

**Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska**

STERO

Sprawozdanie z bloku mobilnego

Michał Kwarciński, Kacper Marchlewicz

Warszawa, 2022

Spis treści

| | |
|-------------------------|---|
| 1. Część mobilna | 2 |
| 1.1. Projekt 1 | 2 |
| 1.2. Projekt 2 | 5 |
| 1.3. Laboratorium 3 | 6 |

1. Część mobilna

1.1. Projekt 1

1. Implementacja kodu

Na pierwszym laboratorium stworzyliśmy skrypt do sterowania robotem w ramach jazdy, z określoną prędkością i późniejszym kierunkiem skrętu, po kwadracie, o określonej długości boku. Wszystkie te parametry podawane są w pliku .launch ewentualnie możliwe do modyfikowania w pliku lab1.cpp. W czasie pracy nad projektem zmodyfikowaliśmy go. Dodaliśmy liczenie obecnego położenia na podstawie minionego czasu i zadanej przez użytkownika prędkości. Owe położenie (pozycja x, y oraz kąt) są publikowane na odpowiedni temat. Stworzyliśmy również plik lab1er.cpp. Odbiera on informacje od powyższego tematu, jak i również położenie z gazebo i odometrii. Na podstawie tych informacji liczymy pojedyncze (chwilowe) błędy dla porównania odometrii - gazebo i nasze obliczenia czasowe - gazebo. Uzyskane dwa rodzaje błędów sumujemy. Otrzymane wyniki (błędy chwilowe jak i zsumowane) publikujemy.

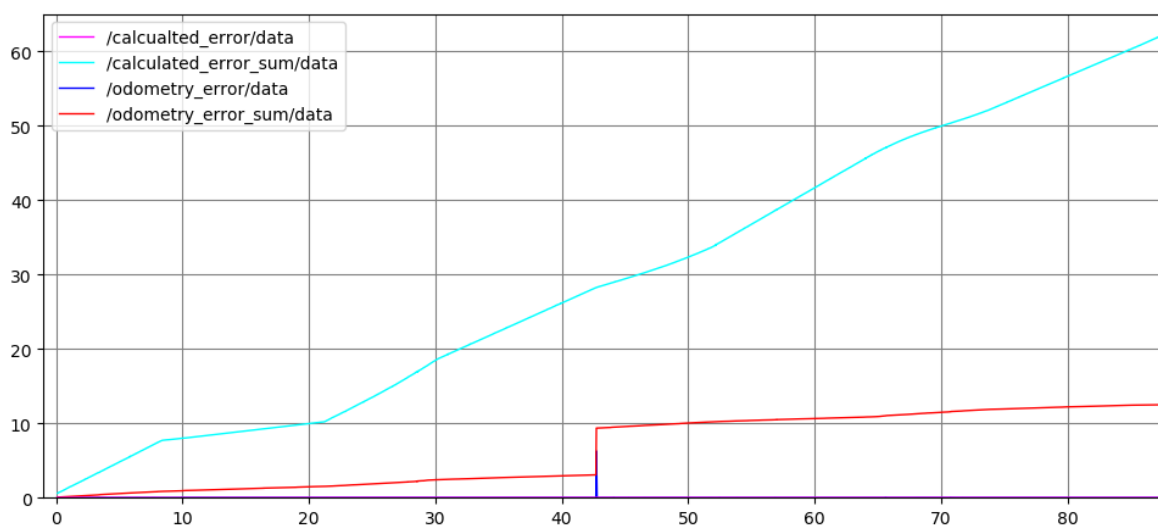
2. Testy

Przy testowaniu dokładności wykonywania kwadratu użyliśmy trzech wersji zadanych prędkości:

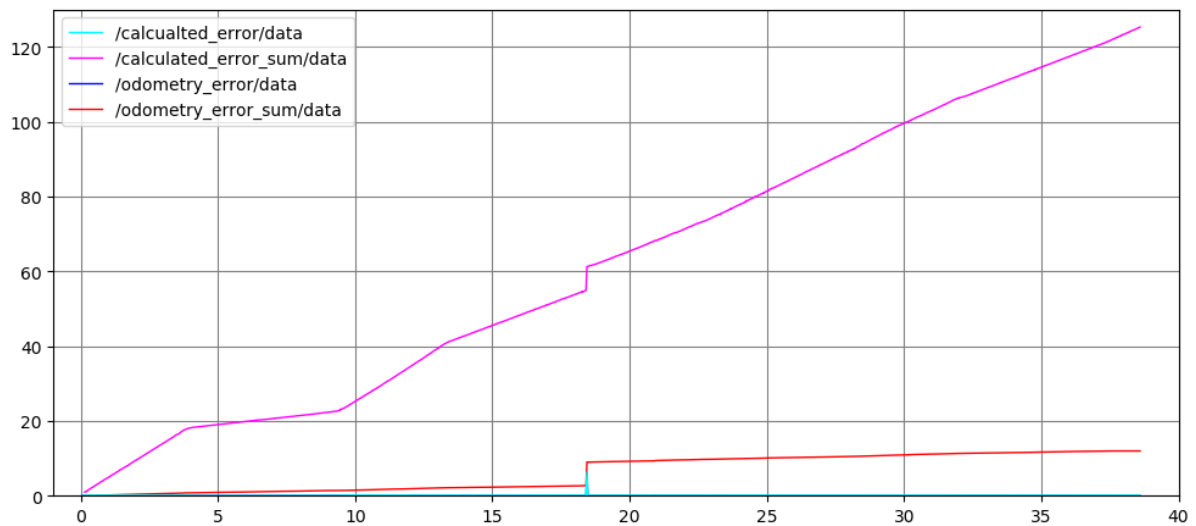
| | Prędkość liniowa | Prędkość obrotowa |
|------------------|------------------|-------------------|
| Prędkości niskie | 0,22631579 | 0,12368421 |

Na podstawie powyższych prędkości możemy wygenerować kolejne zadane prędkości: Prędkości średnie: $2,4 * \text{prędkości niskie}$ Prędkości wysokie: $2,4 * \text{prędkości średnie}$

