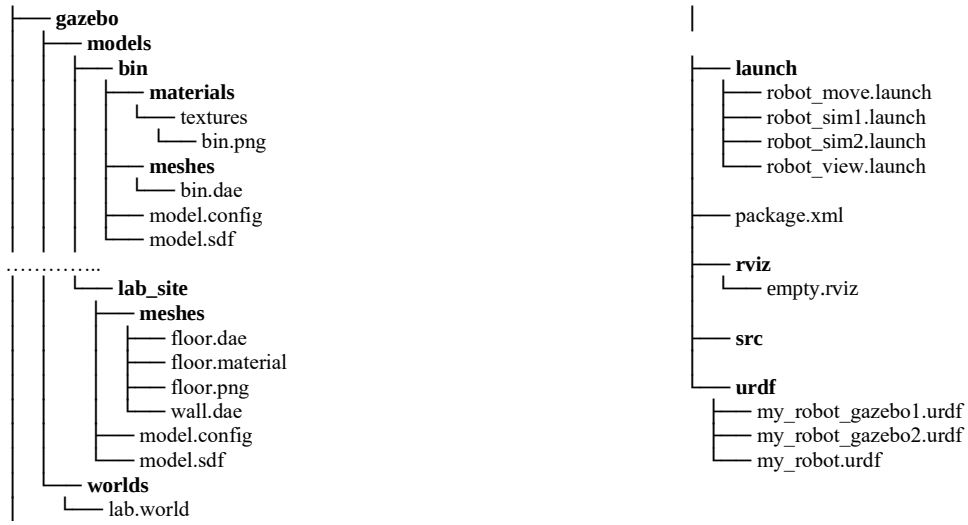
LABORATORIUM 08
ŚRODOWISKO SYMULACYJNE W ROS: STEROWANIE I WIZUALIZACJA ROBOTA.

Cel zajęć

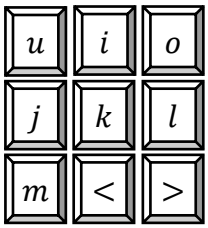
Poznanie podstawowych elementów systemu ROS pozwalających na pracę w symulowanym środowisku.

Konfiguracja

W systemie został umieszczony dodatkowy pakiet o nazwie *my_robot*. Struktura folderów zawierających źródła tego pakietu została przedstawione na poniższym rysunku.



Dodatkowo w systemie został zainstalowany pakiet *teleop_twist_keyboard*, który umożliwia ręczne sterowanie robotem z klawiatury i publikuje komendy ruchu na topicu */cmd_vel*. Układ klawiszy sterujących na klawiaturze odwzorowuje kierunki ruchu robota:



i, <	do przodu i do tyłu
j, *l*	skręt w lewo i w prawo
u, *o*	do przodu w lewo i w prawo
m, >	do tyłuw lewo i w prawo

q, *z*	zwiększenie i zmniejszenie prędkości o 10%
w, *x*	zmiana prędkości liniowej o 10%
e, *c*	zmiana prędkości kątowej o 10%
pozostałe	stop

Ćwiczenia

Uruchom system ROS i wykonaj poniższe polecenia(wyniki zapisz w sprawozdaniu).

1. Otwórz plik *my_robot.urdf*, który zawiera opis struktury robota. Elementami tej struktury są:
   - **linki**, które reprezentują sztywne części robota,
   - **jointy** (połączenia), które łączą linki i definiują sposób ich ruchu względem siebie.

   Przypisz nazwy elementów do odpowiedniej kategorii: **link** lub **joint** (lista nazw znajduje się w sprawozdaniu, wystarczy zidentyfikować po 5 linków i 5 jointów).

2. Uruchom plik launch: *robot_view.launch*. Plik ten ładuje model robota zawarty w pliku *urdf* na serwer parametrów w ROS oraz uruchamia 2 węzły: węzeł *robot_state_publisher* z pakietu *robot_state_publisher* oraz węzeł *rviz* z pakietu *rviz*. Zadaniem węzła *robot_state_publisher* jest publikowanie stanów przegubów robota na podstawie modelu URDF oraz wyznaczanie pozycji i orientacji poszczególnych linków robota, natomiast węzeł *rviz* umożliwia wizualizację samego robota i danych z ROS w formie graficznej.

   a) Po uruchomieniu pliku zwróć uwagę na błąd (pole Global Status: Error w liście ***Displays*** po lewej stronie okna programu ***RViz***, po rozwinięciu pola wyświetlana jest bardziej szczegółowa informacja: *Fixed Frame [map] does not exist*). Problem można wyeliminować wybierając w polu ***Global Options*** → ***Fixed Frame*** jeden z dostępnych układów współrzędnych (układ *map* nie jest zdefiniowany).

