazały zadowalające działanie odometrii.

8.2. Użyteczność

W obecnym stanie robot może być praktycznie zastosowany do nawigacji autonomicznej na krótkich dystansach. Testy długodystansowe nie zostały przeprowadzone. Co oczywiste, system nie dysponuje stabilnością wystarczająco do zatwierdzenia do komercyjnej eksploatacji, ze względu na wielość zmiennych w skomplikowanym zadaniu jakim jest nawigacja autonomiczna. Jednakże, ze względu na docelowe stosowanie w obszarach niezamieszkanych, oczekiwania wobec stabilności i bezpieczeństwa systemu są zdecydowanie mniejsze niż np. w przypadku samochodów autonomicznych. Dzięki temu, zdaniem autora możliwe jest doprowadzenie systemu do stanu użyteczności komercyjnej poprzez dalsze testy i drobne modyfikacje. Wymagane byłoby również utworzenie prostego interfejsu użytkownika, nie wymagające od użytkownika stosowania systemu ROS. Jest to jednak zagadnienie niepowiązane z zakresem pracy.

Z powyższym zastrzeżeniem, robot mógłby być w praktyce zastosowany, np. do eksploracji terenów niebezpiecznych - po wyposażeniu w kamery wykrywające ofiary kataklizmów lub czujnik radiacji do mapowania terenów skażonych lub zadań inspekcyjnych, np. monitorowania szczelności infrastruktury gazowej. Wykorzystanie skanera LiDAR w robocie, w przeciwieństwie do powszechnych rozwiązań bazujących na kamerach, otwiera specyficzną możliwość - robot nie wymaga oświetlenia do funkcjonowania żadnego z systemów. Wobec tego mógłby być wykorzystany do celów zwiadowczych w działaniach wojennych prowadzonych w nocy, bez konieczności stosowania noktowizji. Kolejną zaletą wykorzystania LiDARa jest ogólna niezależność od warunków oświetleniowych, co czyni system stabilniejszym w praktyce - raz wypracowany zestaw ustawień powinien działać w każdych warunkach, w przeciwieństwie do systemów wykorzystujących kamery. Zastosowanie robota w przestrzeni kosmicznej stanowi zupełnie inne wyzwanie, ze specyficznymi wymaganiami termicznymi, masowymi i wytrzymałościowymi.

8.3. Możliwości rozwoju

Poza podstawowym, krótkoterminowym planem rozwoju dążącym do zwiększenia stabilności systemu, system mógłby zostać rozszerzony poprzez:

- Wykorzystanie szybszego skanera, np. rozwiązania komercyjnego wraz z komputerem o adekwatnej mocy obliczeniowej.
- Rozszerzenie platformy mobilnej o uniwersalny interfejs umożliwiający instalację modułów zawierających dodatkowe czujniki lub efektory, zależne od zastosowania. Interfejs obejmowałby zarówno system montażu jak i komunikację oraz zasilania.
- Eksperymenty nad wykorzystaniem algorytmów uczenia maszynowego do części systemu nawigacji autonomicznej, zamiast standardowych rozwiązań. Potencjalnie

pozwoliłoby to na poprawę dokładności i dostosowywanie działania systemu do warunków nie branych pod uwagę w obecnej realizacji. Zależnie od stopnia integracji wymagane byłoby jednak pozyskanie dużej ilości danych, od której bezpośrednio zależałaby jakość systemu. Potencjalnie taki system mógłby działać wraz z rozwiązaniem konwencjonalnym, gdzie wyniki systemów byłyby porównywane by uniknąć nieoczekiwanych błędów wynikających z charakterystyki algorytmów uczenia maszynowego.

Bibliografia

[1] "Encyklopedia PWN", Dostęp zdalny (01.01.2024): https://encyklopedia.pwn.pl/haslo/robot;3968150.html.

[2] "NASA Science Mars 2020 mission Perseverance rover", Dostęp zdalny (01.01.2024): https://mars.nasa.gov/mars2020/spacecraft/rover/communications/, 2020.

[3] *ROS 2 Documentation: Humble*, Dostęp zdalny (01.01.2024): https://docs.ros.org/en/humble/index.html, 2023.

[4] T. Winiarski, "MeROS: SysML-Based Metamodel for ROS-Based Systems", *IEEE Access*, t. 11, s. 82 802–82 815, 2023.

[5] J. Yang, M. Dong i J. Ye, "A Literature Review of the Rocker-Bogie Suspension for the Planetary Rover", w *International Seminar on Artificial Intelligence, Networking and Information Technology*, ResearchGate, 2017.