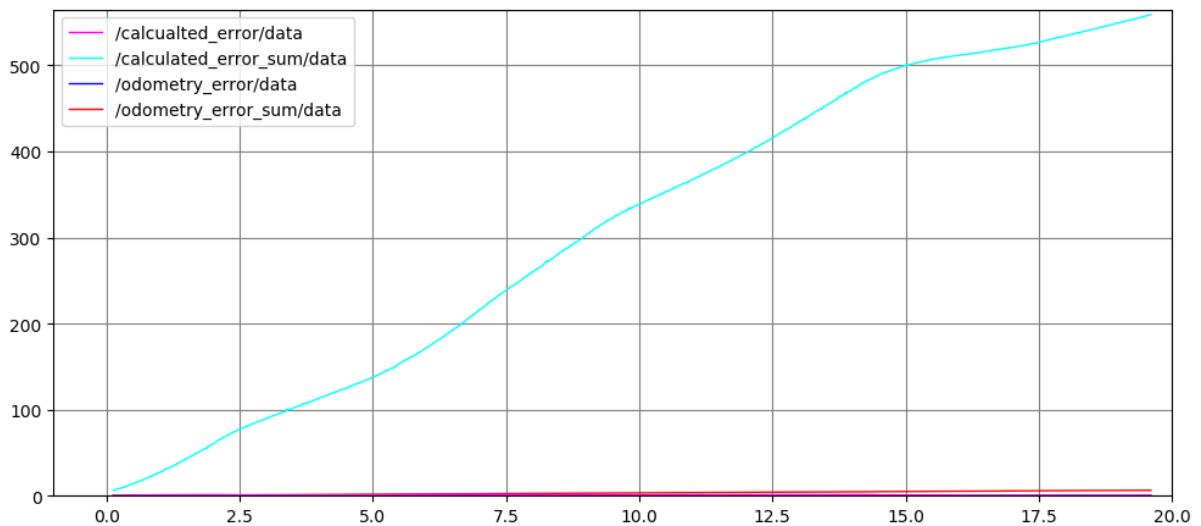
3. Wyniki: Dla sterowanie odometrycznego: Błąd dla prędkości niskich:



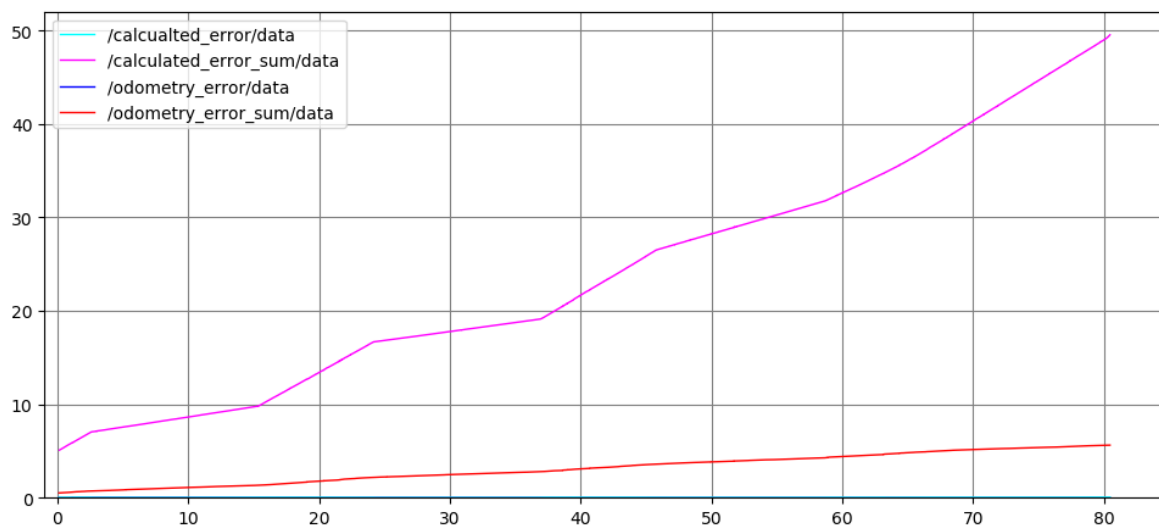
Błąd dla prędkości średnich:



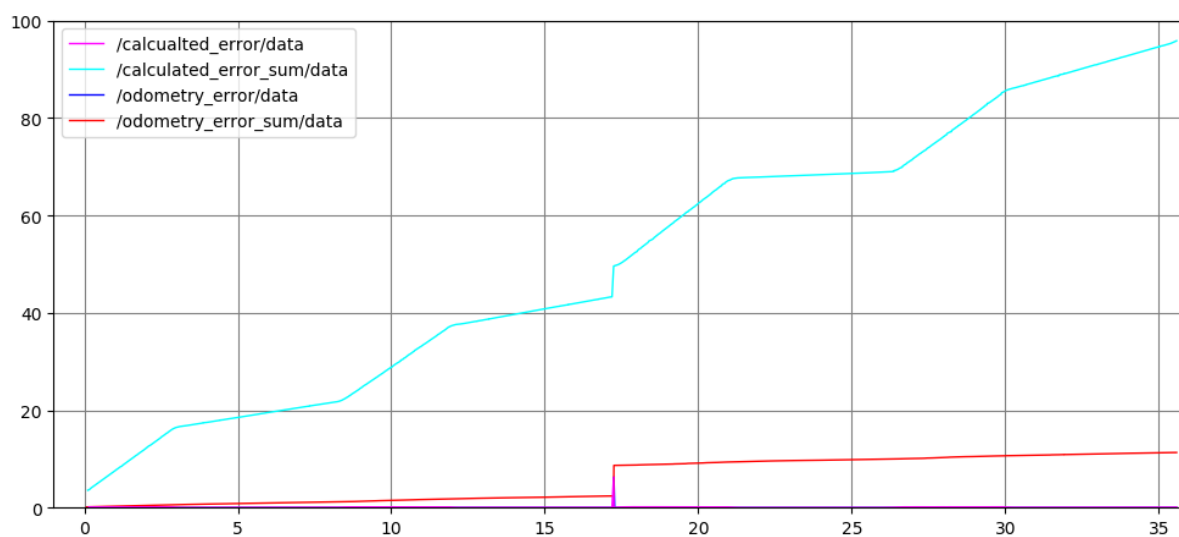
Błąd dla prędkości wysokich:



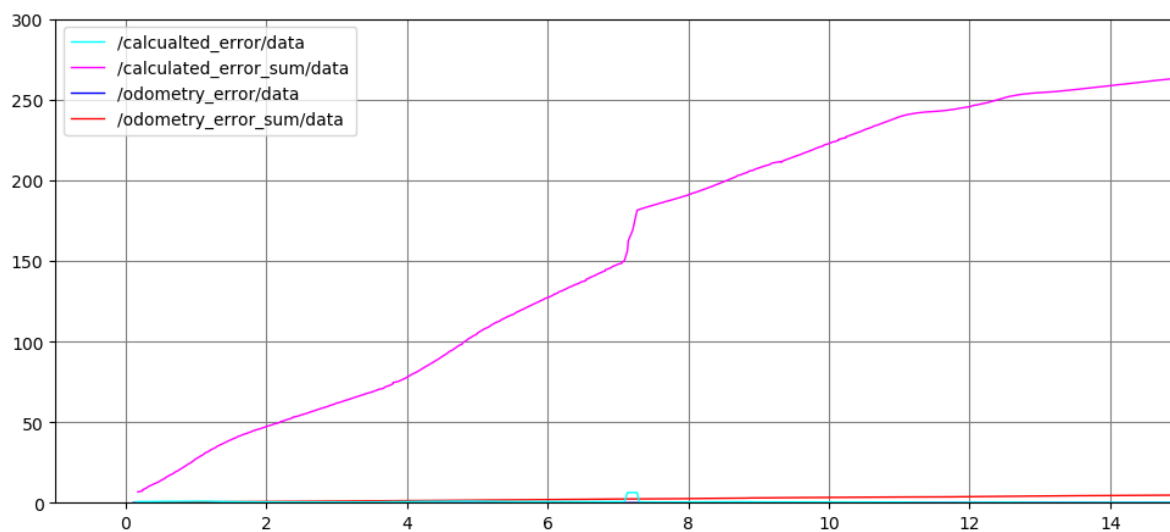
Dla sterowania zakładającego idealne wykonanie zadanych prędkości: Błąd dla prędkości niskich:



Błąd dla prędkości średnich:



Błąd dla prędkości wysokich:



4.Wnioski: Na każdym wykresie suma błędów ciągle się zwiększa. Są obserwowalne w miarę okresowe zmiany narastanie błędów w wyniku zmiany stanu z jazdy na zakręt. Widoczny na wykresie prędkości średnich i szybkich nagły wzrost błędów pojawia się w momencie przechodzenia kąta z wartości -P do P. Widać również jak maleje czas symulacji wraz ze wzrostem prędkości. Można zauważyć, że błąd rośnie wraz z wzrostem prędkości, jest to naturalne, ponieważ wraz ze wzrostem prędkości rośnie niedokładność w stosowaniu prędkości nie patrząc na to, że prawdziwa maszyna (a taką symuluje gazebo) najpierw przyśpiesza a potem musi jeszcze zwolnić. Błędy odometryczne są znacząco mniejsze (około rząd wielkości) od błędów obliczonych na podstawie prędkości, pokazuje to przewagę lokalizowania robota na podstawie odometrii.

1.2. Projekt 2

1.Implementacja kodu

Na drugim laboratorium stworzyliśmy dwie mapy w rosie: „korytarz” i „piętro”. Na tych mapach będziemy testować stworzony w ramach projektu drugiego: node do nawigacji robota mobilnego (Tiago), parametry bibliotek do nawigacji, plik launchowego, pliki „świata”, które zawierają informacje o naszych dwóch mapach. Korzystamy z gotowych bibliotek *costmap_2d*, *global_planner*, *base_local_planner*, *rotate_recovery* i *base_local_planner*, do inicjalizacji działania lokalnego i globalnego plannerów, zachowania „odkleszczania” i lokalnej mapy kosztów.

Nasz węzeł ROS:

- Subskrybuje */move_base_simple/goal* – otrzymujemy informacje o punkcie docelowym zadane przez nas w Rvizie
- Korzysta z lokalnego i globalnego plannerów, lokalnej mapy kosztów i „odkleszczania”
- Wysyła żądania zmian prędkości do sterownika robota
- Realizuje bezpieczny ruch do kolejnych pozycji
- Zwraca stan wykonania

2.Rezultaty

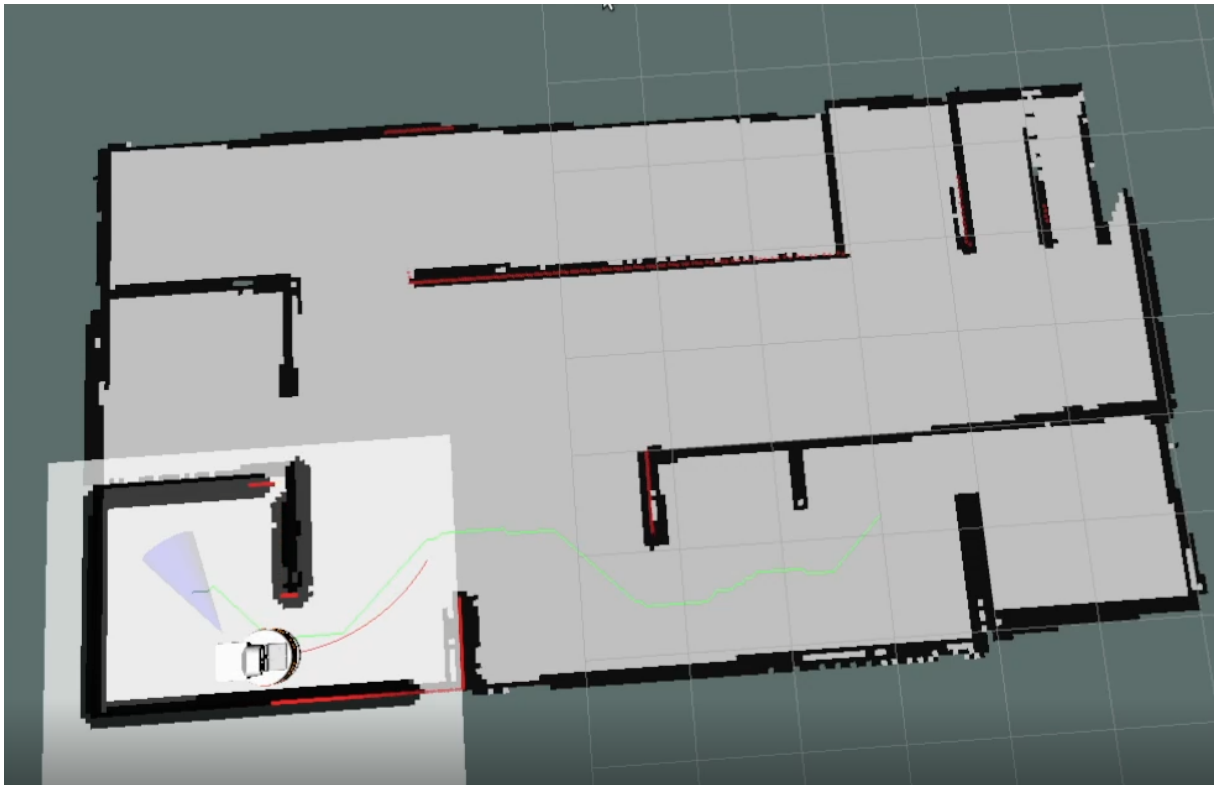
Ostatecznie otrzymaliśmy poprawną symulację. Robot poruszał się poprawnie po mapach. Czasami poruszając się po globalnym planie, najczęściej w miejscach, gdzie algorytm planowania A* generował „ostre” zakręty, robot uruchamiał „RecoveryBehaviors” robił obrót o około 360 stopni i kontynuował jazdę na zadanej trasie. Wjeżdżał poprawnie w wejścia o szerokości do obwodu robota (54 centymetry – z dokumentacji Tiago).

