

**Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska**

Sterowanie i symulacja robotów

**Sprawozdanie z projektu i laboratorium nr 1
Blok mobilny**

Kaniuka Jan, Krasnodębski Przemysław

Warszawa, 2021

Spis treści

1. Wstęp	2
2. Część laboratoryjna	3
2.0.1. Zapoznanie ze środowiskiem	3
2.0.2. Sterowanie prędkościowe	3
2.0.3. Omówienie implementacji	4
2.0.4. Test kwadratu z wizualizacją w Rviz	4
3. Część projektowa	5
3.0.1. Implementacja względnego sterowania pozycyjnego	5
3.0.2. Porównanie dokładności osiągania pozycji z wykorzystaniem pozycji referencyjnej	6
3.0.3. Praca z rzeczywistym robotem	9
3.0.4. rosbag	9
3.0.5. rqtplot	9
3.0.6. Analiza danych z pliku <i>.bag</i>	10
3.0.7. Struktura komunikacji	15
4. Podsumowanie	16

1. Wstęp

Sprawozdanie zostało przygotowane w ramach zajęć z przedmiotu *Sterowanie i symulacja robotów* w semestrze zimowym roku 2021. Część laboratoryjna została zrealizowana podczas zajęć w dniu 19.10.2021. Główny przedmiot rozważań podczas projektu oraz laboratorium stanowiła analiza dokładności przemieszczania się robota mobilnego podczas względnego sterowania pozycyjnego. W pierwszym przypadku robot poruszał się na podstawie wyłącznie zadawanych prędkości, a w drugim wariancie wykorzystywał dane z odometrii. Programy zaimplementowane podczas prac nad projektem zostały również uruchomione na rzeczywistym robocie *Tiago* firmy PAL w Laboratorium Robotyki na Wydziale EiTI.



Rys. 1.1: Robot *Tiago* (*Rico*)

2. Część laboratoryjna

2.0.1. Zapoznanie ze środowiskiem

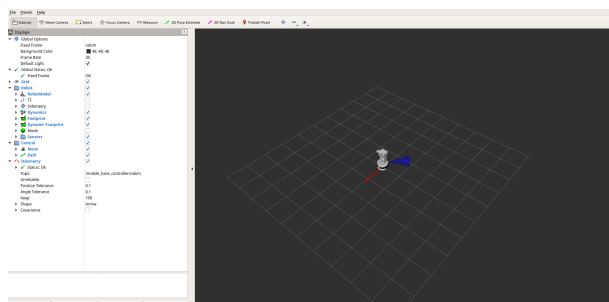
Pracę w laboratorium rozpoczęliśmy od odpowiedniego skonfigurowania stanowiska do pracy. Stworzyliśmy przestrzeń roboczą i podłączyliśmy się do pakietów robota *Tiago*. Następnie przeszliśmy do folderu źródeł i pobraliśmy stosowne repozytorium. W pierwszej kolejności uruchomiliśmy programy Gazebo oraz RViz w celu wizualizacji robota w pustym świecie (symulacji).

2.0.2. Sterowanie prędkościowe

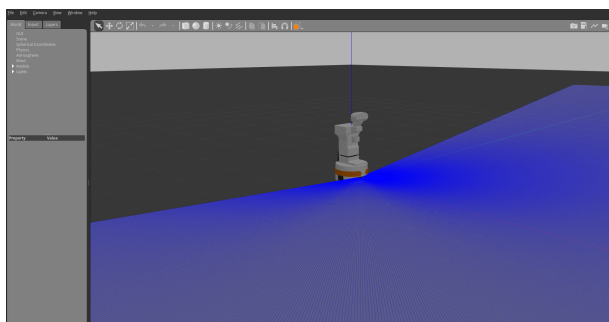
W naszym przypadku sterowanie to polega na inicjalizacji ruchu w punkcie $(0,0,0)$ $[x,y,theta]$. Kolejne sterowania wyznaczone są na podstawie zrealizowanej (idealnie) poprzedniej akcji ruchu. Zakładamy więc, że znając prędkość i zadając konkretny czas ruchu, robot dojedzie do pożądanej pozycji. Testy mieliśmy wykonać dla trzech, różnych prędkości o następujących wartościach podanych przez Prowadzącego:

	linear	angular
low	0.04 m/s	0.11 rad/s
medium	0.096 m/s	0.264 rad/s
high	0.23 m/s	0.6336 rad/s

