Informacje ogólne

- Zadania przedstawione w niniejszej instrukcji należy wykonać **samodzielnie** na systemie operacyjnym Ubuntu 18.04.2 (x64). Hasło użytkownika to `tello`.
- Wykonanie każdego zadania należy **udokumentować zrzutami ekranu obejmującymi cały ekran monitora.** Niezastosowanie się do powyższego będzie równoznaczne z pominięciem zadania przy ocenie.
- Po zakończeniu pracy należy usunąć pliki własne.

Zagadnienia

- środowisko ROS 2 Eloquent Elusor (ang. *Robot Operating System*),
- narzędzie symulacyjne Gazebo 9,
- symulator tello_ros,
- komunikacja z jednostką latającą z poziomu terminala systemowego.

Materiały pomocnicze

- https://docs.ros.org/en/eloquent/
- https://docs.python.org/3.6/
- http://gazebosim.org/tutorials
- https://github.com/clydemcqueen/tello_ros
- https://www.jetbrains.com/pycharm/download/#section=linux

Wprowadzenie

Symulator Gazebo

Gazebo to symulator zbudowany w oparciu o dedykowany, wysokowydajny silnik fizyczny ODE (*Open Dynamics Engine*) oraz bibliotekę OpenGL, służącą do generowania obrazu w czasie rzeczywistym. Może on zostać zainstalowany jako część środowiska ROS, pozwalającą na fizyczną symulację obiektów z zachowaniem wysokiego stopnia realizmu. Niewątpliwą zaletą Gazebo jest pełna symulacja otoczenia (w tym oświetlenia, cieni, tekstur), z którym może zachodzić interakcja, zarówno na poziomie fizycznym (kolizje) jak i na poziomie wizualnym (modele sensorów realizujących rzeczywiste zadania). Co istotne, Gazebo może zostać uruchomione samodzielnie, poza ekosystemem ROS.

Narzędzie może zostać uruchomione za pośrednictwem menu systemowego lub z poziomu terminala (polecenie `gazebo`). Uruchomienie bez żadnych dodatkowych parametrów poskutkuje wczytaniem domyślnego, pustego świata. Twórcy oddali do dyspozycji użytkowników przykładowe światy wraz z modelami obiektów, które ułatwiają zapoznanie się ze środowiskiem. Są one dostępne w lokalizacji[1] `/opt/ros/eloquent/share/gazebo_plugins/worlds/`.

Wczytanie przykładowego świata może odbyć się przy pomocy polecenia wywołanego z poziomu terminala systemowego:

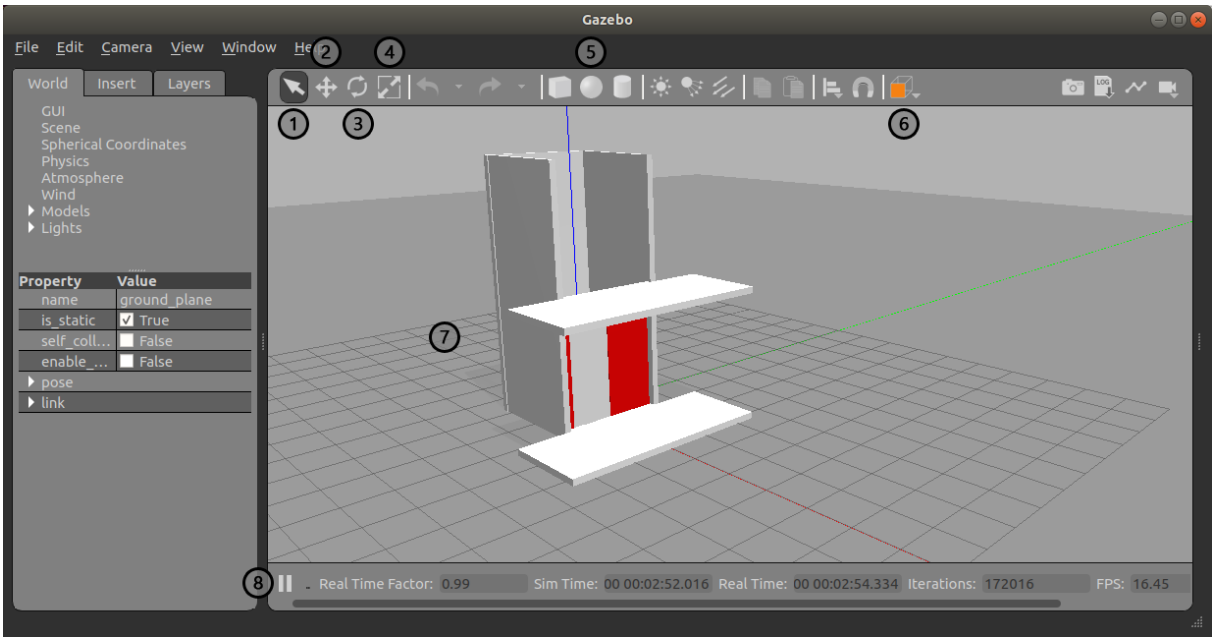
```
gazebo
/opt/ros/eloquent/share/gazebo_plugins/worlds/gazebo_ros_elevator_demo.world
```