Z tego co wyczytaliśmy na forum rosa błędy widoczne na filmiku ekstrapolacji wynikają z działania na maszynie wirtualnej.

Na widocznym poniżej zrzucie ekranu widać jedną z dwóch map.

Widać na niej:

- Zieloną trajektorie – plan globalny
- Czerwoną trajektorie – plan lokalny
- Wybielony kwadrat z robotem w centrum – obszar „generacji” plany lokalnego, przemieszcza się razem z robotem
- Ciemno-szary kolor obok ścian – działanie warstwy inflacji (żeby robot w miarę możliwość nie jeździł obok ścian)

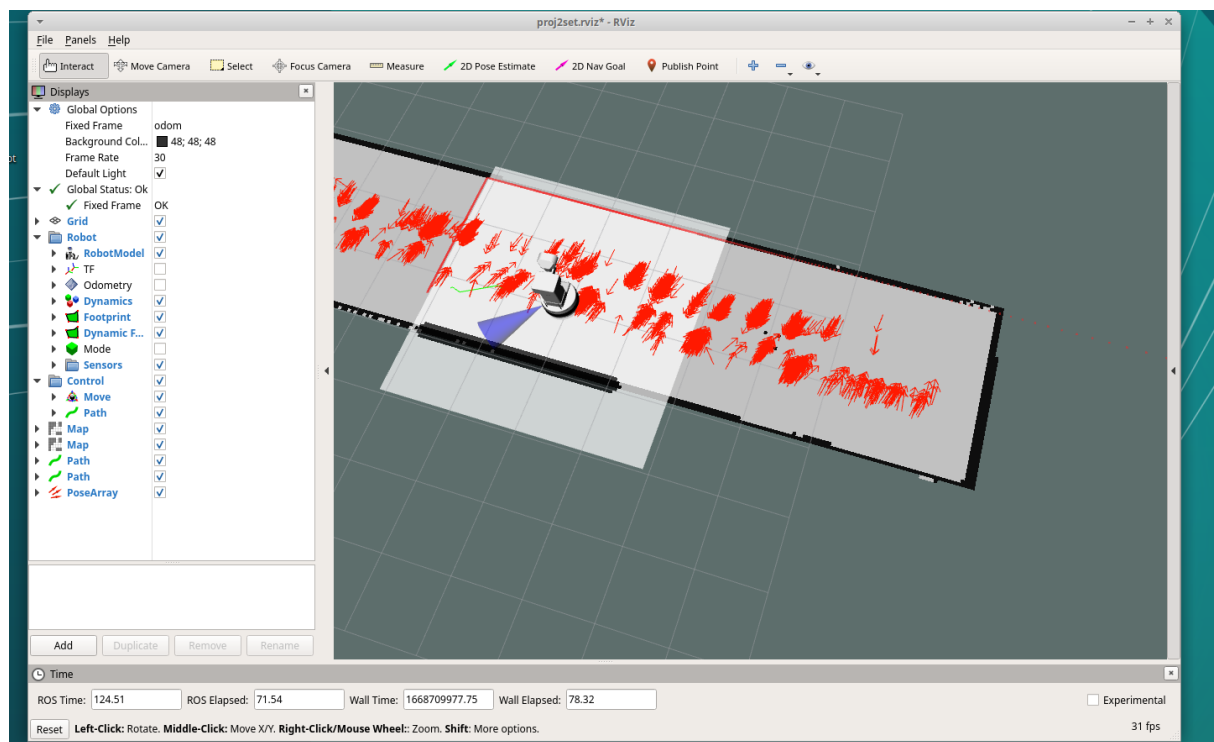
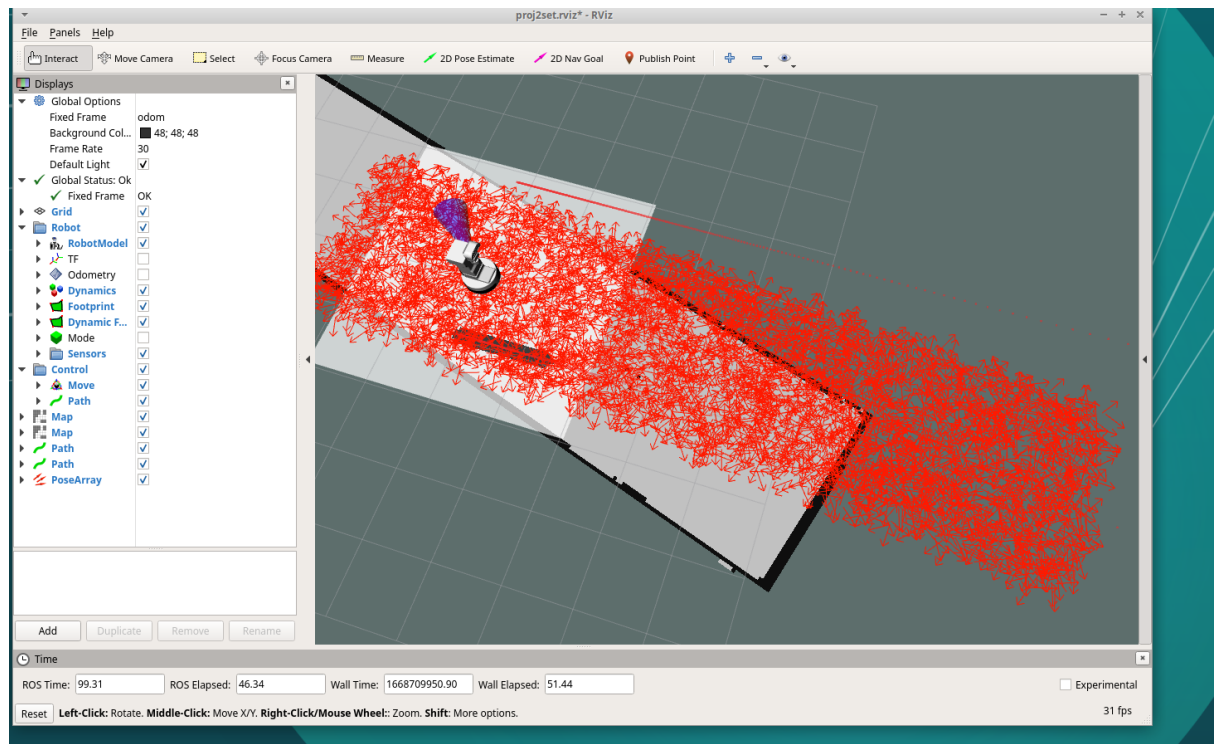


