Roboty Mobilne

Kierunek	Termin
Automatyka i Robotyka	*Czw TP 17:05*
Imię, nazwisko, numer albumu,grupa	**Data**
Piotr Koczy 259367 *Paweł Łyszczarz 259258*	*7 października 2023*
Temat sprawozdania	**Prowadzący**
Etap II - progres w pracach projektowych	*dr. inż. Michał Błędowski*

Spis treści

1 Cel projektu

Celem projektu jest stworzenie symulacji komputerowej robota mapującego w ROS 2. Robot po umieszczeniu w wybranym pokoju/mapie, będzie przemieszczał się i zbierał informację z czujnika odległości LIDAR, dzięki któremu będzie w stanie określić kształt pomieszczenia w którym się znajduje i odzwierciedlić go w postaci mapy. Planowane jest aby robot nieustannie poruszał się po pomieszczeniu do momentu kiedy wszystkie możliwe ściany zostaną już wykryte i naszkicowane, wówczas robot powinien wrócić to pozycji w której zaczął proces mapowania. Do wizualizacji i tworzenia realistycznych modeli środowisk jak i również symulowania zachowania robotów będziemy wykorzystywać narzędzia takie jak Gazebo oraz Rviz. Zamierzamy również skorzystać z algorytmu SLAM (Simultaneous Loaclization and Mapping), który umożliwi trafne mapowanie otoczenia podczas ruchu robota oraz środowiska nawigacyjnego Nav2, które pozwala na wyznaczanie ścieżki do zadanego punktu na mapie.

2 Założenia projektowe

1. Bazowanie na platformie TurtleBot.
2. Praca w symulacyjnym środowisku ROS 2 z wykorzystaniem RViz i Gazebo.
3. Do pozyskiwania danych o otoczeniu użyte zostaną czujniki typ lidar, encodery położeń kół oraz IMU.
4. Implementacja algorytmu mapowania podczas lokalizacji SLAM.
5. Skorzystanie z paczki Nav2 wchodzącej w skład ROS2 do nawigowania.
6. Autonomiczne mapowanie przestrzeni wokół robota.

3 Podział pracy na poszczególne etapy następujące po sobie

- Etap I
    1. 03.04 – Przygotowanie (instalacja) środowiska ROS2 i wymaganych pakietów
    2. 10.04 –Wyświetlenie modelu robota w RViz
    3. 22.04 – Uruchomienie symulacji ręcznie sterowanego robota w Gazebo
- Etap II
    1. 29.04 – Zintegrowanie SLAM z Nav2
    2. 11.05 – Mapowanie przestrzeni ręcznie sterowanym robotem
- Etap III
    1. 30.05 – Autonomiczne mapowanie robotem poruszającym się w losowych kierunkach
    2. 22.06 – Implementacja algorytmu logicznego mapowania przez robota (dążenie do niewykrytych jeszcze przestrzeni)

