



**Politechnika
Śląska**

PROJEKT INŻYNIERSKI

Budowa mapy otoczenia z wykorzystaniem robota mobilnego

Krzysztof GRADEK

Nr albumu: 300362

Kierunek: Automatyka i Robotyka

Specjalność: Technologie Informatyczne

PROWADZĄCY PRACĘ

dr inż. Krzysztof Jaskot

**KATEDRA Katedra Automatyki i Robotyki
Wydział Automatyki, Elektroniki i Informatyki**

Gliwice 2024

Tytuł pracy

Budowa mapy otoczenia z wykorzystaniem robota mobilnego

Streszczenie

Projekt koncentruje się na implementacji systemu autonomicznej nawigacji robota mobilnego, z naciskiem na dwa kluczowe aspekty: tworzenie mapy otoczenia oraz realizację precyzyjnej nawigacji typu z punktu do punktu w zmapowanej przestrzeni wykorzystując rozwiązanie typu SLAM. Rozwiązanie opiera się na dwóch współpracujących ze sobą mikrokontrolerach - Raspberry Pi 4, który odpowiada za obsługę czujnika RPLidar A1 oraz wykonywanie algorytmów mapowania i nawigacji, oraz Arduino Nano zarządzającym silnikami z enkoderami, zapewniającymi precyzyjne sterowanie ruchem robota. Wykonana konfiguracja zrealizowana została w językach C++ oraz Python, z wykorzystaniem narzędzi z ekosystemu ROS 2 (Robot Operating System 2), takich jak Nav2 (Navigation 2), SLAM Toolbox (Simultaneous Localization and Mapping Toolbox) oraz ROS2 Control (Robot Operating System 2 Control).

Słowa kluczowe

Mapowanie, robot mobilny, lokalizacja, SLAM, ROS

Thesis title

Construction of an Environment Map Using a Mobile Robot

Abstract

The project focuses on implementing an autonomous mobile robot navigation system, emphasizing two key aspects: environment mapping and precise point-to-point navigation in the mapped space using SLAM type solution. The solution is based on two cooperating microcontrollers - Raspberry Pi 4, which handles the RPLidar A1 sensor and executes mapping and navigation algorithms, and Arduino Nano managing motors with encoders, providing precise robot motion control. The implemented configuration was realized using C++ and Python languages, utilizing tools from the ROS 2 (Robot Operating System 2) ecosystem, such as Nav2 (Navigation 2), SLAM Toolbox (Simultaneous Localization and Mapping Toolbox) and ROS2 Control (Robot Operating System 2 Control).

Key words

Mapping, mobile robot, localization, SLAM, ROS

Spis treści

1	Wstęp	1
1.1	Wprowadzenie w problem	1
1.2	Osadzenie problemu w dziedzinie	2
1.3	Cel pracy	2
1.4	Zakres pracy	2
1.5	Struktura pracy	3
1.6	Wkład własny autora	3
2	Analiza tematu	5
2.1	Osadzenie tematu w kontekście aktualnego stanu wiedzy	5
2.2	Szczegółowe sformułowanie problemu	5
2.3	Studia literaturowe	6
2.4	SLAM Toolbox	6
2.4.1	Navigation2 (Nav2)	7
2.4.2	ROS2 Control i sterowanie napędami	8
2.4.3	Lokalizacja Monte Carlo	8
2.5	Wybór rozwiązań	8
3	Wymagania i narzędzia	9
3.1	Wymagania funkcjonalne	9
3.2	Przypadki użycia	9
3.3	Specyfikacja komponentów	9
3.3.1	Jednostki sterujące	9
3.3.2	Napęd	10
3.3.3	Zasilanie	10
3.4	Metodyka i etapy realizacji	11
3.4.1	Etap 1: Przygotowanie platformy sprzętowej	11
3.4.2	Etap 2: Implementacja sterowania napędem	11
3.4.3	Etap 3: Integracja sensorów	11
3.4.4	Etap 4: Implementacja oprogramowania	11