1.3. Laboratorium 3

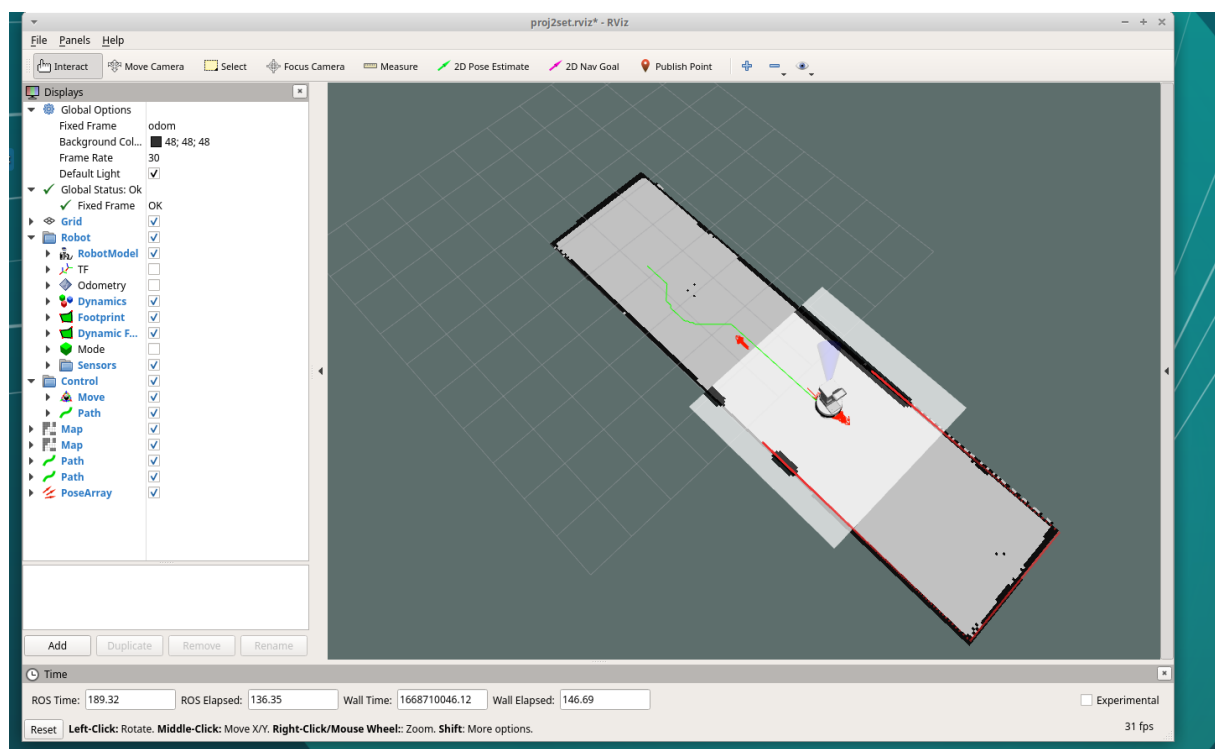
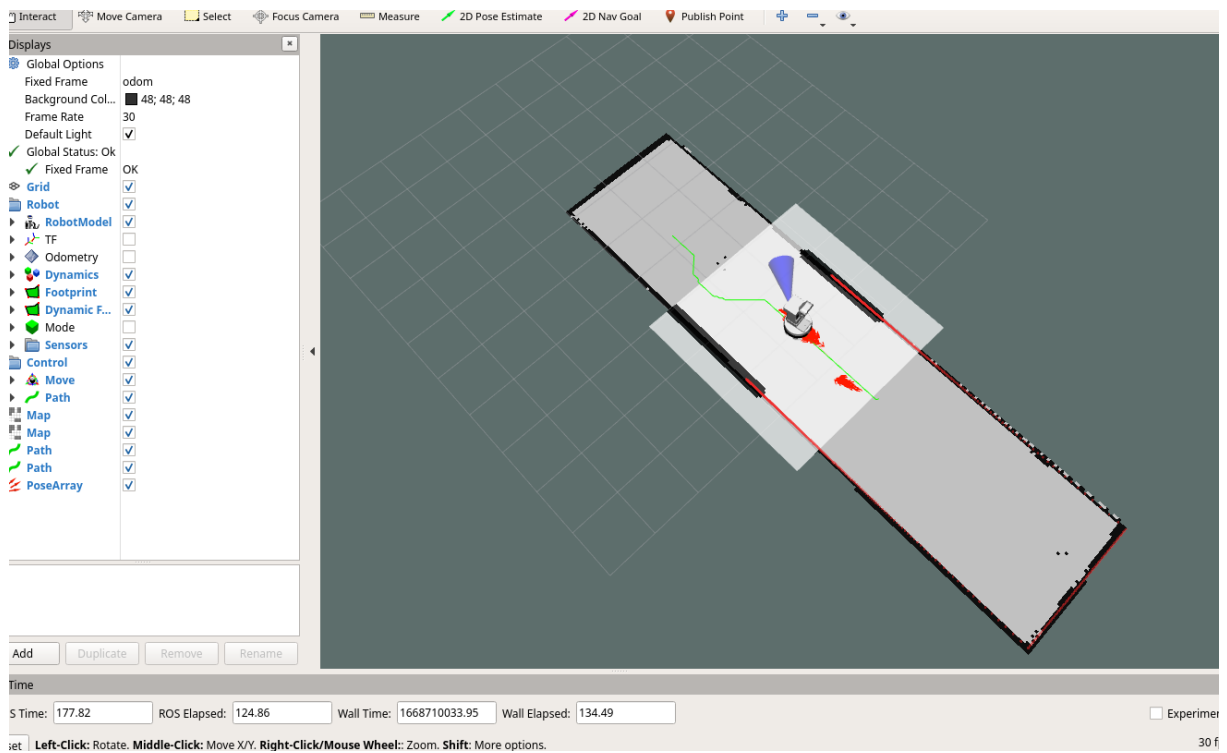
Zakomentowaliśmy, używaną do projektu 2 i wcześniej, linijkę w pliku launch odpowiadającą za static transformation. Zamiast niej użyliśmy lokalizacji globalnej z algorytmu AMCL. Na tym laboratorium testujemy działanie powyższej metody na stworzonych wcześniej mapach (korytarz i labirynt). Z domyślnymi parametrami rezultaty są zadowalające 1.3 i 1.3. Obserwując umiejscowienie strzałek na początku i w trakcie jazdy, można dojść do wniosku, że na początku robot nie dokładnie jest w stanie się zlokalizować (duże rozstrzelanie strzałek obok robota 1.3 i 1.3). Potem w trakcie jazdy strzałki zdecydowanie zbliżają się do robota. Przy za małej ilości cząsteczek robot ma zdecydowane problemy z poruszaniem się 1.3. Potrafi nagle uznać, że znajduje się w ścianie albo gubi się i porusza się w zdecydowanie większej odległości od ścieżki globalnej.