(testy sprawdzające poprawność działania programu wykonywaliśmy względem danych z tematu `/gazebo/model_states/pose[1]/position`)



Rys. 2.1: Widok z wizualizatora *RViz*



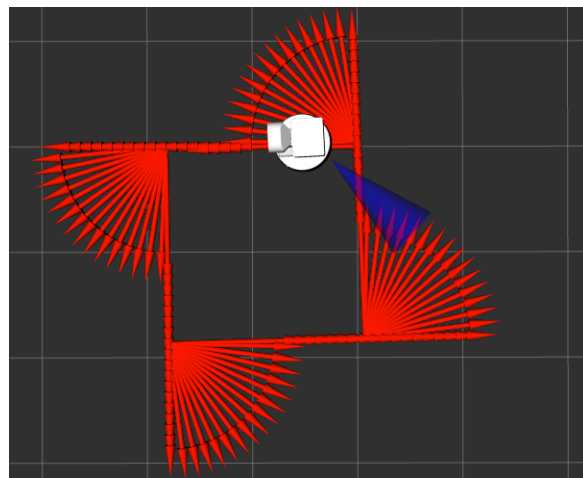
Rys. 2.2: Widok z symulatora *Gazebo*

2.0.3. Omówienie implementacji

Parametrami w naszym programie są: długość boku kwadratu oraz prędkość liniowa i obrotowa. Prędkość zadana jest stała, więc przekształcając równanie dla ruchu jednostajnego możemy wyliczyć czas. Zadajemy ruch liniowy przez wyznaczony okres czasu, a następnie ruch obrotowy o kąt 90 stopni. Zakładamy (możliwe, że naiwnie), że robot wykonał powierzone mu zadanie tzn. przebył dystans w odpowiednim czasie.

2.0.4. Test kwadratu z wizualizacją w Rviz

Podczas zajęć udało nam się sprawdzić poprawność działania zaimplementowanej struktury sterowania. Poniżej załączamy obraz ze ścieżką wykonaną przez robota podczas sterowania prędkościowego przy małej prędkości.

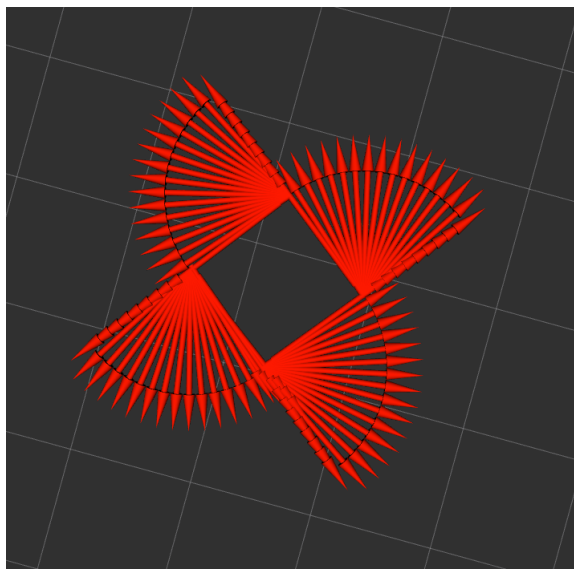


Rys. 2.3: Sterowanie prędkościowe (prędkość *mała*)

