Inteligentne Maszyny

Skrypt do ćwiczeń

Semestr: 2022 L

Autor: Wojciech Dudek

Spis treści

SEKCJA 1

Zadania

1.1 Działania na przestrzeniach pracy

Zadanie polega na utworzeniu przestrzeni pracy w katalogu HOME. Przestrzeń pracy:

- ma mieć nazwę (przestrzeń pracy zlokalizowana w katalogu): **my_catkin_ws**,
- ma rozszerzać pakiety ROS zlokalizowane w katalogu:
  `/opt/ros/melodic`
- ma zawierać szkielet pakietu, który:
  - ma nazwę: `package-<nr-indeksu>`
  - ma zależeć od pakietów: rospy, geometry_msgs, nav_msgs, std_msgs
  - ma mieć określoną wersję: 0.0.3
  - ma mieć autora: <nr-indeksu>
- ma zawierać pakiet z repozytorium: https://github.com/RCPRG-ros-pkg/ros_tutorials pobranego w inne miejsce w katalogu użytkownika (np. $HOME/git),

Drugą częścią zadania jest utworzenie drugiej przestrzeni pracy, która:

- ma nazwę: **your_catkin_ws**
- rozszerza przestrzeń pracy **my_catkin_ws**,
- zawiera szkielet pakietu, który:
  - zależy od pakietów: package-<nr-indeksu>, rospy, geometry_msgs, nav_msgs, std_msgs,
  - ma nazwę: **robot_package**

1.2 Narzędzia ROS w konsoli

Aby testować narzędzia terminala należy najpierw uruchomić 'rosmaster' poleceniem:

```
> roscore
```

UWAGA! Jeśli konsola wypisze błąd, należy się zsource'ować do odpowiedniego pliku konfiguracyjnego (np. source /opt/ros/melodic/setup.bash).

Następnie sprawdzamy działanie wszystkich poleceń wskazanych na slajdach pierwszego wykładu (dostępnego na moodle):

1. Uruchomienie węzła publikującego wiadomość tekstową,
2. Uruchomienie węzła subskrybującego wiadomość tekstową (z wypisywaniem treści wiadomości)
3. Wypisanie aktywnych węzłów,
4. Sprawdzenie informacji o węzłach,
5. Sprawdzenie typu tematu na którym węzły się komunikują,
6. Sprawdzenie struktury wiadomości wymienianej na określonym temacie,
7. Sprawdzenie aktualnych wartości zmiennych systemowych ROS:
   - ścieżek do pakietów ROS,
   - adres rosmaster'a,
   - adres ROS komputera na którym uruchomiona jest konsola,
8. wypisanie listy wszystkich dostępnych pakietów,
9. Zabicie obu uruchomionych węzłów za pomocą polecenia:

   `> rosnode kill <nazwa-węzła>`

   oraz obserwacja efektu przez polecenie:

   `> rosnode list`

   oraz przez spojrzenie na konsolę, w której był uruchomiony zabity węzeł.
10. Uruchomienie węzła publikującego i subskrybującego z pakietu samouczka:

    `> rosrun roscpp_tutorials talker`

    `> rosrun roscpp_tutorials listener`
11. Wywołanie poleceń ze slajdów prezentacji z pierwszego wykładu dotyczących tych węzłów (tytuły slajdów: "przykład użycia konsoli – talker", "przykład użycia konsoli – topic", "przykład użycia konsoli – listener")
12. Uruchomienie węzła publikującego wiadomość z określoną częstotliwością.

1.3 Analiza symulowanego systemu robota

Pierwszym etapem jest uruchomienie robota turtlebot w symulatorze Gazebo. Najpierw należy określić rodzaj robota turtlebot, którego chcemy symulować. Robi się to przez ustawienie odpowiedniej zmiennej środowiskowej:
`> export TURTLEBOT3_MODEL=waffle_pi`
Symulację uruchamia się przez polecenie:
`> roslaunch turtlebot3_gazebo turtlebot3_world.launch`
Drugim etapem jest zapoznanie z oknem symulatora. Należy wykonać następujące czynności:

1. dodać prosty kształt do symulowanego świata,
2. przenosić go za pomocą kolorowych osi układu współrzędnych obiektu,
3. obracać obiekt,

4. skalować obiekt,
5. dodać inne modele z bazy modeli gazebo.org:
   (a) W zakładce 'insert' pojawi się drzewo ścieżek do modeli,
   (b) Rozwinąć gałąź: http://gazebosim.org/models
   (c) Poprzez kliknięcie nazwy modelu i przeniesienie kursora na ekran wizualizacji świata,
6. Włączyć wizualizację poszczególnych elementów robota turtlebot:
   (a) kolizje,
   (b) środek masy,
   (c) punkty kontaktu,
   (d) złącza,

Trzecim etapem jest analiza systemu sterowania robota turtlebot. W tym celu należy:

1. uruchomić system nawigacji robota turtlebot:

   `roslaunch turtlebot3_navigation turtlebot3_navigation.launch`
2. uruchomić narzędzie rqt_graph:

   `> rosrun rqt_graph rqt_graph`
3. zadać różne pozycje lokalizacji robota za pomocą programu Rviz (przycisk 2D Pose Estimate w górnym pasu narzędzi)
4. zadać cel nawigacji za pomocą programu Rviz (przycisk 2D Nav Goal w górnym pasku narzędzi)
5. zapoznać się z obiektami/liniami wizualizowanymi przez program Rviz,
6. Zadać cel ruchu robota z terminala poprzez interfejs tematu:

   `/move_base_simple/goal`
7. napisać (w C++) węzeł publikujący cel ruchu na powyższym temacie. Cel ruchu jest przekazywany do węzła jako argument. Węzeł ma być utworzony w pakiecie **package-<numer-indeksu>** w przestrzeni pracy **my_catkin_ws** oraz skompilowany za pomocą polecenia:

   `> catkin build`.

   UWAGA! Przy pisaniu węzła można się wspierać samouczkiem: http://wiki.ros.org/roscpp/Overview/Publishers%20and%20Subscribers.
8. Wysłać pakiet na własne repozytorium w grupie `ima-22l` o nazwie: <pierwsza litera imienia><nazwisko>. Adres przykładowego adresu repozytorium dla studenta Adam Kowalski: https://gitlab-stud.elka.pw.edu.pl/ima-22l/akowalski

   Przydatne! Można użyć poleceń:
   `git init`,
   `git remote add origin <adres-repo>`,
   `git commit -m <treść wiadomości>`,
   `git push origin master`.