Politechnika Poznańska
Wydział Automatyki, Robotyki i Elektrotechniki

Konrad Jędrzejewski

Autonomiczne Roboty Mobilne

Laboratorium Problemowe 2

Spis treści

1 Wstęp **3**
1.1 Omówienie realizowanego tematu 3

2 Struktura programu **3**
2.1 Tworzenie mapy zajętości 3
2.2 Eksploracja środowiska 3
2.3 Pomiar czasu i sprawdzenie osiągnięcia celu 3
2.4 Graf węzłów i tematów ROS2 4

3 Wyniki **4**

4 Wnioski **6**

1. Wstęp

1.1. Omówienie realizowanego tematu

Rozważanym problemem jest stworzenie programu umożliwiającego robotowi mobilnemu Turtlebot3 Waffle funkcjonowanie w środowisku z nieznaną mapą oraz zlokalizowanie i dotarcie do wyznaczonego punktu w labiryncie w jak najkrótszym czasie. Robot wyposażony jest w skaner laserowy o zasięgu 3,5 metra oraz sensor IMU.

Implementację algorytmu rozwiązującego przedstawiony problem umieszczono w repozytorium na serwisie GitHub.

2. Struktura programu

2.1. Tworzenie mapy zajętości

Robot nie posiada wiedzy na temat środowiska, w którym się znajduje, dlatego musi sam ją zdobywać i przetwarzać. Posiadając położenie jego samego z tematu */odom* i położenie obiektów w jego otoczeniu z tematu */scan* agent jest w stanie stworzyć mapę zajętości i się w niej lokalizować. Wykorzystano gotową paczkę slam_toolbox. Umożliwia ona tworzenie mapy zajętości na bieżąco publikując ją w temacie */map*.

2.2. Eksploracja środowiska

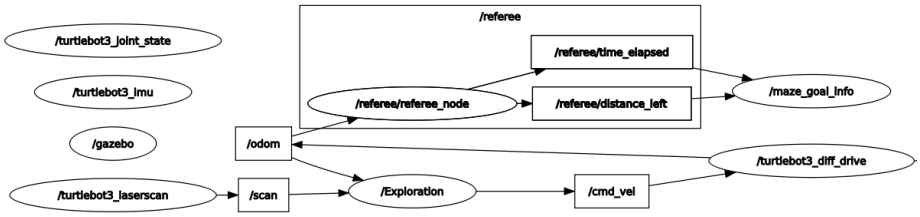
Odnalezienie punktu docelowego wewnątrz labiryntu może być niemożliwe wykorzystując metodę trywialną. Ponieważ robot nie posiada wiedzy o labiryncie pojawiając się w nim, skuteczniejszym sposobem jest ciągła eksploracja i badanie nieodkrytych fragmentów środowiska. Ciągłe poszukiwanie nowych miejsc na mapie gwarantuje agentowi dostanie się do punktu docelowego, jeśli tylko istnieje do niego ścieżka. Wykorzystano paczkę ROS2 autonomous_exploration, w której zaimplementowana została metoda Frontier Based Exploration. Polega ona na wyznaczaniu ścieżki robota w miejsca na mapie zajętości, w których odkryty i pusty obszar sąsiaduje z nieodkrytym. Uruchomienie tej paczki tworzy węzeł */Exploration*, który subskrybuje tematy */odom*, */scan* i */map*, wyznacza ścieżkę robota podczas eksploracji i publikuje jego prędkości w temacie */cmd_vel*.

2.3. Pomiar czasu i sprawdzenie osiągnięcia celu

Do weryfikacji w jakim czasie robot dostał się do wyznaczonego punktu w labiryncie, napisano paczkę maze_goal. Zawiera ona implementację węzła */maze_goal_info*, który subskrybuje tematy */referee/distance_left* oraz */referee/time_elapsed*. Węzeł wyświetla odległość robota od punktu docelowego i czas od uruchomienia symulacji, oraz informuje gdy ten osiągnie cel.

2.4. Graf węzłów i tematów ROS2

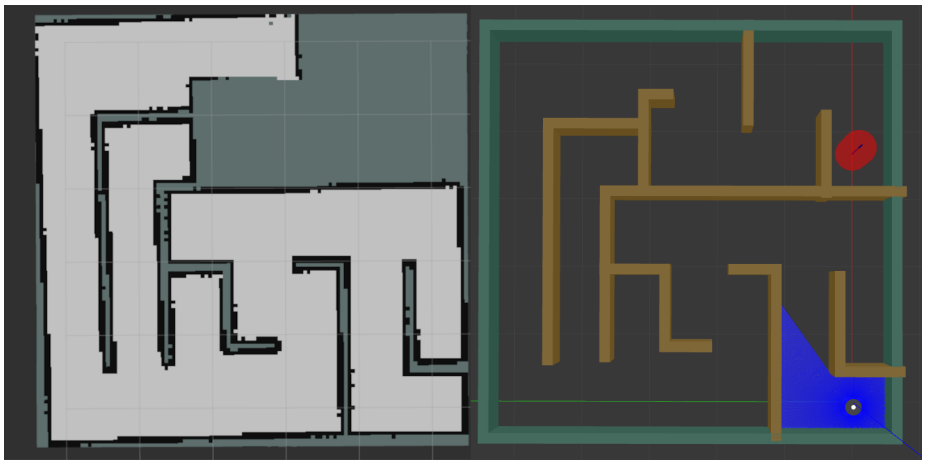
Graf przedstawiający strukturę programu z uwzględnieniem węzłów i tematów ROS2 przedstawiono na poniższym rysunku 1.