   b) Program ***RViz*** pozwala na wizualizację robota. Robot może być pokazany po wybraniu do wyświetlania elementu ***RobotModel*** (po kliknięciu przycisku ***Add*** w dolnej części listy ***Displays*** i wskazaniu wizualizacji ***RobotModel***). Przeanalizuj informacje wyświetlane w polu, zwróć uwagę na to, że elementy robota mogą być wyłączane i włączane (***Links***). W sprawozdaniu opisz wygląd elementów: *lidar*, *left_wheel*, *front_left_caster*.

   c) Program ***RViz*** umożliwia wizualizację układów współrzędnych przypisanych do poszczególnych linków robota. Aby je wyświetlić, należy dodać do listy wizualizacji element typu ***TF*** (przycisk ***Add***, następnie wybór wizualizacji ***TF***). Przeanalizuj informacje widoczne w listach ***Frames*** i ***Tree***. Zwróć uwagę, że poszczególne elementy list można włączać i wyłączać, co ułatwia analizę struktury. W sprawozdaniu przedstaw hierarchię układów współrzędnych opisującą strukturę robota w formie listy wielopoziomowej – każdy poziom listy powinien odpowiadać kolejnemu stopniowi zagłębienia w drzewie transformacji. Dodatkowo sprawdź i opisz, w jaki sposób zostały rozmieszczone bryły geometryczne reprezentujące linki *base*, *base_footprint* oraz *left_wheel* względem ich lokalnych układów współrzędnych – określ, w jakich kierunkach są zwrócone osie układów względem brył.

   d) Uruchom węzeł *rqt_tf_tree* z pakietu *rqt_tf_tree*. Porównaj pokazywane przez węzeł drzewo układów współrzędnych z wynikami z podpunktu c). Zrób zrzut ekranu i dołącz do sprawozdania.

   e) Zapisz konfigurację programu ***RViz*** do pliku *robot.rviz* w folderze *rviz* pakietu *my_robot* (należy wybrać opcję ***File*** → ***Save Config As***). W pliku konfiguracyjnym zapisane zostaną dodane w tym zadaniu elementy wizualizacji: ***RobotModel*** oraz ***TF***). Plik *robot.rviz* będzie ładowany automatycznie przez plik *robot_sim1.launch* wykorzystywany w zadaniu następnym.

   f) Zamknij ***RViz*** i przerwij działanie pliku *robot_view.launch*, który został uruchomiony na początku tego zadania.

3. Uruchom plik launch: *robot_sim1.launch*. Plik ten wywołuje plik uruchomieniowy z poprzedniego zadania oraz dodatkowo uruchamia symulację robota w środowisku ***Gazebo***. ***Gazebo*** to symulator robotów, który umożliwia testowanie modeli w realistycznym środowisku fizycznym. W pliku *robot_sim1.launch* wczytywany jest świat zapisany w pliku *lab.world* oraz tworzony jest robot na podstawie nowego pliku URDF: *my_robot_gazebo1.urdf*. W porównaniu do modelu z poprzedniego zadania, ten rozszerzony plik URDF zawiera dodatkowe informacje niezbędne dla poprawnego działania w symulatorze ***Gazebo***: definicję brył kolizyjnych oraz właściwości fizyczne poszczególnych elementów robota. W pliku tym znajduje się również wywołanie pluginu *libgazebo_ros_diff_drive.so*, który: nasłuchuje poleceń ruchu publikowanych w topicu *cmd_vel* oraz generuje topic *odom*, opisujący odczytywaną pozycję i orientację symulowanego robota – odpowiada wiec za pełną symulację zachowania platformy kołowej.

   a) Uruchom plik launch: *robot_move.launch*. Plik ten uruchamia węzeł *teleop_twist_keyboard* z pakietu *teleop_twist_keyboard* modyfikując domyślne wartości parametrów określających prędkości robota. Spróbuj poruszać robotem używając klawiszy opisanych w sekcji ***Konfiguracja***.

   b) Zwróć uwagę czy robot porusza się w świecie ***Gazebo*** oraz w jaki sposób reaguje na przeszkody.