4	[Właściwy dla kierunku – np. Specyfikacja zewnętrzna]	13
5	[Właściwy dla kierunku – np. Specyfikacja wewnętrzna]	15
6	Weryfikacja i walidacja	17
7	Podsumowanie i wnioski	19
	Bibliografia	21
	Spis skrótów i symboli	25
	Źródła	27
	Lista dodatkowych plików, uzupełniających tekst pracy	29
	Spis rysunków	31
	Spis tabel	33

Rozdział 1

Wstęp

Poniższy projekt obejmuje implementację systemu autonomicznej nawigacji robota mobilnego, z naciskiem na dwa kluczowe aspekty: tworzenie mapy otoczenia oraz realizację precyzyjnej nawigacji typu z punktu do punktu w zmapowanej przestrzeni. Tego typu zadania są kluczowe w dziedzinie robotyki mobilnej, umożliwiając robotom samodzielne poruszanie się w nowych nieznanych przestrzeniach. W tym rozdziale przedstawiono cel pracy, jej zakres oraz strukturę.

1.1 Wprowadzenie w problem

Jednoczesna lokalizacja i mapowanie - SLAM to proces, w którym robot konstruuje mapę nieznanego środowiska podczas jednoczesnej lokalizacji w tym środowisku i śledzenia swojej trajektorii poruszania się [2]. Jest to jedno z kluczowych zagadnień w robotyce mobilnej, umożliwiając robotom samodzielne poruszanie się w przestrzeni. W praktyce SLAM jest realizowany za pomocą zestawu sensorów, takich jak skanery laserowe, kamery RGB-D, czy IMU, oraz algorytmów, które przetwarzają dane z tych sensorów w celu budowy mapy i lokalizacji robota.

1.2 Osadzenie problemu w dziedzinie

Ten projekt zalicza się do dziedziny robotyki mobilnej i systemów autonomicznych. Roboty mobilne są szeroko wykorzystywane w przemyśle, logistyce, czy badaniach naukowych. Realizacja systemu SLAM i autonomicznej nawigacji wymaga integracji wielu zaawansowanych technologii. Główne wyzwania techniczne obejmują wykorzystanie i synchronizację:

- Sensorów w tym LiDAR
- Algorytmów SLAM
- Szkieletów programistycznych dla robotów jak ROS
- Systemów nawigacji jak Nav2
- Systemów sterowania jak ROS2 Control
- Systemów wizualizacji i analizy danych
- Systemów komunikacji i zarządzania danymi

1.3 Cel pracy

Głównym celem pracy jest zaprojektowanie i implementacja systemu mapowania otoczenia z wykorzystaniem robota mobilnego. W ramach realizacji tego zadania przewidziano budowę platformy mobilnej, implementację systemu sterowania robotem oraz integrację niezbędnych czujników i urządzeń. Kluczowym elementem jest realizacja algorytmów SLAM (Simultaneous Localization and Mapping), które umożliwiają jednoczesną lokalizację robota i tworzenie mapy otoczenia. Dodatkowo, system ma zapewniać możliwość nawigacji z punktu do punktu oraz sterowania manualnego.

1.4 Zakres pracy

Realizacja projektu obejmuje szereg wzajemnie powiązanych zadań. Pierwszym etapem jest dogłębna analiza istniejących rozwiązań w dziedzinie mapowania i nawigacji robotów mobilnych, która stanowi podstawę do dalszych prac. Na tej bazie opracowany zostanie projekt systemu sterowania robotem, a następnie przeprowadzona zostanie jego implementacja. Kolejnym krokiem jest integracja komponentów sprzętowych i programowych w spójny system. Szczególną uwagę poświęcono implementacji algorytmów SLAM i nawigacji, które stanowią rdzeń funkcjonalności robota. Całość prac kończy seria testów i walidacja stworzonego rozwiązania.

1.5 Struktura pracy

Praca składa się z sześciu następujących rozdziałów:

- Rozdział pierwszy zawiera wstęp, w którym przedstawiono cel pracy, jej zakres oraz strukturę.
- Rozdział drugi zawiera analizę tematu, w której przedstawiono problem SLAM, stan aktualny dziedziny oraz przegląd istniejących rozwiązań.
- Rozdział trzeci zawiera wymagania i narzędzia, w którym przedstawiono wymagania funkcjonalne i нефункционалне, przypadki użycia, narzędzia oraz metodykę pracy nad projektem.
- Rozdział czwarty zawiera podstawy teoretyczne, w którym przedstawiono podstawowe pojęcia i definicje z dziedziny robotyki mobilnej.
- Rozdział piąty zawiera wymagania i specyfikację użytkową, w którym przedstawiono wymagania użytkownika oraz specyfikację funkcjonalną i нефункционалną systemu.
- Rozdział szósty zawiera weryfikację i walidację, w którym przedstawiono testy i wyniki działania systemu.
- Rozdział siódmy zawiera podsumowanie, w którym przedstawiono wnioski z pracy oraz możliwości dalszego rozwoju projektu.

1.6 Wkład własny autora

W ramach pracy autor samodzielnie:

- Zaprojektował i zbudował platformę mobilną
- Zaimplementował sterowniki urządzeń
- Zintegrował komponenty sprzętowe i programowe
- Zaimplementował i dostosował algorytm SLAM
- Przeprowadził testy i optymalizację systemu

Rozdział 2

Analiza tematu

W niniejszym rozdziale przedstawiono analizę problemu jednoczesnej lokalizacji i mapowania (SLAM) oraz autonomicznej nawigacji robotów mobilnych. Omówiono aktualny stan wiedzy w tej dziedzinie, sformułowano szczegółowo problem badawczy oraz dokonano przeglądu dostępnych rozwiązań i algorytmów. Na podstawie tej analizy wybrano optymalne narzędzia i metody do realizacji projektu.

2.1 Osadzenie tematu w kontekście aktualnego stanu wiedzy

Problem jednoczesnej lokalizacji i mapowania (SLAM) oraz autonomicznej nawigacji robotów mobilnych stanowi jeden z kluczowych obszarów badań w dziedzinie robotyki. W ostatnich latach obserwuje się znaczący postęp w tej dziedzinie, głównie dzięki rozwojowi wydajnych algorytmów optymalizacji, poprawie jakości i dostępności czujników jak LiDAR, wzrostowi mocy obliczeniowej komputerów oraz powstaniu zaawansowanych szkieletów programistycznych jak ROS 2.

2.2 Szczegółowe sformułowanie problemu

Problem postawiony w niniejszej pracy obejmuje dwa główne aspekty. Pierwszy z nich to mapowanie otoczenia, które wymaga efektywnej akwizycji danych z czujników, przetwarzania chmur punktów, estymacji pozycji robota oraz łączenia kolejnych skanów w spójną mapę. Drugi aspekt to autonomiczna nawigacja, gdzie system musi zapewniać precyzyjną lokalizację w znanej mapie, planowanie ścieżki z uwzględnieniem przeszkód oraz dokładną kontrolę ruchu robota.

2.3 Studia literaturowe

2.4 SLAM Toolbox

SLAM Toolbox jest nowoczesnym rozwiązaniem dla ROS 2, umożliwiającym robotom tworzenie i aktualizację map 2D w czasie rzeczywistym [4]. System implementuje zaawansowaną wersję algorytmu Pose Graph SLAM, oferując kilka trybów pracy:

- **Synchroniczny SLAM** - sekwencyjne przetwarzanie skanów LiDAR, gwarantujące spójność mapy
- **Asynchroniczny SLAM** - równoległe przetwarzanie skanów dla zwiększonej wydajności
- **Lifelong SLAM** - ciągle mapowanie z wydajnym zarządzaniem pamięcią

Proces mapowania łączy dane z czujnika LiDAR oraz informacje odometryczne. Wykorzystując algorytm Correlative Scan Matching, system określa względną pozycję robota i buduje graf pozycji, gdzie węzły reprezentują położenia robota, a krawędzie określają względne transformacje między nimi.

Kluczowym elementem jest optymalizacja z wykorzystaniem biblioteki Ceres Solver [1]. Jest to narzędzie, które pomaga znaleźć najlepsze dopasowanie między różnymi pomiarami, działając podobnie do "układanki", gdzie próbujemy ułożyć elementy tak, aby wszystko do siebie pasowało.