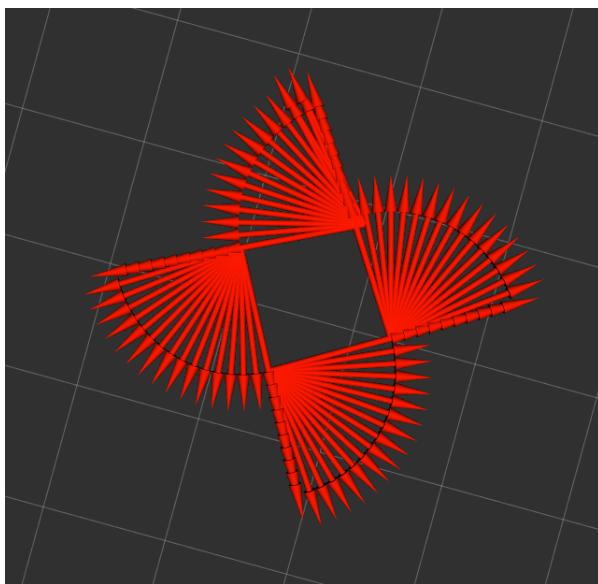
3. Część projektowa

3.0.1. Implementacja względnego sterowania pozycyjnego

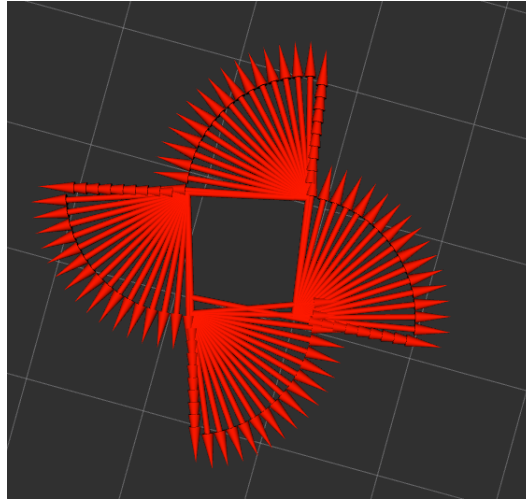
Poniżej zamieszczamy testy kwadratu dla trzech, różnych wartości prędkości: liniowej i kątowej:



Rys. 3.1: Prędkość *mała*



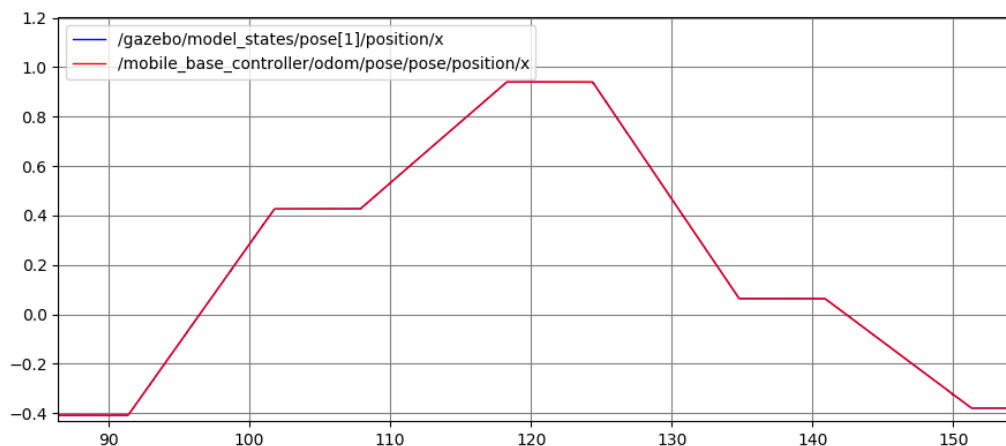
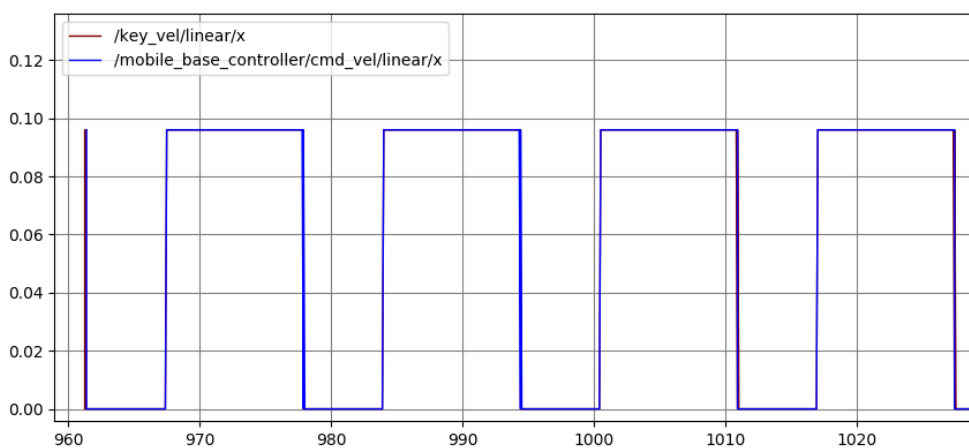
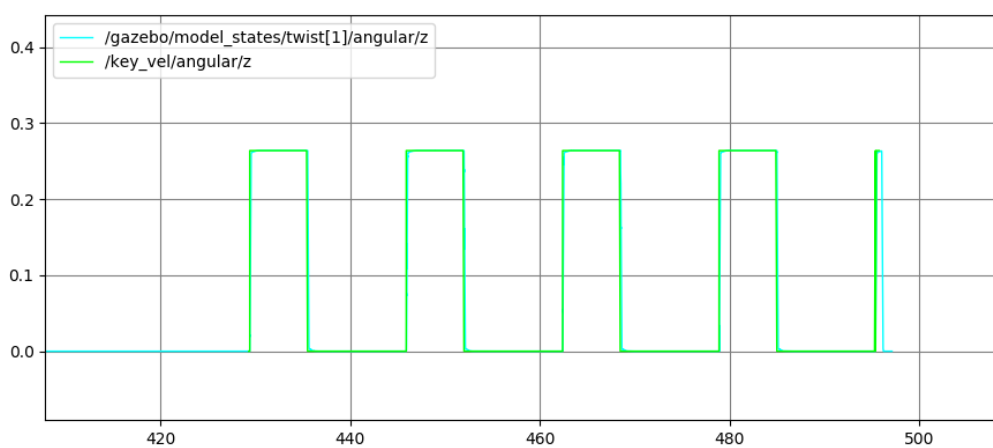
Rys. 3.2: Prędkość *średnia*