[1] Ścieżka ta jest poprawna tylko w przypadku użycia Gazebo dołączonego do dystrybucji ROS Eloquent Elusor.

Na ekranie powinno wyświetlić się okno widoczne na rysunku 1. Interfejs narzędzia składa się z kilku części:

- belki menu głównego,
- panelu pozwalającego na zarządzanie światem i jego elementami składowymi,
- obszaru roboczego (7) wraz z belką narzędzi (u góry) oraz belką informacyjną (na dole) (8).

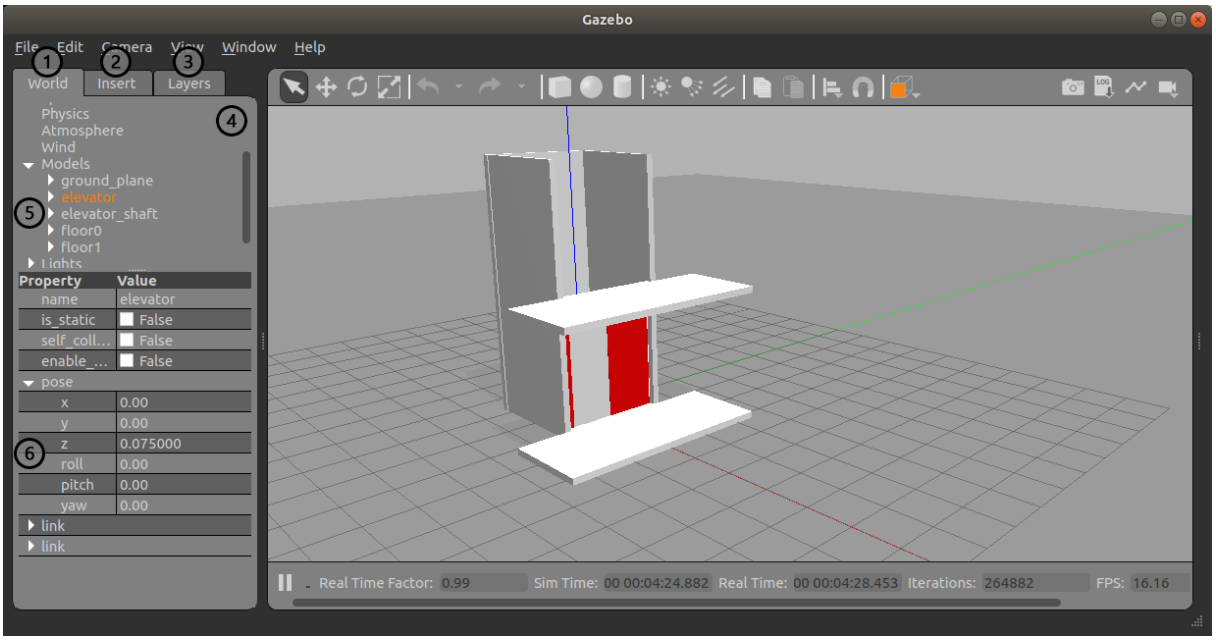
Ikony znajdujące się na belce narzędzi pozwalają kolejno na interakcję z otoczeniem (1), przenoszenie elementów otoczenia (2), rotację elementów otoczenia (3) oraz na zmianę ich rozmiaru (4). Możliwe jest także tworzenie prostych obiektów, takich jak prostopadłościany, kule, czy walce (5). Bardzo przydatną funkcją jest także możliwość wyboru pozycji kamery spośród predefiniowanej ich listy (6). Przydaje się to m.in. podczas obserwacji symulowanego obiektu wykonującego ruch tylko w jednej lub dwóch płaszczyznach.