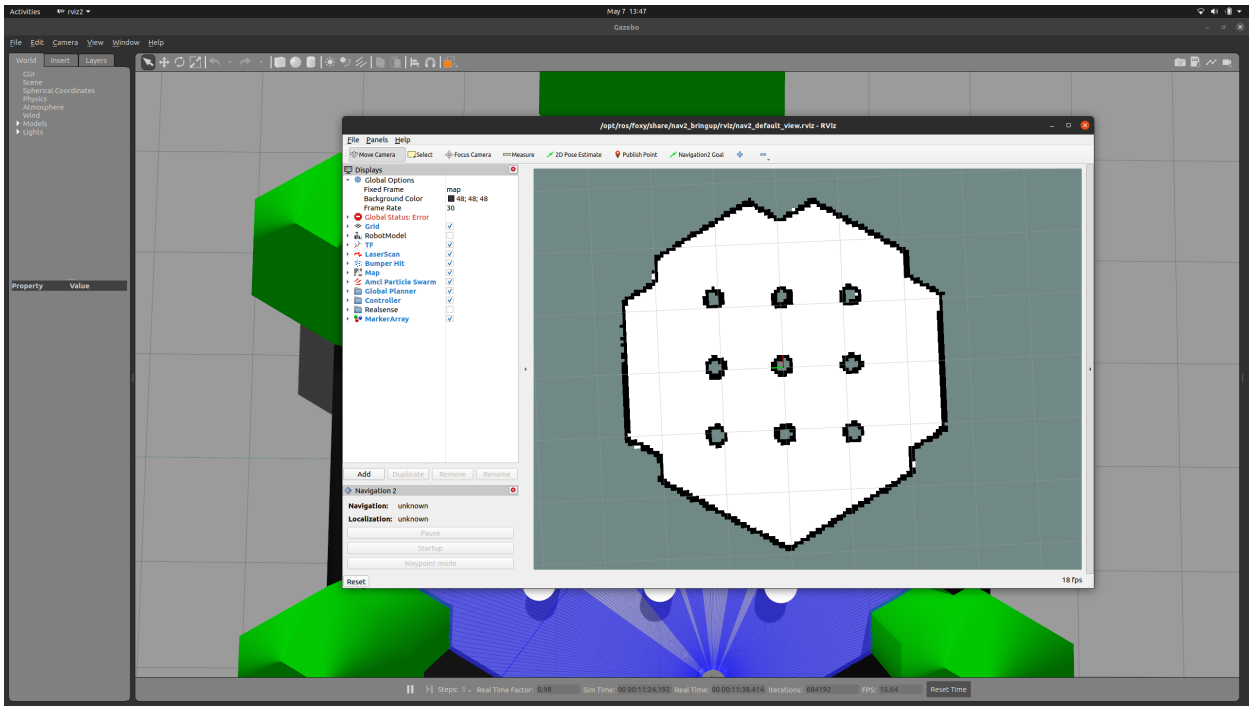
4 Podział pracy na członków zespołu

1. Przygotowanie (instalacja) środowiska ROS2 i wymaganych pakietów
   - Paweł Łyszczarz
   - Piotr Koczy
2. Wyświetlenie modelu robota w RViz
   - Paweł Łyszczarz
3. Uruchomienie symulacji ręcznie sterowanego robota w Gazebo
   - Piotr Koczy
4. Zintegrowanie SLAM z Nav2
   - Paweł Łyszczarz
5. Mapowanie przestrzeni ręcznie sterowanym robotem
   - Piotr Koczy
6. Autonomiczne mapowanie robotem poruszającym się w losowych kierunkach
   - Paweł Łyszczarz, Piotr Koczy
7. Implementacja algorytmu logicznego mapowania przez robota (dążenie do niewykrytych jeszcze przestrzeni)
   - Paweł Łyszczarz, Piotr Koczy

5 Etap II - progres w pracach nad projektem

Do tego momentu prac zaplanowane było zintegrowanie wszystkich wymaganych środowisk, co pozwoliło na wyświetlanie modeli robota oraz mapy po której miałby się poruszać.