Optymalizator rozwiązuje problem minimalizacji metodą najmniejszych kwadratów:

$$\min_x \frac{1}{2} \sum_i \rho_i(\|f_i(x_i)\|^2) \quad (2.1)$$

Ta matematyczna formuła, choć wygląda skomplikowanie, opisuje prosty koncept:

- x to pozycje robota, które chcemy znaleźć
- $f_i(x_i)$ mierzy, jak bardzo nasze oszacowanie pozycji różni się od rzeczywistych pomiarów
- ρ_i to funkcja, która pomaga ignorować błędne pomiary (działa jako filtr odrzucający "podejrzane" dane)
- Cały proces dąży do znalezienia takich pozycji robota, dla których suma wszystkich błędów jest najmniejsza

Algorytm działa iteracyjnie, czyli krok po kroku poprawia swoje oszacowania, podobnie jak podczas strojenia radia - powoli przekreścamy pokrętkę, aż znajdziemy najczystszy sygnał. Wykorzystuje do tego metodę Levenberga-Marquardta, która łączy w sobie:

- Szybkość metody Gaussa-Newtona (dobre do precyzyjnych korekt)
- Stabilność metody największego spadku (pomocne przy większych korektach)

Dzięki takiemu podejściu SLAM Toolbox może tworzyć dokładne mapy, nawet gdy niektóre pomiary są niedokładne lub błędne.

2.4.1 Navigation2 (Nav2)

System Navigation 2 (Nav2) stanowi kompleksową strukturę do nawigacji robotów w ROS 2. Poniżej opisano główne komponenty i ich funkcjonalności:

- **BehaviorTree Navigator** - centralny moduł decyzyjny implementujący:
 - Hierarchiczny system podejmowania decyzji
 - Sekwencyjne przetwarzanie zadań nawigacyjnych
 - Elastyczne zarządzanie zachowaniem robota
- **Global Planner** - planowanie globalnej trasy:
 - Algorytm A* do znajdowania optymalnej ścieżki
 - Algorytm Dijkstry do wyznaczania najkrótszej drogi
 - Metody oparte na polach potencjałowych
- **Local Planner** - lokalny kontroler ruchu:
 - DWB - sterowanie w oknie dynamicznym
 - TEB - optymalizacja trajektorii z uwzględnieniem ograniczeń
 - Algorytmy unikania przeszkód
- **Recovery Behaviors** - zachowania awaryjne:
 - Czyszczenie mapy kosztów
 - Skanowanie otoczenia przez obrót
 - Wycofywanie z obszarów kolizji

Nawigacja opiera się na dwóch mapach kosztów:

- Global Costmap - globalna mapa środowiska
- Local Costmap - lokalna mapa aktualizowana na bieżąco

System współpracuje z algorytmami SLAM i lokalizacji, zapewniając nawigację w czasie rzeczywistym. Wykorzystuje standardowe interfejsy ROS 2 do komunikacji z innymi komponentami.

2.4.2 ROS2 Control i sterowanie napędami

System sterowania napędami bazuje na infrastrukturze ROS2 Control [3], która zapewnia abstrakcję sprzętową i standaryzowane interfejsy kontrolerów. W projekcie wykorzystano implementację DiffDrive Arduino [6], która realizuje sterowanie napędem różnicowym z wykorzystaniem enkoderów do precyzyjnego pozycjonowania. System implementuje odometrię przez zliczanie impulsów enkoderów i estymację pozycji robota na podstawie modelu kinematycznego.

2.4.3 Lokalizacja Monte Carlo

Do lokalizacji w znanej mapie wykorzystano algorytm Adaptacyjnej Lokalizacji Monte Carlo (AMCL) [5]. Metoda ta używa filtrów cząsteczkowych do probabilistycznej estymacji pozycji robota. Algorytm automatycznie dostosowuje liczbę cząstek w zależności od niepewności lokalizacji, co pozwala na zachowanie dobrej dokładności przy optymalnym wykorzystaniu zasobów obliczeniowych.