Rys. 3.3: Prędkość *duża*

3.0.2. Porównanie dokładności osiągania pozycji z wykorzystaniem pozycji referencyjnej

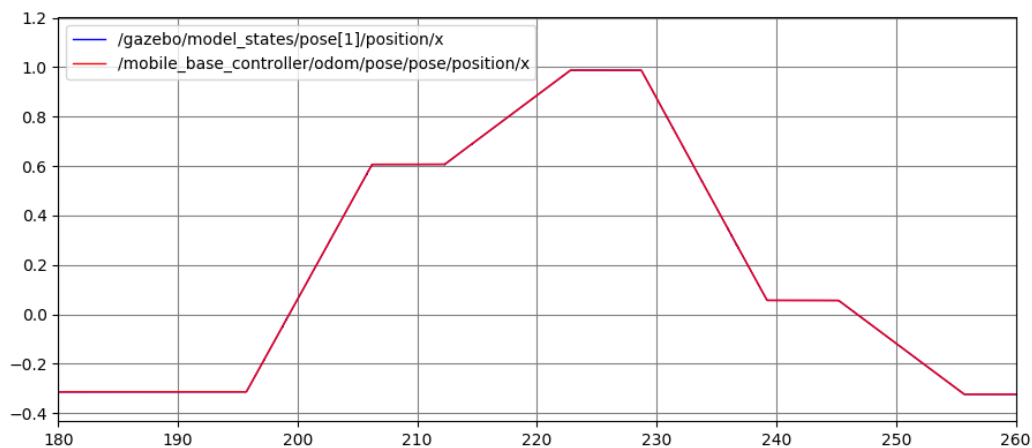
Dokonaliśmy analizy działania wewnętrznych regulatorów robota podczas symulacji w reakcji na zadawane komendy ruchu. Przebiegi wartości zadanej i osiąganych parametrów ruchu przedstawiono na rysunkach poniżej. Z powodu bardzo podobnych wyników dla różnych prędkości i implementacji przedstawiono tylko niektóre wykresy.