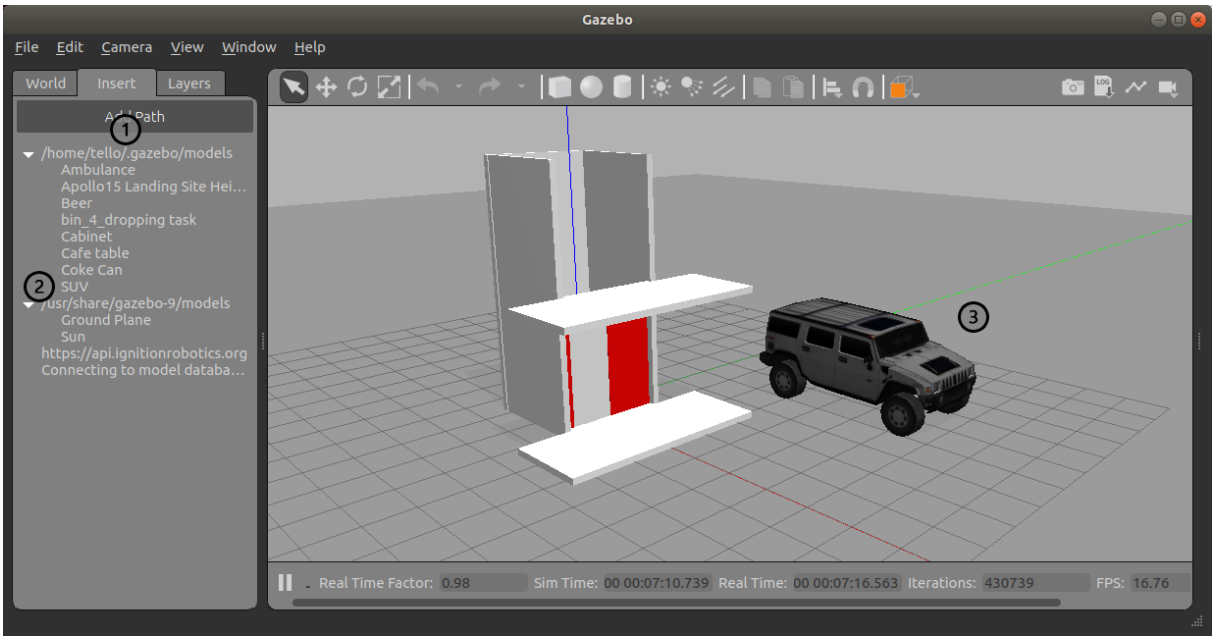
Rysunek 1 – struktura okna głównego narzędzia Gazebo (obszar roboczy)

Zarządzanie światem oraz jego elementami składowymi odbywa się z poziomu panelu umiejscowionego po lewej stronie (rysunek 2). Widoczne są tam trzy zakładki: zarządzanie całym światem (1), dołączanie predefiniowanych modeli (2) oraz warstwy świata (3). Pierwsza zakładka zawiera listę wszystkich dodanych obiektów (4). Jeśli dany obiekt składa się z wielu składowych, są one widoczne w formie drzewka (np. *elewator*). Całość została podzielona na kilka grup obiektów, w tym wiatr, modele (5) i oświetlenie. Wybierając dowolny element z listy, możliwa jest konfiguracja jego parametrów, w tym określenie jego translacji oraz rotacji (6).
Druga zakładka pozwala na dodanie do świata predefiniowanych obiektów (rysunek 3). Ich lista jest synchronizowana z serwerem dostawcy oprogramowania. Standardowo widoczne są dwie sekcje – modele lokalne (1) oraz modele zdalne (2), dostępne po stronie zewnętrznego serwera. Klikając na dowolny element, możliwe jest jego przeniesienie i umieszczenie w symulowanym świecie.