5.1 Wyświetlanie mapy w środowisku Rviz

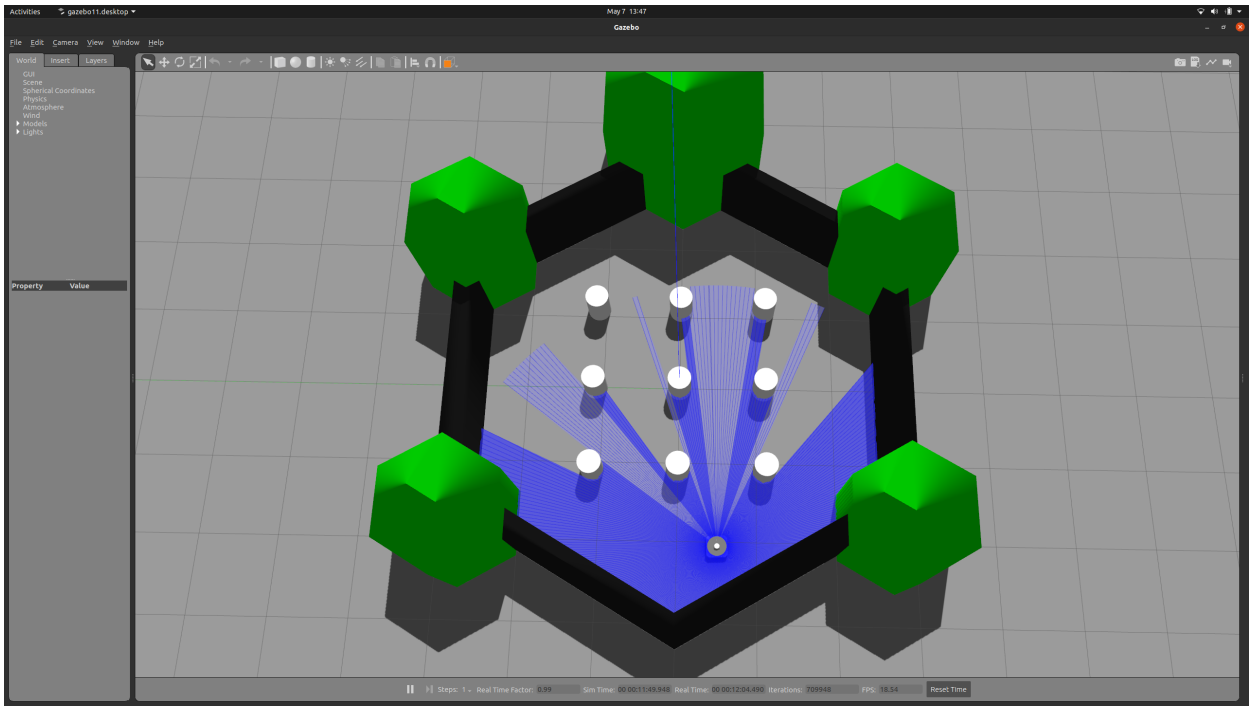


Rysunek 1: Podstawowa mapa do testów

Jak widać na załączonym wyżej obrazku udało się programowo wgrać podstawową mapę i wyświetlić ją w środowisku Rviz. Mapa nie zawiera jeszcze na swoim planie modelu robota.

5.2 Uruchomienie symulacji ręcznie sterowanego robota w środowisku Gazebo

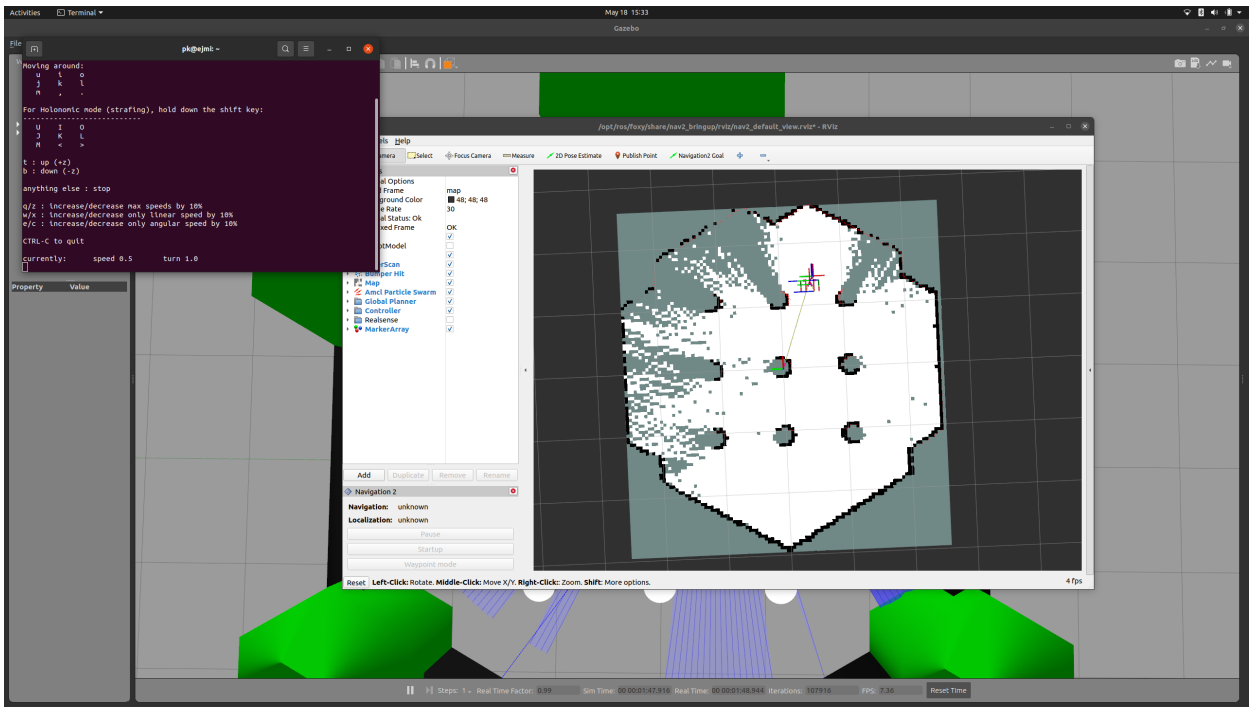
Kolejnym krokiem było uruchomienie symulacji w aplikacji Gazebo wraz z modelem robota (w tym przypadku skorzystaliśmy z jednego z najbardziej podstawowych modeli robota którym jest Turtlebot3).