Sterowanie prędkościowe

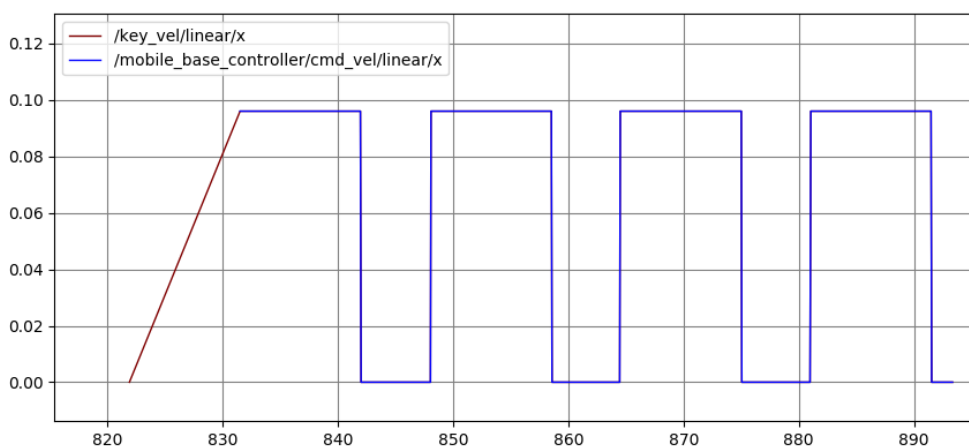
Rys. 3.4: Pozycje liniowe x zadane i wykonaneRys. 3.5: Prędkości liniowe x zadane i wykonane

Rys. 3.6: Prędkości kątowe zadane i wykonane

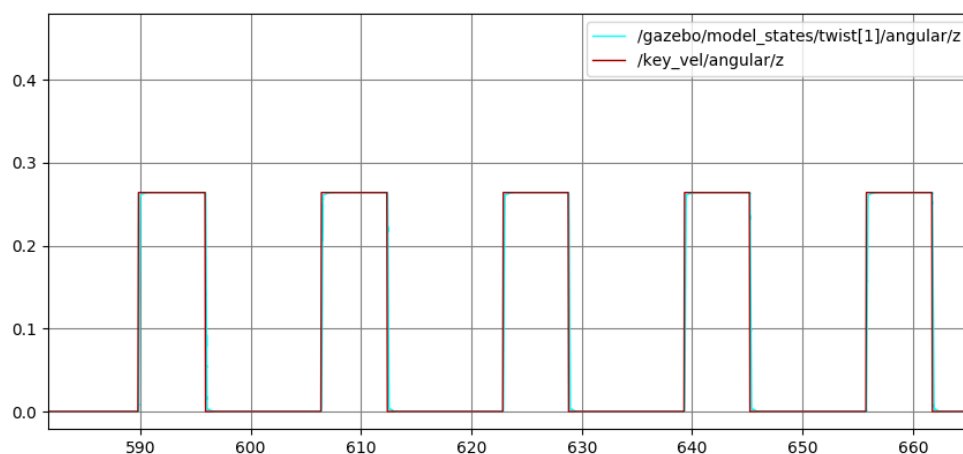
Sterowanie z odometrią



Rys. 3.7: Pozycje liniowe x zadane i wykonane



Rys. 3.8: Prędkości liniowe x zadane i wykonane



Rys. 3.9: Prędkości kątowe zadane i wykonane

Powyższe dane zebrano podczas ruchu robota ze średnią prędkością. Wyraźnie widać, że prędkości oraz pozycje zadane są równe referencyjnym. Małe rozbieżności można zauważyć podczas dużych zmian wartości prędkości. Wynikają one ze skończonej wartości przyspieszenia robota.

3.0.3. Praca z rzeczywistym robotem

Mieliśmy możliwość uruchomienia naszych programów na rzeczywistym robocie *Tiago*. Celem zadania było zebranie danych do analizy porównawczej pomiędzy lokalizacją globalną, a lokalizacją wynikającą z odometrii. Uruchomiliśmy program zadający ruch po kwadracie z wykorzystaniem odometrii dla trzech wartości prędkości.

3.0.4. rosbag

Podczas pracy z robotem zebraliśmy dane, które zostały poddane późniejszej analizie. Przy użyciu narzędzia *rosbag* mogliśmy "nagrać" wiadomości nadawane na wybranych tematach, aby móc później np. zwizualizować dane z pozycji na wykresie.

Zebrane dane pozyskaliśmy z tematów:

- `/tf`
- `/mobile_base_controller/odom`
- `/robot_pose`

Poniżej załączyliśmy dane obrazujące przekształcenia pomiędzy układami `map` i `odom` w momencie startowym (na początku zbierania danych). Różnica w położeniu tych układów współrzędnych zostanie uwzględniona podczas analizy danych opisanej w dalszej części sprawozdania.

```
At time 1634830380.909
- Translation: [-0.434, 0.499, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.895, 0.446]
           in RPY (radian) [0.000, -0.000, 2.218]
           in RPY (degree) [0.000, -0.000, 127.074]
```

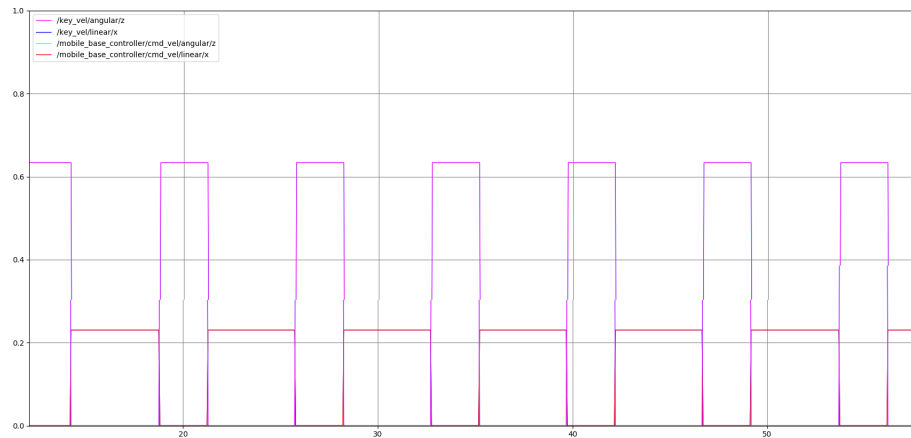
```
At time 1634830659.457
- Translation: [-0.431, 0.601, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.876, 0.482]
           in RPY (radian) [0.000, -0.000, 2.137]
           in RPY (degree) [0.000, -0.000, 122.432]
```

```
At time 1634830792.527
- Translation: [-0.425, 0.700, 0.000]
- Rotation: in Quaternion [0.000, 0.000, 0.858, 0.514]
           in RPY (radian) [0.000, -0.000, 2.061]
           in RPY (degree) [0.000, -0.000, 118.104]
```

3.0.5. rqtplot

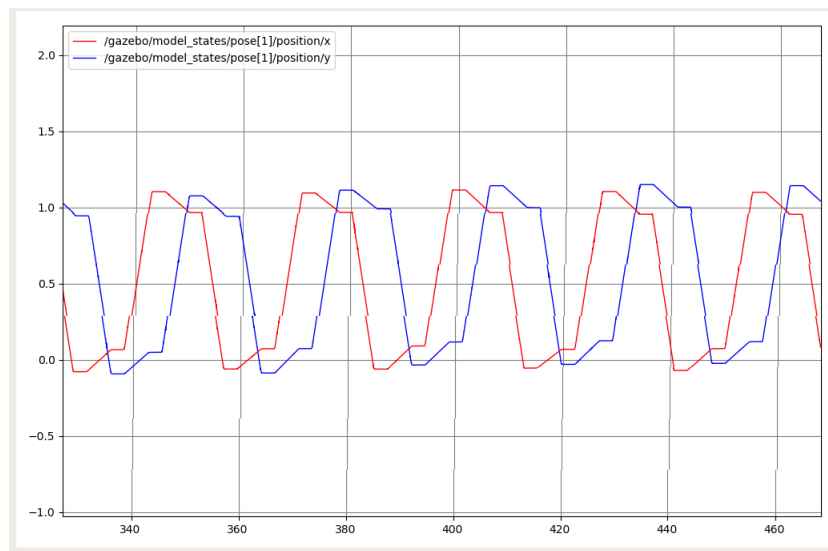
Narzędzie `rqt_plot` pozwoliło nam na wizualizację oraz porównywanie prędkości i/lub pozycji podczas przeprowadzania symulacji. Dzięki temu narzędziu mogliśmy zweryfikować, czy prędkość zadana jest zgodna z prędkością rzeczywistą.

Poniższy wykres pokazuje, że prędkości zadane oraz wykonane są zgodne.