[6] *Igarashi motoren GMBH 2738-125-GC-5*, Dostęp zdalny (01.01.2024): https://asset.conrad.com/media10/add/160267/c1/-/en/000244475DS01/karta-katalogowa-244475-silnik-elektryczny-szczotkowy-uniwersalny-igarashi-2738-125-gc-5-25-w-6-24-vdc.pdf.

[7] *Serwo PowerHD LF-20MG standard*, Dostęp zdalny (01.01.2024): https://botland.com.pl/serwa-typu-standard/3576-serwo-powerhd-lf-20mg-standard-6939670200387.html.

[8] F. Villa, F. Severini, F. Madonini i F. Zappa, "SPADs and SiPMs Arrays for Long-Range High-Speed Light Detection and Ranging (LiDAR)", *Sensors*, t. 21, nr. 11, s. 3839, 2021.

[9] *RPLIDAR A3M1 - Laserowy skaner LIDAR 360 stopni*, Dostęp zdalny (01.01.2024): https://kamami.pl/czujniki-odleglosci/573426-rplidar-a3m1-laserowy-skaner-lidar-360-stopni.html.

[10] A. Kadambi, A. Bhandari i R. Raskar, "3D Depth Cameras in Vision: Benefits and Limitations of the Hardware", w Springer Cham, 2014, s. 3–26.

[11] *ZED 2 Camera and SDK Overview*, Dostęp zdalny (01.01.2024): https://cdn2.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf, 2019.

[12] *Artec Ray II*, Dostęp zdalny (01.01.2024): https://www.artec3d.com/portable-3d-scanners/laser-ray.

[13] *Grove - IMU 9DOF v2.0 - MPU-9250*, Dostęp zdalny (01.01.2024): https://www.seeedstudio.com/Grove-IMU-9DOF-v2-0.html, 2020.

[14] *GPS Ublox NEO-M8N*, Dostęp zdalny (01.01.2024): https://rc-planeta.pl/pl/gps-i-kompasy/1020-gps-ublox-neo-m8n-z-kompasem-do-pixhawk-apm-5297773808819.html.

[15] A. Becker, "Kalman Filter", Dostęp zdalny (01.01.2024): https://www.kalmanfilter.net/multiSummary.html, 2023.

[16] M. Bajracharya, M. W. Maimone i D. Helmick, "Autonomy for Mars Rovers: Past, Present, and Future", *Computer*, t. 41, nr. 12, s. 44–50, 2008.

[17] *Leo Rover*, Dostęp zdalny (01.01.2024): https://www.leorover.tech/the-rover, 2023.

[18] J. Tang, Y. Chen, A. Kukkoa i H. K. et. al., "SLAM-Aided Stem Mapping for Forest Inventory with Small-Footprint Mobile LiDAR", *Forests*, t. 6, nr. 12, s. 4588–4606, 2015.

[19] K. Bohlmann, A. Beck-Greinwald, S. Buck i H. Marks, "Autonomous Person Following with 3D LIDAR in Outdoor Environments", *Journal of Automation, Mobile Robotics & Intelligent Systems*, t. 7, nr. 2, s. 24–29, 2013.

[20] E. Maulana, M. A. Muslim i A. Zainuri, "Inverse kinematics of a two-wheeled differential drive an autonomous mobile robot", w *Electrical Power, Electronics, Communications, Controls and Informatics Seminar*, ResearchGate, 2014.

[21] *DC Motors : Performance*, Dostęp zdalny (01.01.2024): https://www.johnsonelectric.com/en/resources-for-engineers/dc-motors/dc-motors-performance, 2023.

[22] *sensor_msgs/msg/PointCloud2 Message*, Dostęp zdalny (01.01.2024): https://docs.ros2.org/foxy/api/sensor_msgs/msg/PointCloud2.html, 2020.

[23] *sensor_msgs/msg/Imu Message*, Dostęp zdalny (01.01.2024): https://docs.ros2.org/foxy/api/sensor_msgs/msg/Imu.html, 2020.

[24] *sensor_msgs/msg/NavSatFix Message*, Dostęp zdalny (01.01.2024): https://docs.ros2.org/foxy/api/sensor_msgs/msg/NavSatFix.html, 2020.

[25] M. Zawadzki, "Repozytorium - System autonomicznej nawigacji dla terenowego robota mobilnego", Dostęp zdalny (01.01.2024): https://github.com/MarcelZawadzki/autonomous-rover-thesis/, 2024.