Rysunek 2: Symulacja mapy i robota w Gazebo

5.3 Uruchomienie SLAM i mapowanie przestrzeni ręcznie sterowanym robotem

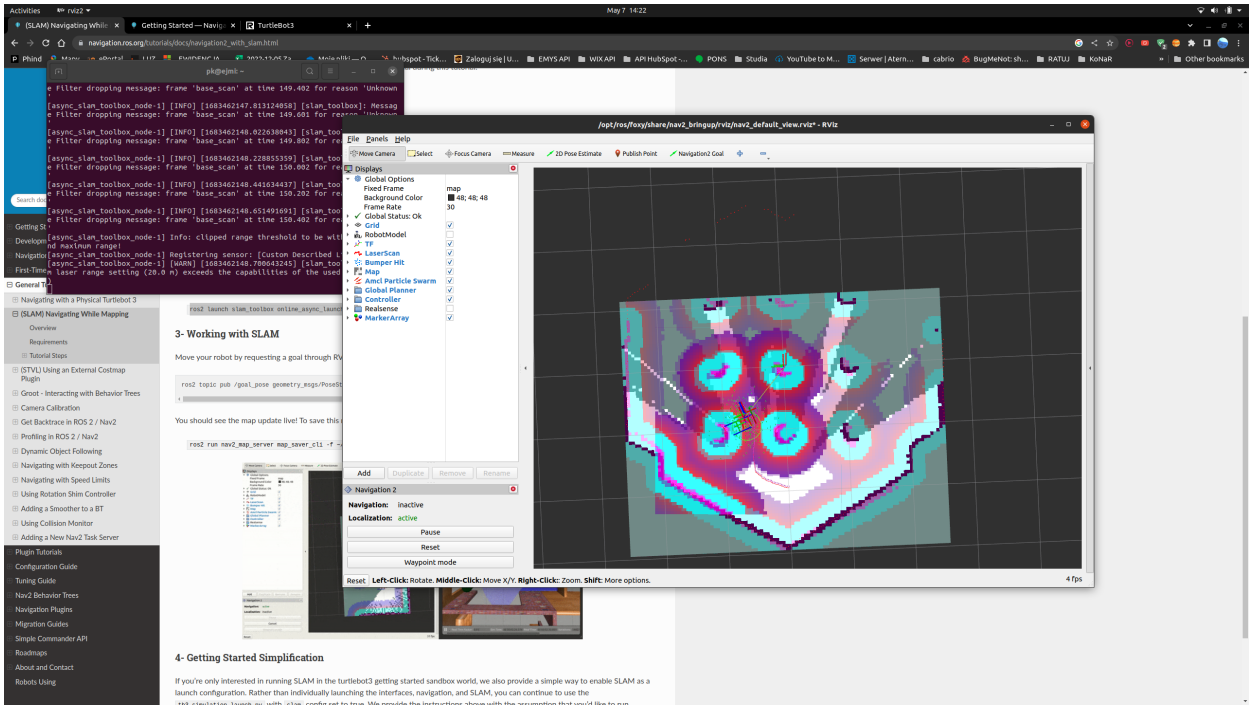
SLAM pozwolił na mapowanie i zapamiętywanie terenu otaczającego model robota, który był przez nas ręcznie przemieszczany po przestrzeni mapy.