Ostatnia zakłada – warstwy – pozwala na zarządzanie warstwami zdefiniowanymi w ramach danego świata. Są one tworzone w pliku SDF[2] (*Simulation Description Format*).



Rysunek 2 – struktura okna głównego narzędzia Gazebo (zakładka World)



Rysunek 3 – struktura okna głównego narzędzia Gazebo (zakładka Insert)

Interakcja z symulowanym światem jest możliwa z poziomu konsoli. Można tego dokonać na dwa sposoby – za pośrednictwem komend Gazebo lub za pośrednictwem komend ROS'a. Jeśli środowisko ROS jest zainstalowane, Gazebo uruchamia przy starcie węzeł główny ROS'a (tzw. *ROS Master*). Jest to istotne z punktu widzenia publikacji *topic'ków*, które są dostępne w przestrzeni Gazebo oraz

[2] http://sdformat.org/

ROS'a (o ile zainstalowany). Ich lista jest mocno zróżnicowana i może zostać wyświetlona odpowiednio przy użyciu polecenia

```
gz topic -l
```

dla przypadku Gazebo oraz

```
ros2 topic list
```

dla przypadku ROS'a. Należy jednocześnie nadmienić, że w przypadku Gazebo niemożliwa może okazać się kompleksowa symulacja obiektu z poziomu terminala. W takim przypadku należy przygotować odpowiedni kod klienta w języku C++.

Zasada działania omawianego przykładu (modelu windy) została dokładnie opisana na stronie GazeboSim[3]. Aby wymusić jej ruch należy wysłać żądanie, podając jako argument numer piętra (0 lub 1). Polecenie dla piętra 1 będzie wyglądało następująco:

```
ros2 topic pub /demo/elewator_demo std_msg msg/String data:\ \'1\ '\
```

Z poziomu samego terminala i polecenia gz, możliwe jest tylko pełne sterowanie drzwiami windy

```
gz joint –m elewator –j door –f 2
```

gdzie `-m` oznacza nazwę modelu, `-j` nazwę połączenia, a `-f` siłę. Winda ponadto powinna także w interakcję z elementami świata Gazebo (patrz: dokumentacja przykładu).

Bibliotek tello_ros

W sieci dostępnych jest wiele dedykowanych modeli, które wiernie odzwierciedlają jednostki fizyczne. Przykładem tego może być biblioteka `tello_ros`, która pozwala na symulację oraz sterowanie rzeczywistą jednostką drona DJI Ryze Tello[4]. Istotną cechą tej biblioteki jest to, że możliwa jest symulacja jednostki latającej w narzędziu Gazebo oraz – po ukończeniu testów – przeniesienie zadań na jednostkę fizyczną przy niewielkim nakładzie pracy. Na potrzeby zajęć przygotowane zostały skrypty, które w sposób znaczący ułatwiają pracę z symulatorem. Znajdują się one na pulpicie i możliwe jest ich uruchomienie za pomocą dwukrotnego kliknięcia.

Do udostępnionych materiałów zaliczyć można:

- `start_tello.sh` – uruchamia symulator `tello_ros` w narzędziu Gazebo,
- `tello_cam.sh` – uruchamia narzędzie `rqt_image_view`, pozwalające na podgląd obrazu z kamery drona (w czasie rzeczywistym),
- `tello_takeoff.sh` – startuje jednostką latającą,
- `tello_land.sh` – ląduje jednostką latającą,
- `tello_rotate.sh` – wykonuje rotację wokół osi Z.