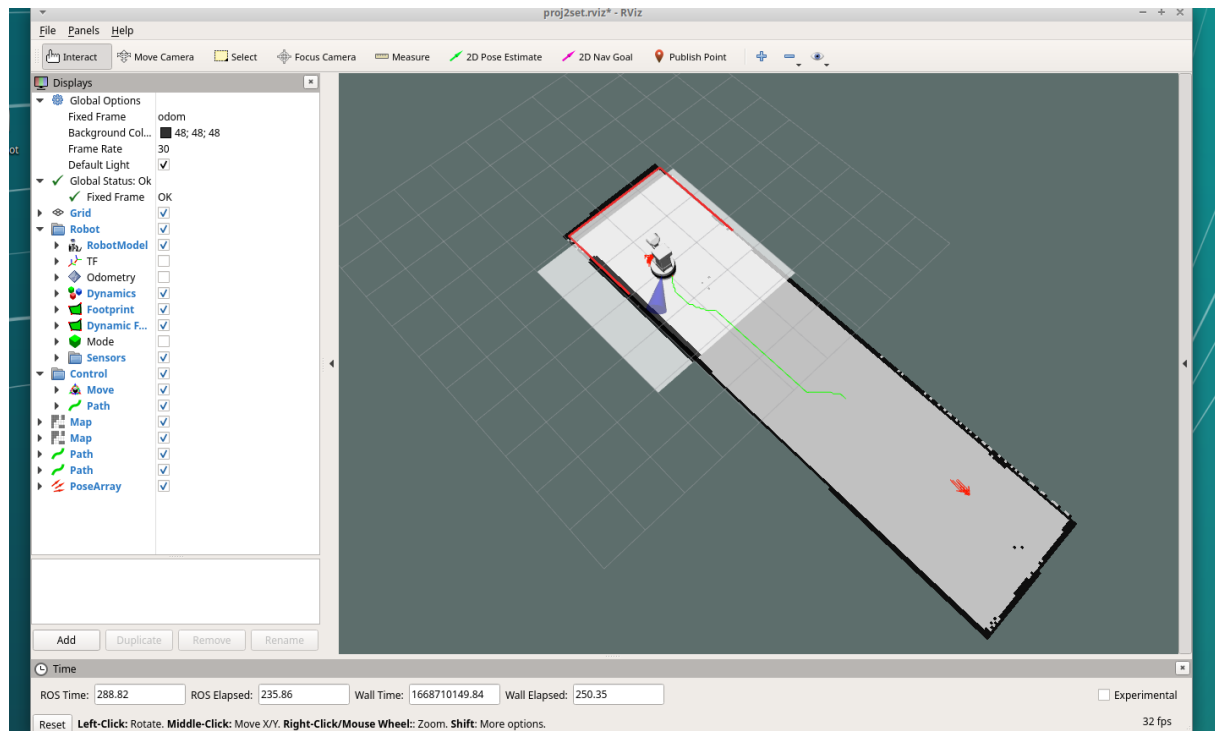
Rezultaty dla korytarza:

Dla za dużej ilości cząstek:



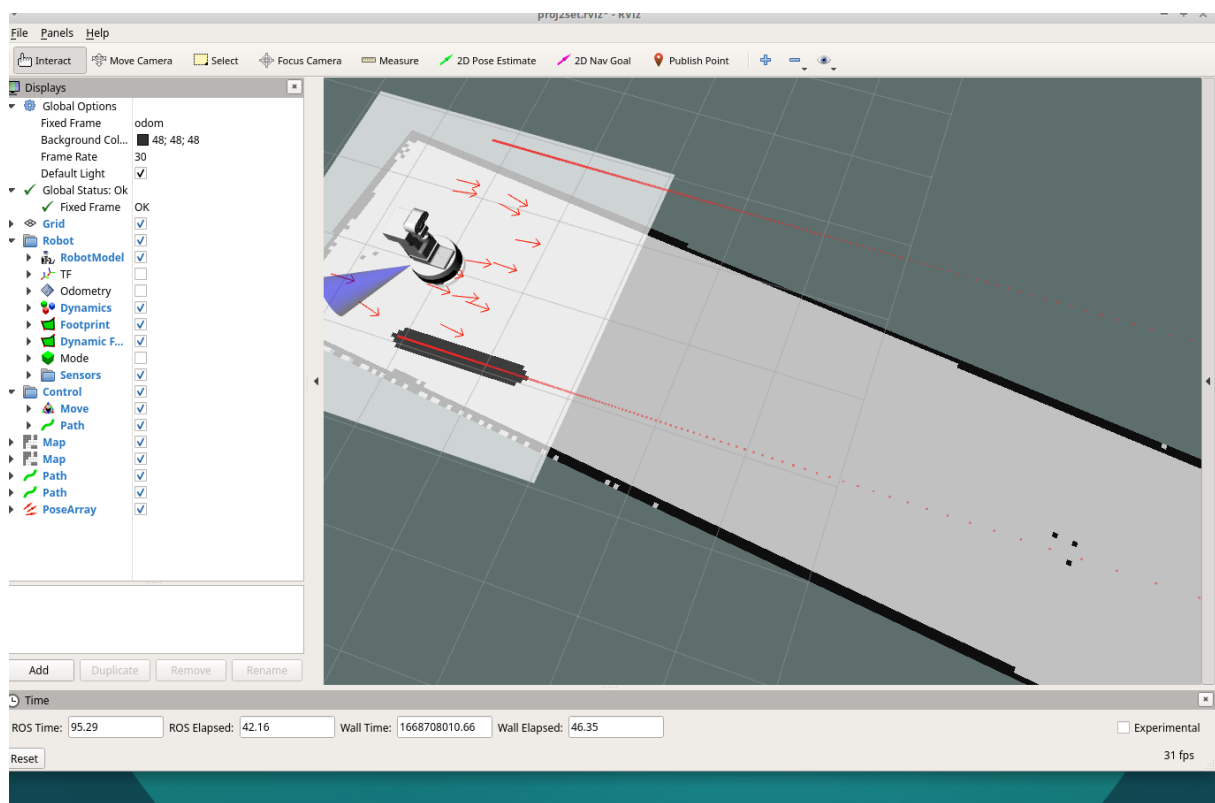
Dla domyślnych parametrów:

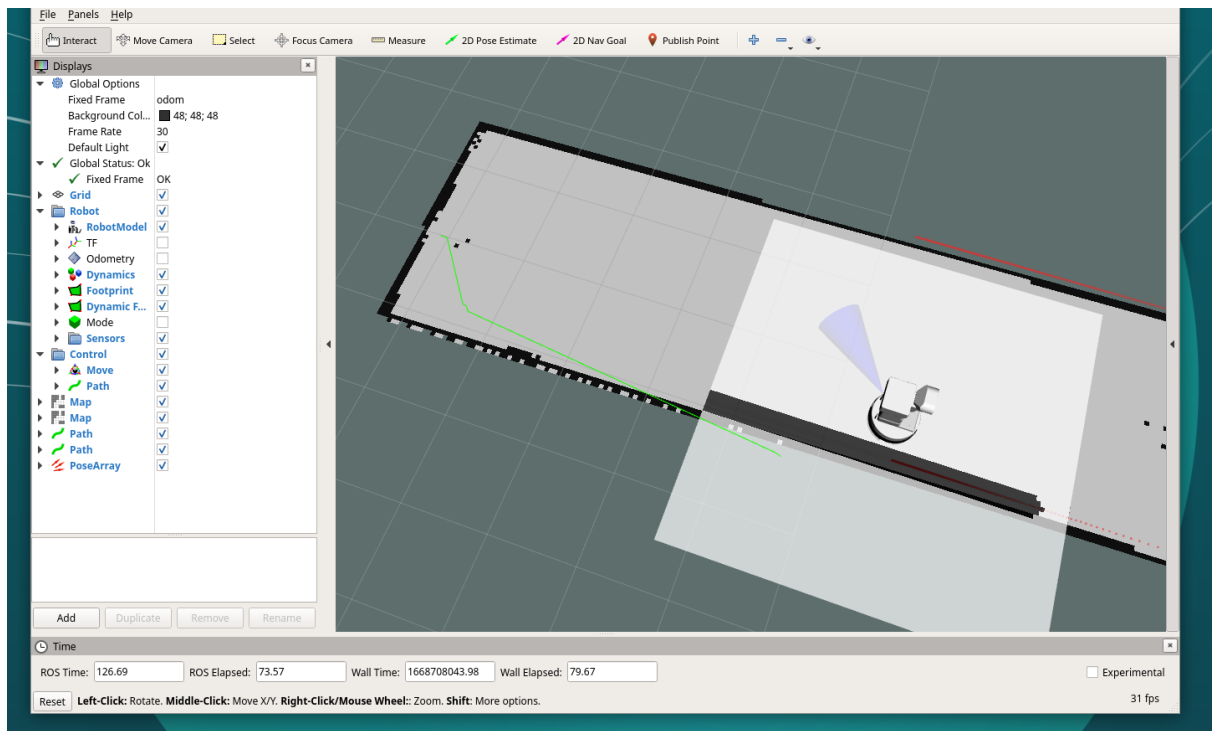
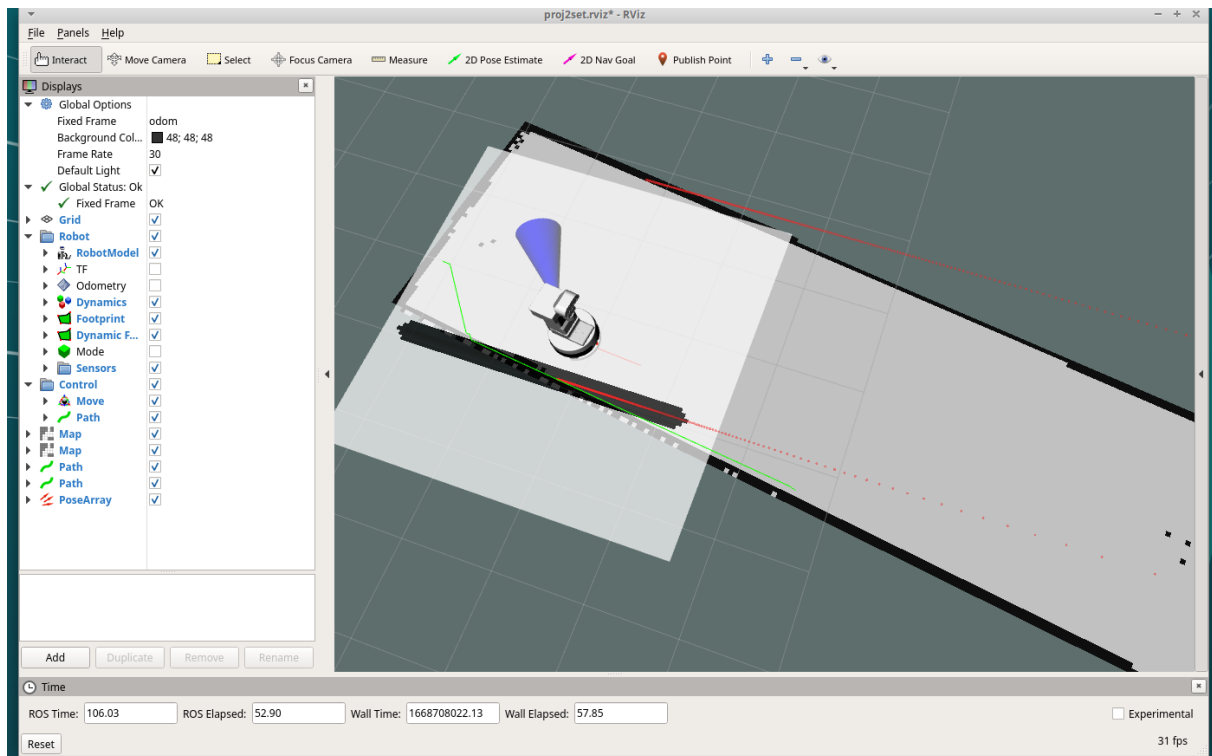