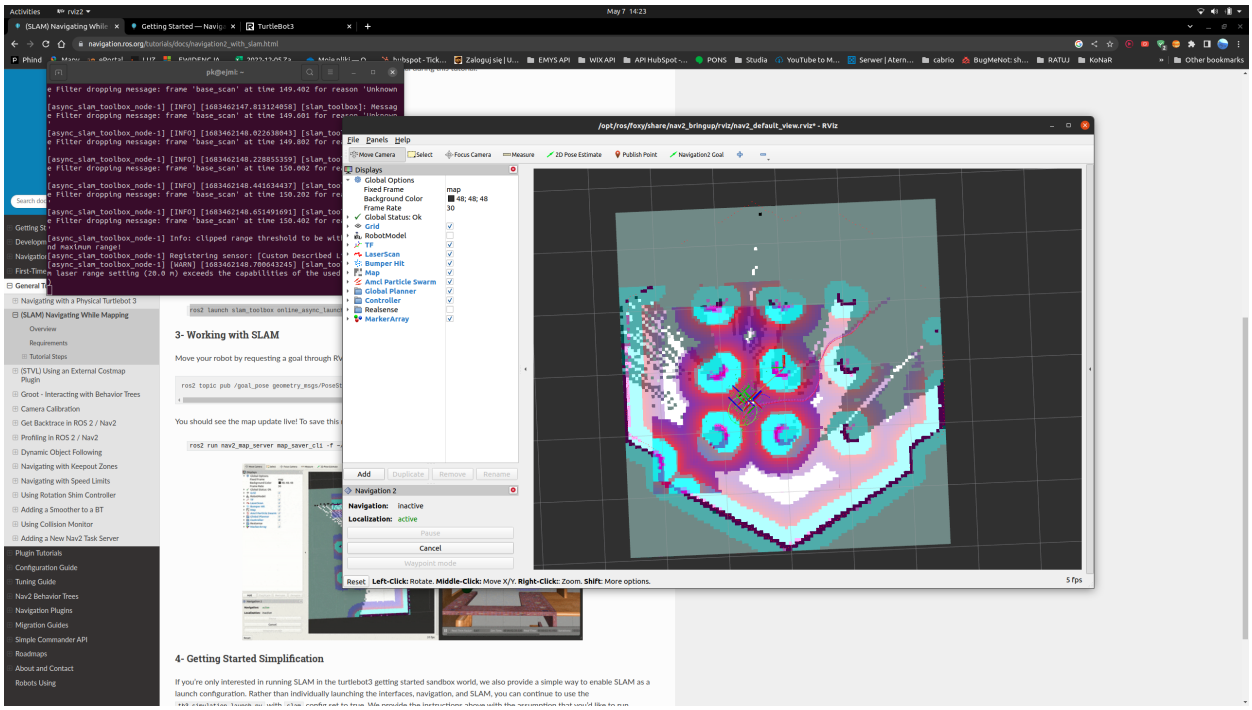
Rysunek 3: Ręczne mapowanie SLAM

5.4 Uruchomienie Nav2

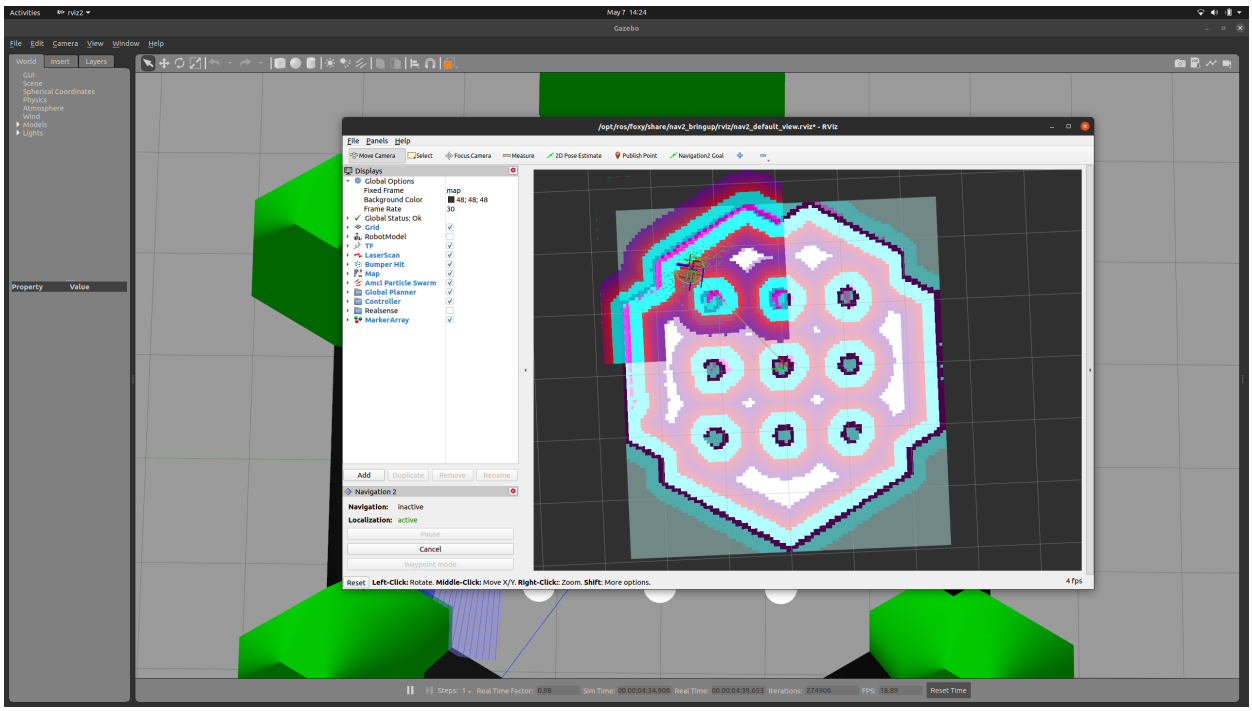
Node Nav2 nakłada na mapę tworzoną przez SLAM warstwę kosztów (przeszkody i ich otoczenie mają wyższy koszt niż pusta przestrzeń).
Rviz pozwala na wysłanie punktu docelowego do Nav2, wtedy zaczyna się wyznaczanie optymalnej ścieżki i robot rusza w zadanym kierunku.