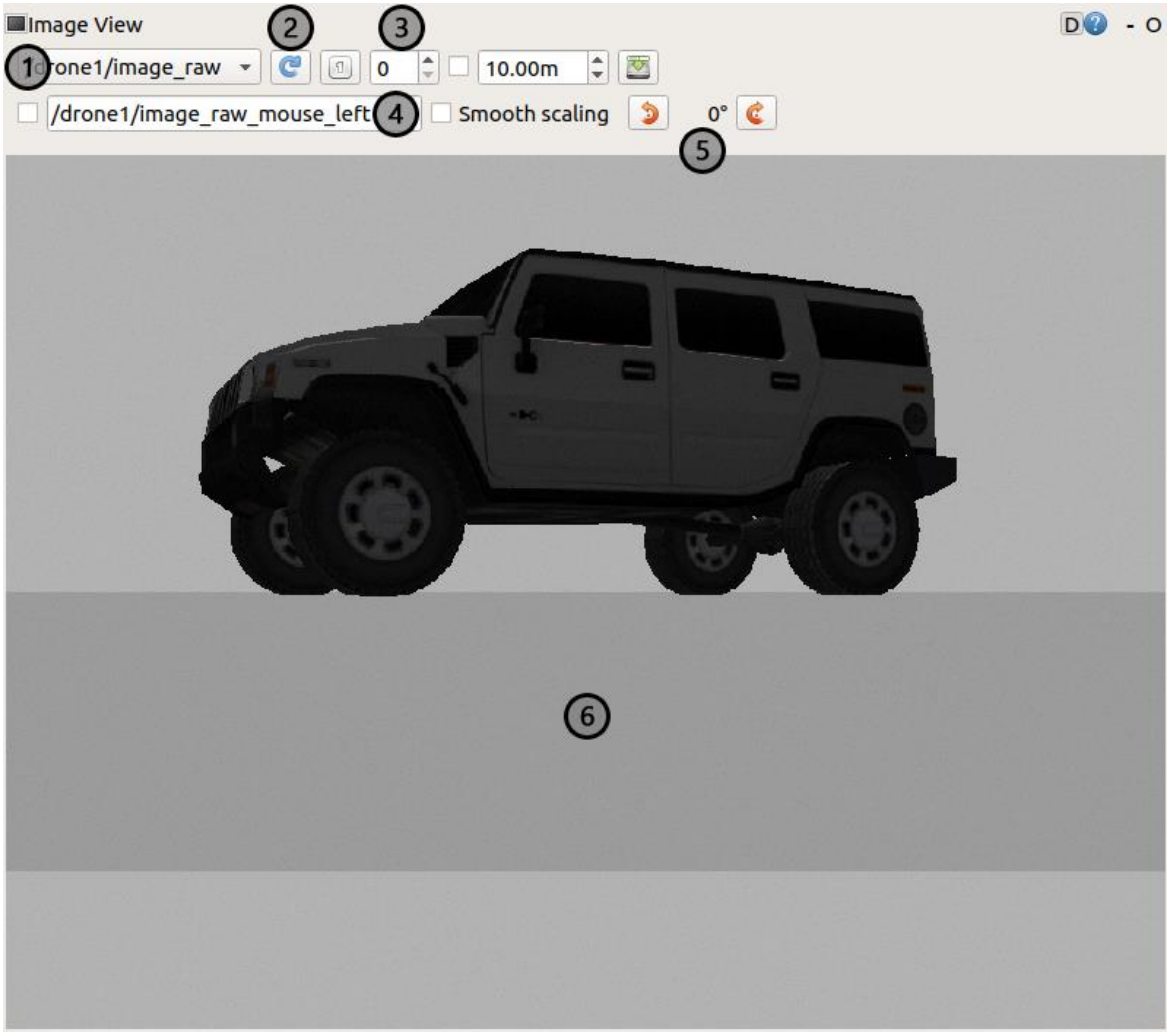
Wszystkie skrypty (oprócz `start_tello.sh`) pozwalają także na interakcję z rzeczywistym dronem. Nadmienić warto, że dodatkowo dołączony został folder *readme*, zawierający materiały z repozytorium

[3] http://gazebosim.org/tutorials?tut=elevators
[4] https://store.dji.com/pl/product/tello

biblioteki `tello_ros` oraz skrypt `start_g2rr.sh` kopiujący (republikujący) niezbędne *topic'ki* odometryczne z Gazebo do ROS'a.