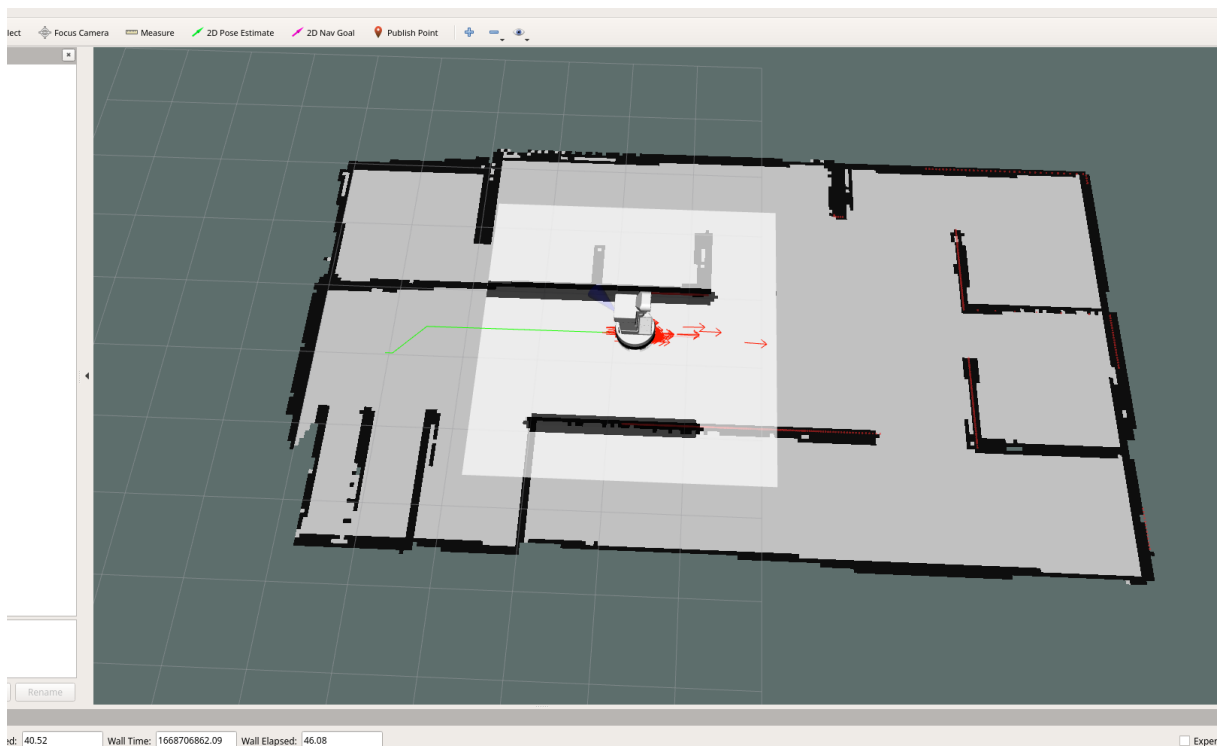
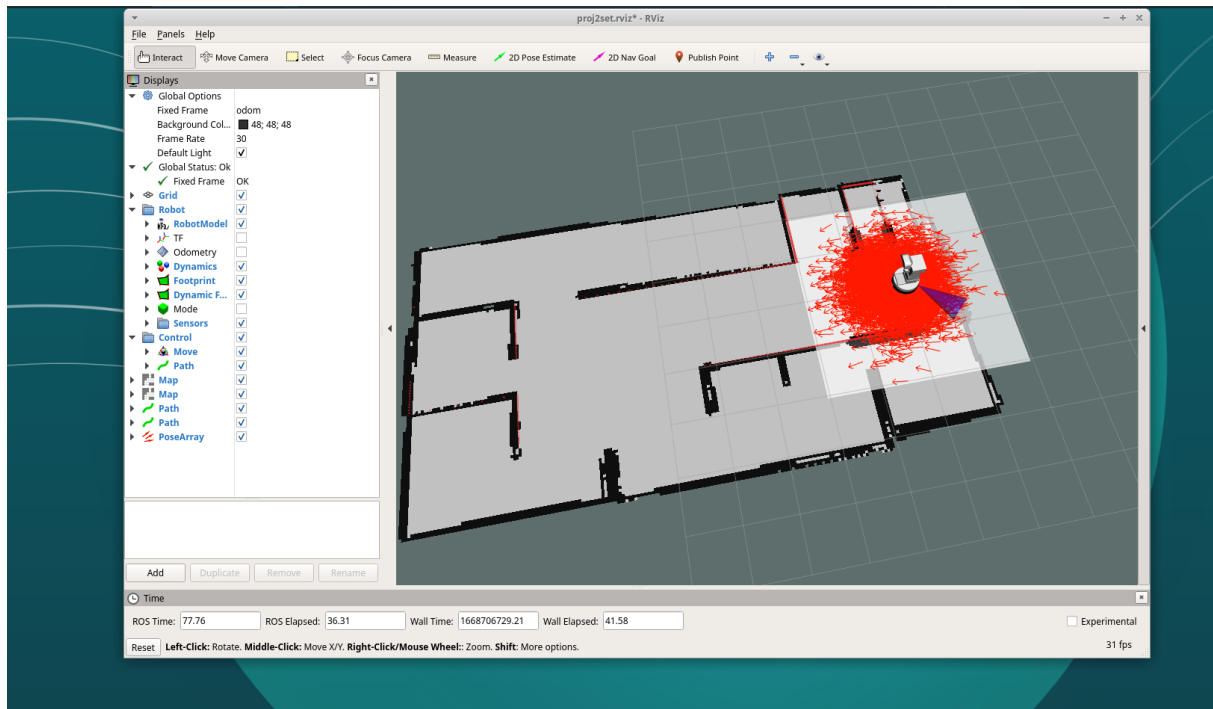
Można zauważyć, że AMCL nie jest w stanie czasami określić gdzie robot się znajduje. Pokazuje, że równie prawdopodobne są dwie lokalizacje i wraz z postępującym ruchem robota się to nie zmienia, ponieważ na podstawie odczytów z czujników na korytarzu, który jest symetryczny z obydwu końców, nie da się jednoznacznie określić która lokalizacja jest poprawna.

Przy za małym parametrze:





Dla labiryntu:
Dla dużej ilości cząsteczek:



Dla parametrów domyślnych:

