ROS - zad 1

Celem tego zadania będzie napisanie programu sterującego robotem, wyposażonym w LIDAR, tak aby przejeżdżał on po łuku trzymając się możliwie blisko jego środka. Aby pobrać pliki niezbędne do wykonania tego zadania trzeba najpierw przejść w terminalu do katalogu src w katalogu roboczym rosa. Przykładowo komenda może wyglądać następująco:

```
cd ros_ws/src
```

A następnie pobrać repozytorium wpisując polecenie:

```
git clone https://bitbucket.org/duszakpio/wseiz_ros_1.git
```

1. Uruchomienie symulacji.

Zadanie będzie realizowane na symulacji robota (aczkolwiek jeśli program zadziała na symulacji to z dużą dozą prawdopodobieństwa można oczekiwać, że na prawdziwym robocie też by zadziałał).

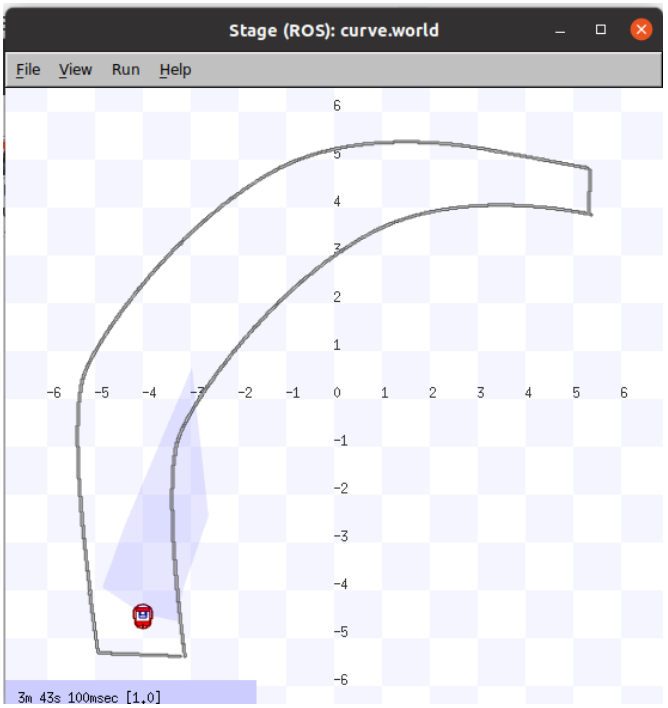
Żeby ją uruchomić należy przejść do katalogu …/wseiz_ros_1/world i wpisać polecenie

```
rosrun stage_ros stageros curve.world
```

Ale wcześniej trzeba uruchomić rdzeń rosa poleceniem

```
roscore
```

Uruchomi to symulator świata składającego się z zakrzywionego korytarza oraz robota na jednym jego końcu.



Można teraz wyświetlić listę topiców poleceniem

```
rostopic list
```

Dwa topiki, które będą dla nas ważne podczas tego ćwiczenia to /base_scan, na którym będą publikowane odczyty z lidaru oraz /cmd_vel, na którym będziemy publikować z jaką prędkością ma jechać robot.

Aby wyświetlić informacje z lidaru należy wpisać polecenie:

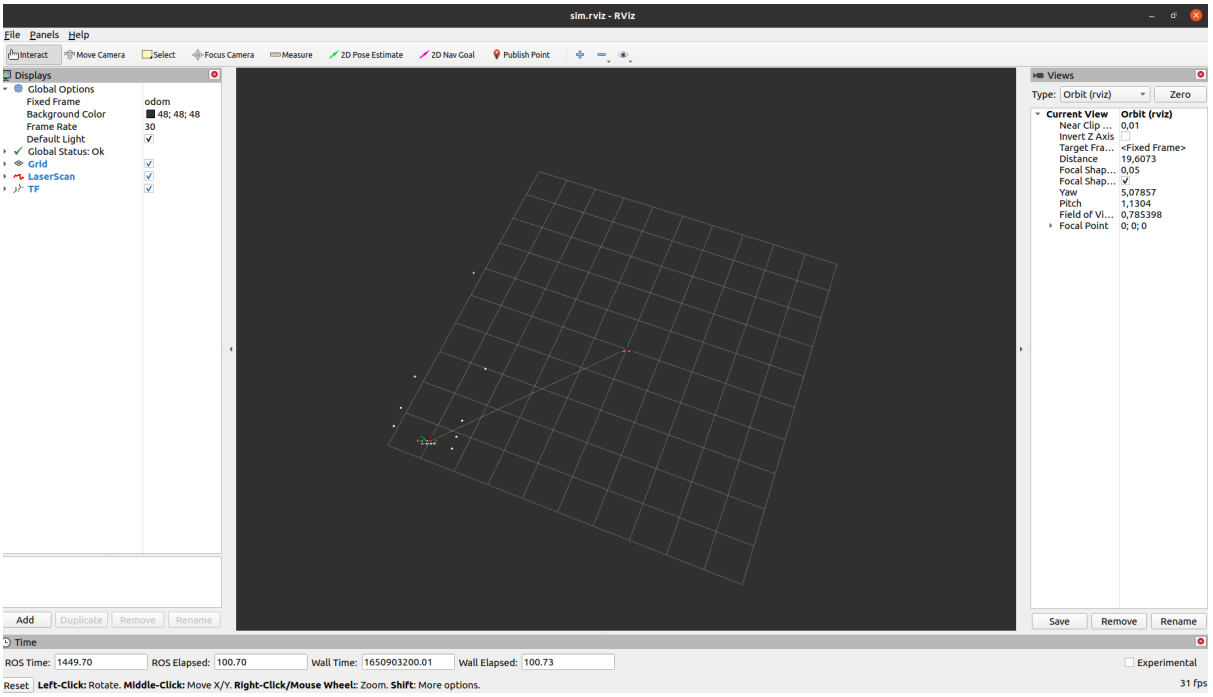
```
rostopic echo /base_scan
```

Jak widać informacje składają się z nagłówka, minimalnego kąta widzenia, maksymalnego kąta widzenia, przyrostu kąta pomiędzy kolejnymi punktami, przyrostu czasu pomiędzy kolejnymi punktami, czasu skanowania, zasięgu (minimalnego i maksymalnego) oraz z kolejnych odczytów (w naszym wypadku jest to 6 odległości).

Aczkolwiek nie jest to najwygodniejszy sposób oglądania danych z LIDARu, w związku z tym skorzystamy z programu do wizualizacji jakim jest RVIZ. Żeby go uruchomić należy wpisać

```
rosrun rviz rviz
```

Uruchomi się on z domyślną konfiguracją. Żeby otworzyć konfigurację potrzebą dla tego zadania trzeba kliknąć file->open config a następnie znaleźć plik sim.rviz znajdujący się w katalogu wseiz_ros_1. Teraz możemy o wiele wygodniej zobaczyć odczyty z sensora, a także obserwować gdzie względem globalnego układu odniesienia znajduje się robot.



Chwytając kursorem myszy robota na symulacji możemy go ręcznie przesuwać i zaobserwować zmieniające się odczyty.

Żeby zobaczyć jakiego typu informacje mają być publikowane na topicu /cmd_vel należy wpisać polecenie

```
rostopic info /cmd_vel
```

Jedną z informacji jakie uzyskamy jest typ informacji jaki powinien być publikowany, czyli geometry_msgs/Twist. Żeby dowiedzieć się jak wygląda wewnętrzna struktura takiego typu trzeba wpisać komendę:

```
rosmsg show geometry_msgs/Twist
```

2. Algorytm sterowania.

W trakcie ćwiczenia będziemy programować robota behawioralnego, to znaczy, że jego działanie będzie wyznaczane bezpośrednio na podstawie obecnych odczytów z sensorów. Możliwe są więc dwa zachowania robota:

- Zatrzymanie się, gdy bezpośrednio przed nim jest ściana
- Jeśli bezpośrednio przed nim nie ma ściany jazda równolegle do ścian

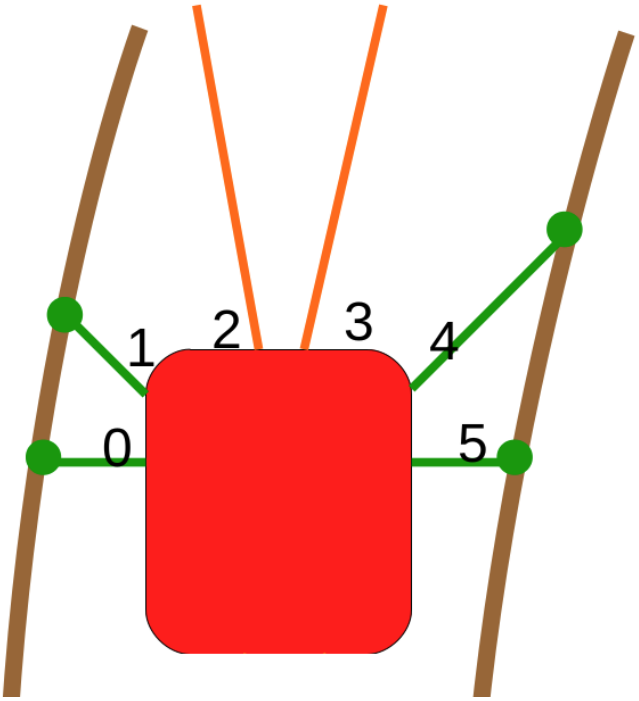
Żeby zrealizować to zadanie potrzebne będą dwie funkcje sensoryczne:

- funkcja sprawdzająca, czy przed robotem jest ściana
- funkcja zamieniająca odczyty lidaru na kierunek ruchu robota

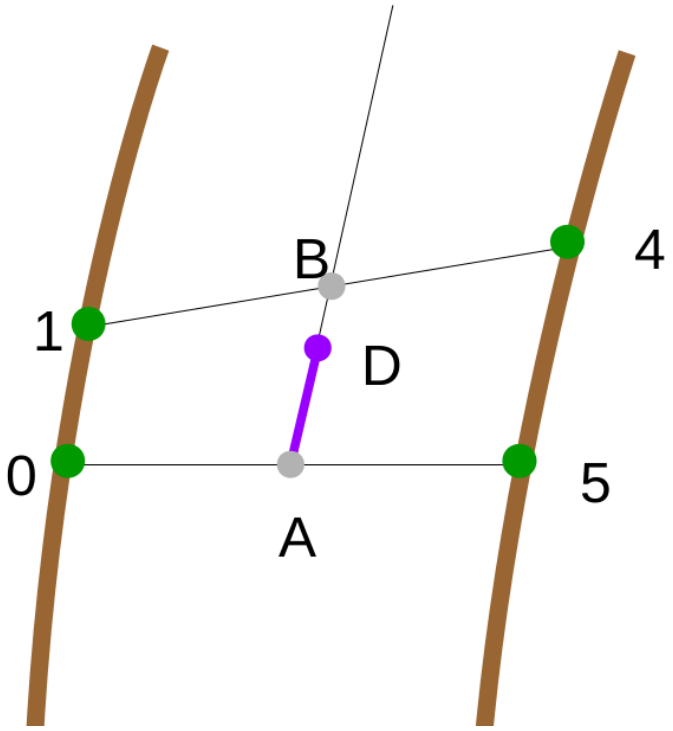
Trzy funkcje sterowania:

- funkcja zatrzymująca robota,
- funkcja przeliczająca kierunek ruchu na sterowanie robota (przy pomocy regulatora PID)
- funkcja wysyłająca sterowanie obliczone przez regulator do silników robota

Dodatkowo funkcja pomocnicza przeliczająca współrzędne biegunowe na kartezjańskie. Robot jest wyposażony w LIDAR dający 6 punktów (w układzie biegunowym) rozmieszczonych w sposób pokazany na poniższym obrazku.



a) Funkcja sprawdzająca, czy przed robotem jest ściana (wykorzystuje odczyty nr 2 i 3) - sprawdza czy robot znajduje się blisko ściany z przodu, jeśli tak zwraca True, w przeciwnym wypadku zwraca False.
b) Funkcja zamieniająca odczyty lidaru na kierunek ruchu robota (wykorzystuje odczyty nr 0, 1, 4 i 5) - wyznacza kierunek w którym robot ma się poruszać, w następujący sposób (patrz rysunek poniżej):
   i) Wyznacza punkt A, leżący w połowie między punktem 0 i 5
   ii) Wyznacza punkt B, leżący w połowie między punktem 1 i 4
   iii) Wyznacza punkt D, leżący na prostej AB w stałej odległości od punktu A (lub tak, żeby stosunek AD/DB był stały)
   iv) Zwraca punkt D



c) Funkcja zatrzymująca robota - wysyła zerowe prędkości liniowe i kątowe do robota
d) Funkcja przeliczająca kierunek ruchu na sterowanie robota (przy pomocy regulatora PID) - realizuje dyskretny regulator PID, który jako wejście (odchyłkę) bierze współrzędną y punktu D, a na wyjściu składową obrotowej prędkości robota
e) Funkcja wysyłająca sterowanie obliczone przez regulator do silników robota - wysyła do robota prędkość liniową (ustawioną jako stała) i kątową (wyznaczoną przy pomocy regulatora PID)

3. Program.

Aby uruchomić IDE i kod programu należy przejść do katologu …/wseiz_ros_1/scripts i uruchomić pycharma poleceniem:

```
pycharm-community wseiz_ros_1.py
```

Uruchomi to IDE od razu z otwartym plikiem wseiz_ros_1.py. Podczas pierwszego uruchamiania zostaniemy zapytani o dwie rzeczy. Czy ufamy temu projektowi

(potwierdzamy) oraz czy otworzyć w LightEdit, czy jako projekt (wybieramy jako projekt). Następnie należy dodać konfigurację. Prowadzący poda jak to zrobić podczas zajęć.

Mamy teraz dwa sposoby aby uruchomić program. Kliknąć zielony trójkąt w PyCharmie lub “zbudować” program poleceniem

```
catkin_make
```

W katologu roboczym ROSa, a następnie wpisując polecenie:

```
rosrun wseiz_ros_1 wseiz_ros_1.py
```