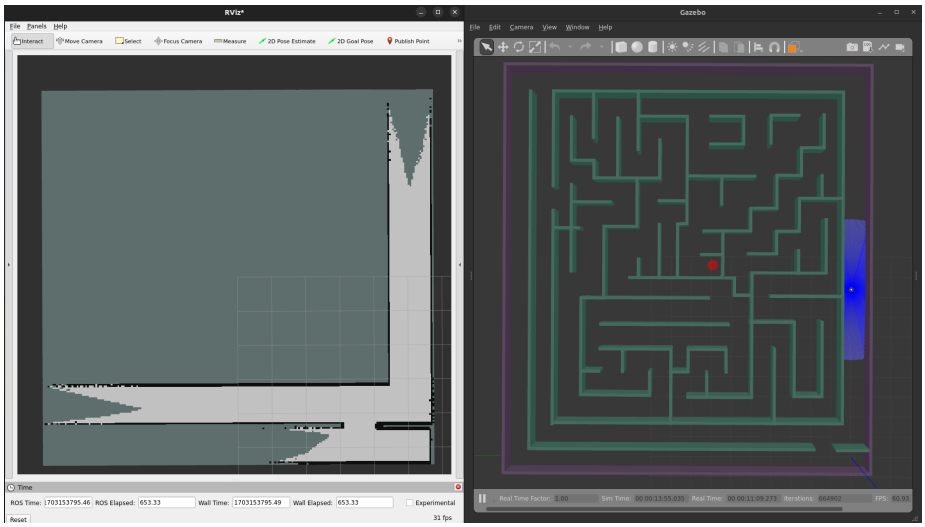
Rysunek 1: Graf węzłów i topików ROS2.

3. Wyniki

Udało się skutecznie zaimplementować program umożliwiający robotowi Turtlebot3 Waffle rozwiązywanie nieznanych labiryntów. Agent poprawnie buduje mapę otoczenia na podstawie pomiarów ze skanera laserowego oraz IMU. Na rysunkach 2-3 przedstawiono widok środowiska wraz z tworzoną mapą podczas eksploracji labiryntu.

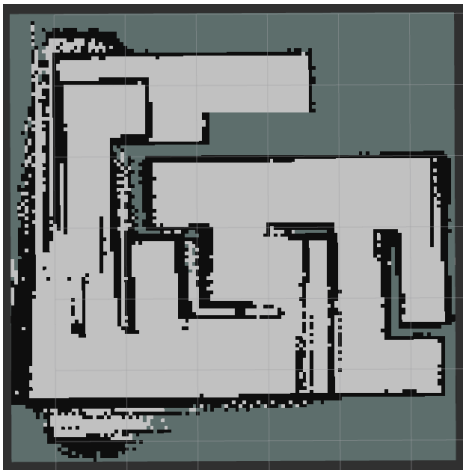


Rysunek 2: Środowisko w Gazebo (prawo) oraz utworzona mapa zajętości (lewo) - mały labirynt.



Rysunek 3: Środowisko w Gazebo (prawo) oraz utworzona mapa zajętości (lewo) - duży labirynt.

Podczas prób osiągnięcia punktu docelowego występował błąd przy tworzeniu mapy zajętości. Robot gubił swoją lokalizację i nadpisywał w sposób przekłamany mapę zajętości. Skutek tego błędu przedstawiono na rysunku 4. Błąd ten występował również przy rozwiązywaniu poprzednich zadań laboratoryjnych ARM wykorzystujących SLAM i wynika on z ograniczeń sprzętowych komputera, na którym uruchamiana była symulacja.



Rysunek 4: Wystąpienie błędu podczas tworzenia mapy zajętości.

4. Wnioski

- Metoda Frontier Based Exploration nie zwraca optymalnej ścieżki pod względem czasu, ponieważ nie można takiej wyznaczyć nie posiadając całkowitej wiedzy na temat środowiska. Metodę tę można by usprawnić wykorzystując znaną odległość robota do punktu docelowego (/referee/distance_left), jednak nie gwarantuje to większej skuteczności w każdym labiryncie.