Rysunek 4: Tworzenie mapy cz.1



Rysunek 5: Tworzenie mapy cz.2



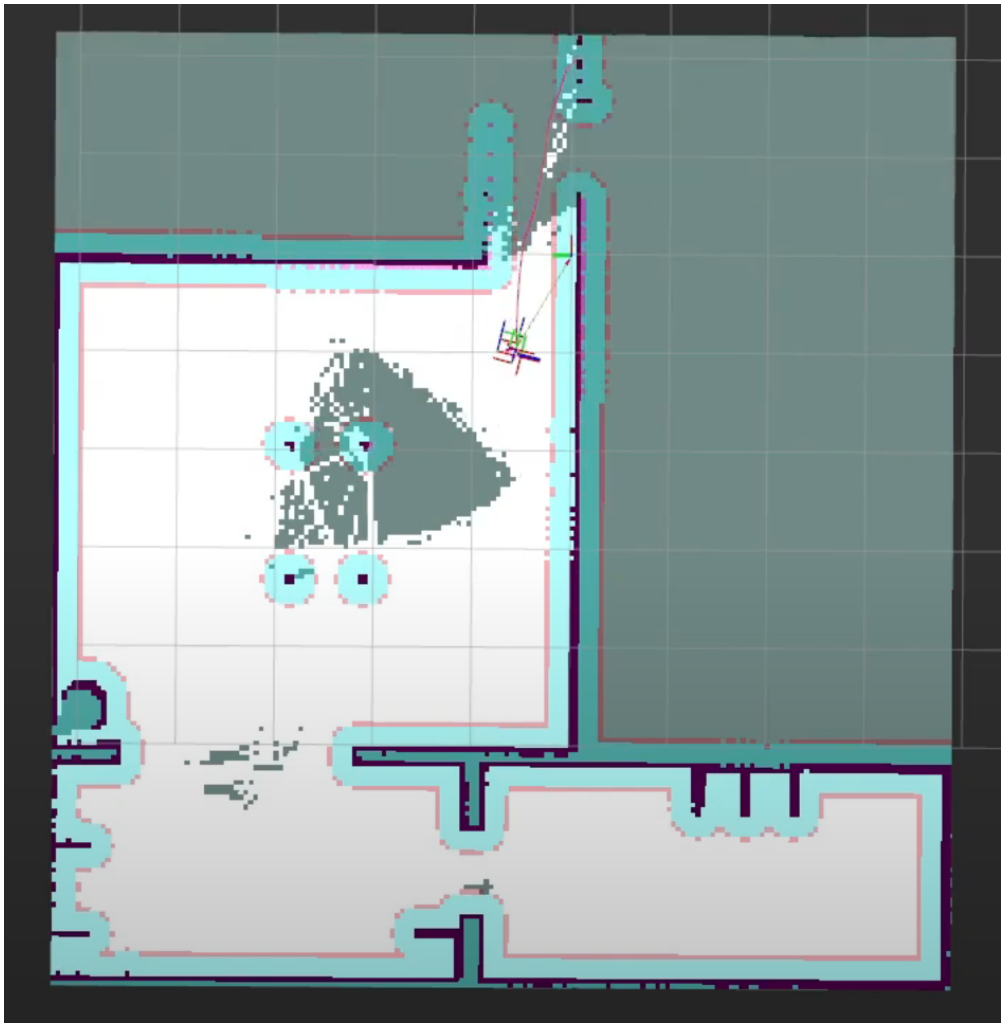
Rysunek 6: Tworzenie mapy cz.3

6 Rezultaty końcowe

Projekt autonomicznie mapującego robota mobilnego został zakończony, osiągając szereg istotnych rezultatów. Ten innowacyjny projekt miał na celu stworzenie inteligentnego robota, który będzie zdolny do samodzielnego mapowania otoczenia i nawigacji w dynamicznym środowisku.