2.5 Wybór rozwiązań

Na podstawie analizy dostępnych narzędzi, w projekcie zdecydowano się na wykorzystanie SLAM Toolbox do mapowania, AMCL do lokalizacji podczas nawigacji, Nav2 do planowania ścieżki i kontroli ruchu oraz ROS2 Control z DiffDrive Arduino do sterowania napędami. Wybór ten podyktowany jest stabilnością rozwiązań, oraz dobrą integracją komponentów w ekosystemie ROS 2 oraz aktywnym wsparciem społeczności i dostępnością dokumentacji.

Rozdział 3

Wymagania i narzędzia

W niniejszym rozdziale przedstawiono wymagania funkcjonalne systemu, przypadki użycia w formie diagramu UML, szczegółową specyfikację wykorzystanych komponentów sprzętowych oraz metodykę i etapy realizacji projektu.

3.1 Wymagania funkcjonalne

System powinien realizować następujące funkcje:

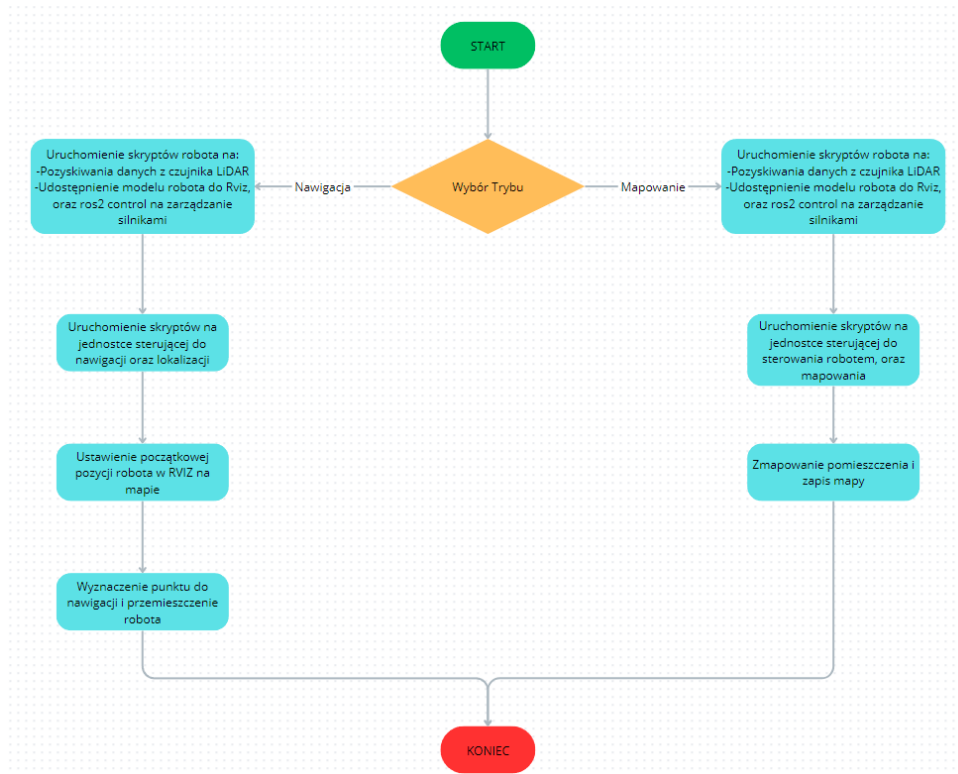
- Zdalne sterowanie robotem poprzez klawiaturę (teleop keyboard) w celu eksploracji i mapowania otoczenia
- Tworzenie i zapisywanie mapy otoczenia w czasie rzeczywistym przy użyciu algorytmu SLAM
- Lokalizacja robota na zapisanej mapie z wykorzystaniem algorytmu AMCL
- Autonomiczna nawigacja do wyznaczonych punktów na mapie z omijaniem przeszkód
- Planowanie i optymalizacja ścieżki przejazdu

3.2 Przypadki użycia

3.3 Specyfikacja komponentów

3.3.1 Jednostki sterujące

- Raspberry Pi 4 - główny komputer pokładowy:
 - System operacyjny Ubuntu 22.04



Rysunek 3.1: Diagram przypadków użycia systemu

- ROS 2 Humble
- Komunikacja przez SSH
- Arduino Nano - sterownik silników:
 - Obsługa enkoderów
 - Komunikacja szeregową z Raspberry Pi