   c) Zwróć uwagę czy robot porusza się w programie ***RViz***. W ***RViz*** włącz za pomocą pola ***Frames*** (w wizualizacji ***TF***) pokazywanie dwóch układów współrzędnych: *base_footprint* i *odom* (układ *odom* jest publikowany przez ***Gazebo*** równocześnie z topiciem o tej samej nazwie, w przypadku rzeczywistego robota jego pozycja i orientacja w przestrzeni jest szacowana na podstawie odometrii i wyrażana względem nieruchomego układu *odom*, pozycja robota względem tego układu zmienia się w trakcie ruchu). Ponownie wykonaj ruch robotem i zaobserwuj zachowanie układów *base_footprint* i *odom*. Uzupełnij sprawozdanie zaznaczając czy zaobserwowano ruch układów i samego robota.

   d) Ustaw w polu ***Global Options*** → ***Fixed Frame*** w programie ***RViz*** układ *odom*. Następnie ponownie wykonaj ruch robotem i zaobserwuj jego zachowanie w ***RViz*** (wprowadzona modyfikacja powoduje przyjęcie układu *odom* jako układu nieruchomego, robot wykonując ruchy porusza się względem tego układu i w efekcie porusza się również w programie ***Rviz***). Na koniec wyłącz pokazywanie układów współrzędnych i zapisz zmodyfikowaną wersję konfiguracji programu (opcja opcję ***File*** → ***Save Config***).

   e) Zamknij programy ***RViz*** i ***Gazebo*** oraz przerwij działanie pliku *robot_sim1.launch*, który został uruchomiony na początku tego zadania.

4. Uruchom plik launch: *robot_sim2.launch*. Plik ten wywołuje plik uruchomieniowy z poprzedniego zadania ze zmienionym plikiem URDF zawierającym model robota (*my_robot_gazebo2.urdf*). W pliku URDF dodane zostało wywołanie pluginu *libgazebo_ros_gpu_laser.so*, który publikuje topic *scan*, zawierający dane z pomiarów z lasera LIDAR, czyli informacje o odległościach robota od otaczających przeszkód. Dla ułatwienia w parametrach pluginu włączony został parametr wymuszający pokazywanie promieni lasera w środowisku Gazebo.

   a) W programie ***RViz*** dodaj do listy wizualizacji element typu ***LaserScan*** (przycisk ***Add***, następnie wybór wizualizacji ***LaserScan***). Ustaw pole ***topic*** w dodanym elemencie na nadawany przez ***Gazebo*** topic *scan*.

   b) Porównaj stan środowiska ***Gazebo*** z odczytami lidara wyświetlanymi w programie ***RViz*** podczas ruchu robota. Zwróć uwagę na zgodność wykrytych przeszkód i ich odległości w obu wizualizacjach oraz na to, jak dokładnie dane z lidara odzwierciedlają rzeczywiste rozmieszczenie obiektów w symulowanym świecie.

   c) Zrób zrzut ekranu pokazujący stan środowiska ***Gazebo*** i wizualizację w programie ***RViz***, a następnie dołącz go do sprawozdania.

   d) Nie zamykaj żadnego z programów – będą wykorzystywane w zadaniu następnym.

5. Uruchom węzeł *slam_gmapping* z pakietu *gmapping*. Węzeł ten realizuje lokalizację i jednoczesne tworzenie mapy (SLAM, Simultaneous Localization and Mapping) na podstawie danych z czujników, takich jak lidar, przetwarzając je w czasie rzeczywistym w celu generowania mapy otoczenia oraz śledzenia pozycji robota względem tej mapy.

   Uwaga! Domyślnie *slam_gmapping* zakłada, że robot ma układ bazowy o nazwie *base_link*. Jeśli w zdefiniowanym modelu robota taki układ nie występuje, to podczas uruchamiania węzła należy ustawić parametr *base_frame*, wskazując nazwę istniejącego układu, np. *base_footprint*:

   ```
   rosrun gmapping slam_gmapping _base_frame:=base_footprint
   ```

   a) Porusz robotem, żeby węzeł *slam_gmapping* mógł zarejestrować ruch i opublikować topic zawierający mapę (domyślnie topic o nazwie *map*).

   b) W programie ***RViz*** dodaj do listy wizualizacji element typu ***Map*** (przycisk ***Add***, następnie wybór wizualizacji ***Map***). Ustaw pole ***topic*** w dodanym elemencie na nadawany przez *slam_gmapping* topic *map*.

   c) Poruszaj robotem w taki sposób, że otrzymać pełną mapę pomieszczenia. Udokumentuj realizację zadania dołączając do sprawozdania zrzut ekranu pokazujący mapę pomieszczenia, w którym znajduje s się robot.