6.1 Mapowanie w środowisku symulacyjnym

Robot jest wyposażony w skaner laserowy (Lidar) dzięki któremu wraz z algorytmami przetwarzania obrazu, robot był w stanie skanować otoczenie i generować dokładne mapy środowiska. W symulacji Gazebo wykorzystano metodę SLAM (Simultaneous Localization and Mapping), która umożliwiła równoczesne lokalizowanie robota oraz tworzenie mapy jego otoczenia. Dzięki temu robot był w stanie samodzielnie eksplorować nieznane obszary i tworzyć precyzyjne mapy z wysokim poziomem szczegółowości. Do wizualizacji zaistniałych operacji i wydarzeń robota, jego otocznia i zmapowanego terenu oprócz wysokiej jakości graficznego wizualizatora Rviz2 wykorzystaliśmy przede wszystkim Nav2, które wyznaczało ścieżkę po której poruszać się miał robot do wybranego punktu 7.



Rysunek 7: Zmapowany teren, robot i ścieżka po której się porusza

6.2 Algorytm odkrywania mapy

Robot planował swoje trasy na podstawie zaimplementowanego algorytmu dopierania optymalnego i jak najbardziej efektywnego punktu, który pozwoli na zmapowanie jak najwięcej nowej przestrzeni. Algorytm opiera się na metodzie kar i nagród i polega ona na ocenie potencjalnych miejsc w które robot mógł się udać zwracając uwagę na ich otoczenie, czy jest to przeszkoda, teren już odkryty czy tez teren jeszcze niezbadany, im więcej pola niezbadanego w około potencjalnego punktu tym punkt zyskuje większą wartość nagrody co w rezultacie może przyczynić się do wyboru tego punktu jako najbardziej optymalnego w danej chwili 8.

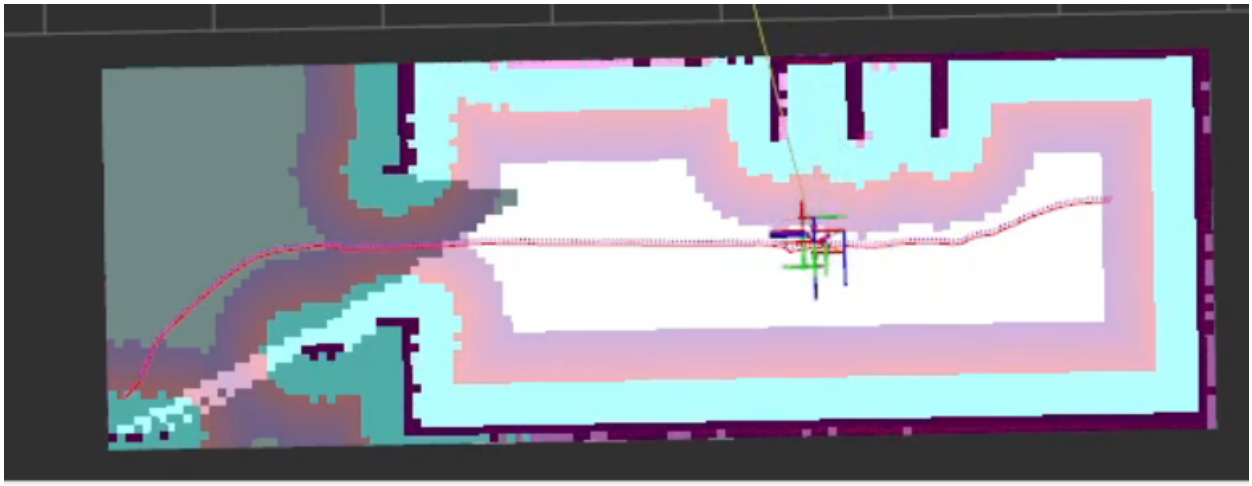
6.2.1 Schemat działania

1. Odczytanie mapy z topicu `/map`
2. Zapisanie do listy wszystkich punktów na granicy terenu znanego i nieznanego
3. Sprawdzenie, czy mapowanie jest zakończone
4. Obliczenie, który z tych punktów jest najlepszy jako cel jazdy
5. Wysłanie celu do Nav2
6. Odpytywanie Nav2, czy robot dotarł do celu

6.2.2 Wybieranie optymalnego punktu

Dla każdego punktu granicznego, czyli w takiego, który ma wartość `-1` (nieznany teren) i w którego otoczeniu znajduje się punkt o wartości `0` (znany teren - ponad 0 to byłaby przeszkoda) obliczany jest zysk informacyjny.

Zysk informacyjny oblicza się poprzez badanie otoczenia punktu w zadanym promieniu - w testach ustaliliśmy, że wartość 5 działa zadowalająco. Zostaje wyznaczona liczba punktów nieznanych i przeszkód - za przeszkodę w otoczeniu naliczana jest kara, a za punkt nieznany nagroda. Punkt o najwyższym wyniku zostaje wybrany na następny cel robota.