3.3.2 Napęd

- 2x silnik DC 12V 240RPM z metalową przekładnią
- Wbudowane enkodery magnetyczne Halla
- Sterownik L298N - dwukanałowy mostek H

3.3.3 Zasilanie

- 6x akumulatory Li-ion 18650:
 - 4 ogniwa (2S2P) dla silników
 - 2 ogniwa (2S) dla elektroniki

- 2x przetwornica step-down:
 - 12V dla silników
 - 5V dla Raspberry Pi

3.4 Metodyka i etapy realizacji

3.4.1 Etap 1: Przygotowanie platformy sprzętowej

- Instalacja systemu Ubuntu 22.04 na Raspberry Pi
- Konfiguracja połączenia SSH
- Instalacja ROS 2 Humble

3.4.2 Etap 2: Implementacja sterowania napędem

- Podłączenie silników do sterownika L298N
- Programowanie Arduino - obsługa silników i enkoderów
- Implementacja komunikacji szeregowej z Raspberry Pi

3.4.3 Etap 3: Integracja sensorów

- Montaż i konfiguracja LiDAR-a
- Kalibracja czujników
- Opracowanie układu mechanicznego i obudowy

3.4.4 Etap 4: Implementacja oprogramowania

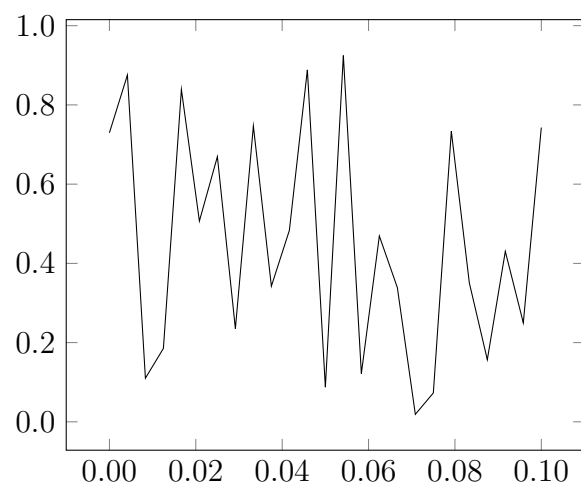
- Konfiguracja pakietów ROS 2:
 - SLAM Toolbox do mapowania
 - Nav2 do nawigacji
 - AMCL do lokalizacji
- Implementacja sterowników urządzeń
- Integracja i testy systemu

Rozdział 4

[Właściwy dla kierunku – np. Specyfikacja zewnętrzna]

Jeśli „Specyfikacja zewnętrzna”:

- wymagania sprzętowe i programowe
- sposób instalacji
- sposób aktywacji
- kategorie użytkowników
- sposób obsługi
- administracja systemem
- kwestie bezpieczeństwa
- przykład działania
- scenariusze korzystania z systemu (ilustrowane zrzutami z ekranu lub generowanymi dokumentami)



Rysunek 4.1: Podpis rysunku po rysunkiem.

Rozdział 5

[Właściwy dla kierunku – np. Specyfikacja wewnętrzna]

Jeśli „Specyfikacja wewnętrzna”:

- przedstawienie idei
- architektura systemu
- opis struktur danych (i organizacji baz danych)
- komponenty, moduły, biblioteki, przegląd ważniejszych klas (jeśli występują)
- przegląd ważniejszych algorytmów (jeśli występują)
- szczegóły implementacji wybranych fragmentów, zastosowane wzorce projektowe
- diagramy UML

Krótką wstawka kodu w linii tekstu jest możliwa, np. **int a;** (biblioteka `listings`). Dłuższe fragmenty lepiej jest umieszczać jako rysunek, np. kod na rys 5.1, a naprawdę długie fragmenty – w załączniku.

```
1 class test : public basic
2 {
3     public:
4         test (int a);
5         friend std::ostream operator<<(std::ostream & s,
6                                         const test & t);
7     protected:
8         int _a;
9
10 };
```

Rysunek 5.1: Pseudokod w `listings`.

Rozdział 6

Weryfikacja i walidacja

- sposób testowania w ramach pracy (np. odniesienie do modelu V)
- organizacja eksperymentów
- przypadki testowe zakres testowania (pełny/niepełny)
- wykryte i usunięte błędy
- opcjonalnie wyniki badań eksperymentalnych

Tabela 6.1: Nagłówek tabeli jest nad tabelą.

ζ	metoda						
	alg. 1	alg. 2	alg. 3			alg. 4, $\gamma = 2$	
			$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Rozdział 7

Podsumowanie i wnioski

- uzyskane wyniki w świetle postawionych celów i zdefiniowanych wyżej wymagań
- kierunki ewentualnych danych prac (rozbudowa funkcjonalna ...)
- problemy napotkane w trakcie pracy

Bibliografia

- [1] Sameer Agarwal, Keir Mierle i The Ceres Solver Team. *Ceres Solver*. Wer. 2.2. Paź. 2023. URL: <https://github.com/ceres-solver/ceres-solver>.
- [2] Luis Bermudez. *Medium - Overview of SLAM*. 2024. URL: <https://medium.com/machinevision/overview-of-slam-50b7f49903b7> (term. wiz. 17.04.2024).
- [3] Bence Magyar Christoph Fröhlich Denis Stogl i Sai Kishor Kothakota. *ros2 control*. 2024. URL: <https://control.ros.org/humble/index.html> (term. wiz. 01.12.2024).
- [4] Steve Macenski. *ROS.org - slam toolbox*. 2021. URL: https://wiki.ros.org/slam_toolbox (term. wiz. 16.12.2021).
- [5] Steve Macenski, Francisco Martin, Ruffin White i Jonatan Gines Clavero. *The Marathon 2: A Navigation System*. 2020. URL: <http://dx.doi.org/10.1109/IRoS45743.2020.9341207> (term. wiz. 03.10.2020).
- [6] Josh Newans. *ROS.org - diffdrive arduino*. 2020. URL: https://wiki.ros.org/diffdrive_arduino (term. wiz. 08.12.2020).

Dodatki

Spis skrótów i symboli

SLAM jednoczesna lokalizacja i mapowanie (ang. *Simultaneous Localization and Mapping*)

LiDAR urządzenie wykonujące wykrywanie światła i określanie odległości (ang. *Light Detection and Ranging*)

IMU inercyjna jednostka pomiarowa (ang. *Inertial Measurement Unit*)

RGB-D kamera rejestrująca obraz RGB oraz informację o głębi (ang. *RGB-Depth*)

Nav2 system nawigacji dla ROS 2 (ang. *Navigation 2*)

ROS system operacyjny dla robotów (ang. *Robot Operating System*)

ROS2 Control system kontroli robotów dla ROS 2 (ang. *Robot Operating System 2 Control*)

SLAM Toolbox zestaw narzędzi do jednoczesnej lokalizacji i mapowania (ang. *Simultaneous Localization and Mapping Toolbox*)

Źródła

Jeżeli w pracy konieczne jest umieszczenie długich fragmentów kodu źródłowego, należy je przenieść w to miejsce.

```
1 if (_nClusters < 1)
2     throw std::string ("unknown_number_of_clusters");
3 if (_nIterations < 1 and _epsilon < 0)
4     throw std::string ("You should set a maximal number of
        iteration or minimal difference — epsilon.");
5 if (_nIterations > 0 and _epsilon > 0)
6     throw std::string ("Both number of iterations and minimal
        epsilon set — you should set either number of iterations
        or minimal epsilon.");
```

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu,
- dane testowe,
- film pokazujący działanie opracowanego oprogramowania lub zaprojektowanego i wykonanego urządzenia,
- itp.

Spis rysunków

3.1	Diagram przypadków użycia systemu	10
4.1	Podpis rysunku po rysunkiem.	14
5.1	Pseudokod w <code>listings</code>	16

Spis tabel

6.1	Nagłówek tabeli jest nad tabelą.	18
-----	--	----