Na dodatkową uwagę zasługuje narzędzie `rqt_image_view`, które pozwala na uzyskanie podglądu z kamery (6). Posiada ono prosty i intuicyjny interfejs, oferujący wszystkie niezbędne opcje. Wśród nich wyróżnić można:

1) listę rozwijalną, zawierającą spis wszystkich (aktywnych) *topic'ów* strumieniujących obraz,
2) przycisk pozwalający na odświeżenie listy (1),
3) kontrolkę umożliwiającą naniesienie linii pomocniczych (siatki) na oryginalny obraz,
4) kontrolkę pozwalającą na opublikowanie *topic'a* ze zmodyfikowanym obrazem,
5) przyciski umożliwiające dokonanie rotacji obrazu (co 90°).



Rysunek 4 – okno narzędzia rqt_image_view

Dostęp do pełnego zbioru narzędzi udostępnianego w ramach pakietu `rqt`[5] można uzyskać przy pomocy komendy:

```
rqt
```

[5] http://wiki.ros.org/rqt

Zadania szczegółowe

1. Za pomocą terminala uruchom narzędzie symulacyjne Gazebo. Jako argument wskaż pełną ścieżkę do pliku `gazebo_ros_tricycle_drive_demo.world`[6].
2. Zapoznaj się interfejsem narzędzia oraz ze sposobami przemieszczania kamery (przesunięcie, obrót, przybliżenie).
3. Korzystając z belki górnej, dodaj do świata obiekt Box i umieść na nim pojazd. Box powinien znajdować się w lokalizacji `0, 0, 0` (x, y, z) przyjmując wartości rotacji odpowiednio `0, 0, 0` (roll, pitch, yaw).
4. Korzystając z terminala, sprawdź listę dostępnych *Topic'ów*. Znajdź *topic* `/demo/cmd_demo` i opublikuj dane pozwalające na poruszanie się pojazdu względem osi X (wartość: `0.2`).
5. Zatrzymaj pojazd i korzystając z Gazebo, ustaw go w pozycji domyślnej (*Reset Model Poses*). Gdzie pojawiły się obiekty?
6. Zasubskrybuj *topic* `/demo/odom_demo` i przeanalizuj pozyskane dane. Szczegółowo omów strukturę ramki danych.
7. Ustaw pojazd na pozycji `0, 0, 0` przy wartościach rotacji `0, 0, 0`. Ustaw Box na pozycji `5, 0, 0`. Opublikuj dane pozwalające na poruszanie się pojazdu względem osi X (wartość `1.0`). Co udało się zaobserwować?
8. Zamknij narzędzie Gazebo.
9. Uruchom węzeł `g2rr` klikając dwukrotnie na skrypt znajdujący się na pulpicie (`start_g2rr.sh`)
10. Uruchom narzędzie `tello_ros` (`start_tello.sh`).
11. Zapoznaj się z dokumentacją omawianego narzędzia (folder `readme`).
12. Uruchom skrypt umożliwiający podgląd obrazu z kamery głównej drona (`tello_cam.sh`). Zapoznaj się z narzędziem `rqt_image_view`.
13. Umieść w świecie (Gazebo) obiekt, który będzie widoczny przez kamerę drona.
14. Wystartuj drona (`tello_takeoff.sh`) obserwując jednocześnie podgląd kamery.
15. Wymuś obrót drona względem osi Yaw (`tello_rotate.sh`).
16. Zasubskrybuj *topic* `/republisher/tello_1/odom` i przeanalizuj pozyskane dane.
17. Korzystając z terminala, zatrzymaj rotację drona. Następnie wyląduj dronem (`tello_land.sh`).
18. W nowym terminalu zasubskrybuj *topic* `/drone1/tello_response`[7].
19. Wykonaj ponownie zadania z punktów 13-17 obserwując jednocześnie zawartość terminala z punktu 18. Co udało się zaobserwować? W jaki sposób dron informuje o wykonaniu polecenia?

[6] /opt/ros/eloquent/share/gazebo_plugins/worlds
[7] Pamiętaj, aby wykonać polecenie `source` odnoszące się do przestrzeni roboczej `tello_ros_ws`.