Rysunek 8: Robot udający się w teren w którym odkryje najwięcej nieznanej przestrzeni

6.2.3 Zakończenie mapowania

Mapowanie zostaje uznane za zakończone, jeśli liczba punktów granicznych spadnie poniżej zadanej wartości. Podczas testów wartość ta została ustalona na 50.

7 Autonomiczne sterowanie

Osiągnięcie autonomicznego sterowania robota mobilnego, umożliwiającego samodzielne poruszanie się, unikanie przeszkód i utrzymywanie ustalonej trasy, zostało osiągnięte dzięki odpowiedniej konfiguracji narzędzia nawigacji Nav2, jest to zaawansowany pakiet nawigacyjny, który zapewnia szereg funkcji niezbędnych do sterowania autonomicznego robotem. Poprzez konfigurację i integrację różnych komponentów, takich jak lokalizacja, planowanie trasy, sterowanie prędkością i unikanie przeszkód, nav2 umożliwia robotowi autonomiczną nawigację w czasie rzeczywistym.

8 Efekty działania i wnioski

W ramach projektu przeprowadzono testy mapowania i poruszania się robota na większej, bardziej skomplikowanej mapie niż wcześniej. Wybrano obszar, który zawierał większą liczbę przeszkód, zawiłych ścieżek i bardziej wymagających warunków nawigacyjnych. Przeprowadzenie testów na bardziej skomplikowanej mapie było ważnym krokiem, aby zweryfikować zdolności i niezawodność systemu nawigacyjnego 10. Niestety przez notoryczne błędy systemów SLAM i Nav2 nigdy nie udało się w pełni zmapować dużej mapy, kilka podejść zostało nagranych:

- Mapowanie przerwane przez błąd **SLAM**
- Mapowanie przerwane przez błąd **Nav2**

Pomimo tych przeciwności udało się uzyskać niezawodne mapowanie domyślnego świata Turtlebot World - algorytm działał tutaj niezawodnie, podobnie jak Nav2 i SLAM. Można zobaczyć to **tutaj**

Rysunek 9: Zmapowany cały Turtlebot World

```
[turtlebot3_explorer-1] [INFO] [1687437938.168750856] [turtlebot3_explo
rer]: Sending goal position: 0.14520869420436266, 2.324625808606932
[turtlebot3_explorer-1] [INFO] [1687437938.171475382] [turtlebot3_explo
rer]: Navigating to goal: 0.14520869420436266 2.324625808606932...
[turtlebot3_explorer-1] [INFO] [1687437948.139448732] [turtlebot3_explo
rer]: Goal succeeded!
[turtlebot3_explorer-1] [INFO] [1687437948.162732099] [turtlebot3_explo
rer]: ----Mapping finished----
[INFO] [turtlebot3_explorer-1]: process has finished cleanly [pid 14806
0]
```

Rysunek 10: Komunikat o zakończeniu mapowania

Po przeprowadzeniu projektu autonomicznie mapującego robota w ROSie, możemy wyciągnąć kilka istotnych wniosków:

- Po pierwsze, opracowane algorytmy mapowania, planowania trasy i sterowania autonomicznego okazały się (w większości przypadków) skuteczne i efektywne. Robot był w stanie precyzyjnie mapować otoczenie, unikać przeszkód i utrzymywać ustaloną trasę nawigacji. To potwierdza, że odpowiednia konfiguracja narzędzia nawigacyjnego Nav2 w ROS jest kluczowa dla osiągnięcia wysokiego poziomu autonomii robota.
- Po drugie, udało się niezawodnie i w pełni zmapować domyślną mapę Turtlebot World, co było głównym celem tego projektu

- Duże problemy sprawiała konfiguracja środowiska Nav2, która przez długi czas uniemożliwiała poprawne poruszanie się robota po dobrze wyznaczonych trasach (stawanie w miejscu, uderzanie w przeszkody). Wnioskiem do tego problemu jest to że warto korzystać z jak najnowszej wersji tego oprogramowania w celu likwidacji uporczywych błędów. Nie mogliśmy tego zrobić, ponieważ wymagało to zmiany dystrybucji ROS 2, na co po prostu zabrakłoby czasu.

9 Bibliografia

Literatura

[1] `https://docs.ros.org/en/foxy/index.html`

[2] `https://navigation.ros.org/tutorials/docs/navigation2_with_slam.html`