[26] *Silnik krokowy JK28HS32-0674*, Dostęp zdalny (01.01.2024): https://botland.com.pl/silniki-krokowe/3604-silnik-krokowy-jk28hs32-0674-200-krokowobr-38v067a006nm-5904422331993.html.

[27] *Serwo TowerPro SG-90 - micro - 180 stopni*, Dostęp zdalny (01.01.2024): https://botland.com.pl/serwomechanizmy/484-serwo-towerpro-sg-90-micro-180-stopni-5904422329488.html.

[28] *LIDAR-LITE V3HP OPERATION MANUAL AND TECHNICAL SPECIFICATIONS*, Dostęp zdalny (01.01.2024): https://static.garmin.com/pumac/LIDAR-Lite_v3HP_Instructions_EN.pdf, 2020.

[29] *MP6500 Stepper Motor Driver Carrier, Digital Current Control*, Dostęp zdalny (01.01.2024): https://www.pololu.com/product/2968, 2023.

[30] J. Fong, "Stepper Torque vs Voltage", Dostęp zdalny (01.01.2024): https://embeddedtronicsblog.wordpress.com/2020/09/23/stepper-torque-vs-voltage/, 2020.

[31] *LIDARLite_Arduino_Library*, Dostęp zdalny (01.01.2024): https://github.com/garmin/LIDARLite_Arduino_Library, 2020.

[32] *NEO-M8 u-blox M8 concurrent GNSS modules Data sheet*, Dostęp zdalny (01.01.2024): https://content.u-blox.com/sites/default/files/NEO-M8-FW3_DataSheet_UBX-15031086.pdf, 2022.

[33] *MPU9250*, Dostęp zdalny (01.01.2024): https://github.com/hideakitai/MPU9250, 2022.

[34] *NeoGPS*, Dostęp zdalny (01.01.2024): https://github.com/SlashDevin/NeoGPS, 2018.

[35] "NMEA 0183", Dostęp zdalny (01.01.2024): https://pl.wikipedia.org/wiki/NMEA_0183, 2022.

[36] *pySerial*, Dostęp zdalny (01.01.2024): https://pyserial.readthedocs.io/en/latest/pyserial.html, 2020.

[37] *imu_complementary_filter*, Dostęp zdalny (01.01.2024): https://wiki.ros.org/imu_complementary_filter, 2022.

[38] *robot_localiztion*, Dostęp zdalny (01.01.2024): https://github.com/cra-ros-pkg/robot_localization, 2023.

[39] *Cartographer ROS Integration*, Dostęp zdalny (01.01.2024): https://google-cartographer.readthedocs.io/en/latest/, 2018.

[40] W. Hess, D. Kohler, H. Rapp i D. Andor, "Real-Time Loop Closure in 2D LIDAR SLAM", w *IEEE International Conference on Robotics and Automation*, Google scholar, 2016, s. 1271–1278.

[41] *Ceres Solver*, Dostęp zdalny (01.01.2024): http://ceres-solver.org/index.html, 2023.

[42] S. Macenski, F. Martin, R. White i J. Ginés Clavero, "The Marathon 2: A Navigation System", w *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[43] *pcregistericp*, Dostęp zdalny (01.01.2024): https://www.mathworks.com/help/vision/ref/pcregistericp.html, 2024.

[44] M. Zawadzki, "Film z testu terenowego - System autonomicznej nawigacji dla terenowego robota mobilnego", Dostęp zdalny (01.01.2024): https://youtu.be/pcwmK5vn5Fg, 2024.

Wykaz symboli i skrótów

EiTI – Wydział Elektroniki i Technik Informacyjnych
EKF – ang. *Extended Kalman filter*
GNSS – ang. *Global Navigation Satellite System*
GNSS – ang. *Global Positioning System*
I2C – ang. *Inter-Integrated Circuit*
IMU – ang. *Inertial measurement unit*
LiDAR – ang. *Light Detection and Ranging*
NASA – ang. *National Aeronautics and Space Administration*
NEMA – ang. *National Electrical Manufacturers Association*
PW – Politechnika Warszawska
ROS – ang. *Robot Operating System*
SLAM – ang. *Simultaneous Localization and Mapping*
SysML – ang. *Systems Modeling Language*
UART – ang. *Universal Asynchronous Receiver-Transmitter*

Spis rysunków

Spis tabel