Rys. 3.10: Prędkości zadane i wykonane

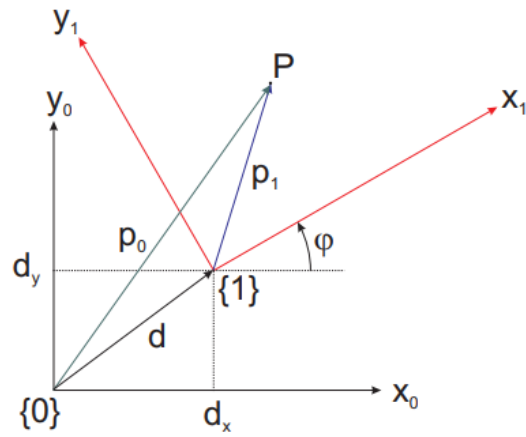
Narzędzie `rqt_plot` pozwoliło nam również na wizualizację danych obrazujących pozycję referencyjną robota.

Rys. 3.11: Pozycja robota z *Gazebo*

3.0.6. Analiza danych z pliku *.bag*

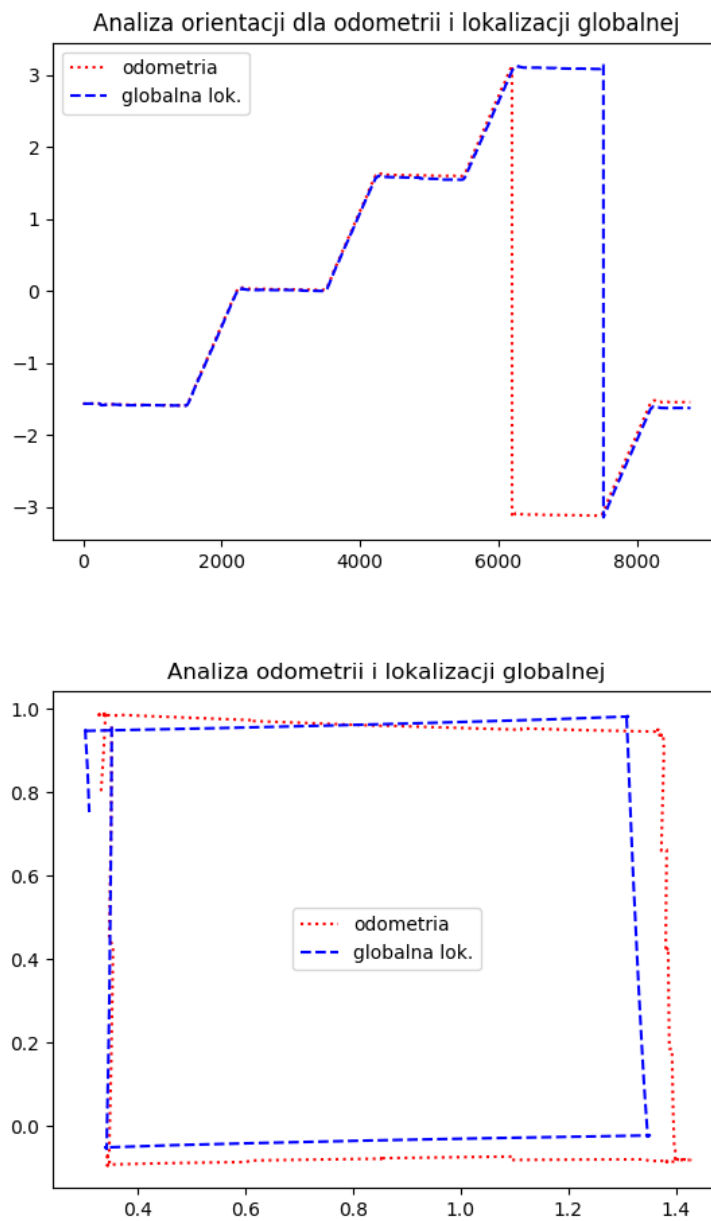
Dane odometryczne w temacie `/mobile_base_controller/odom` są wyrażone w układzie `odom`, a dane z lokalizacji globalnej w układzie `map`. Transformacja między tymi układami jest zmienna w trakcie pracy robota i wynika z działania algorytmu globalnej lokalizacji. Zatem, aby porównać lokalizację robota na podstawie odometrii oraz lokalizację z fuzją innych danych sensorycznych należy porównywać dane z obu wskazanych tematów względem wspólnego układu współrzędnych. Za ten układ należy przyjąć układ `map`.

Przekształcenie wykonaliśmy zgodnie z poniższą grafiką (slajd z wykładu *Wstęp do robotyki* autorstwa dr hab. inż. Wojciecha Szynkiewicza, EiT PW)

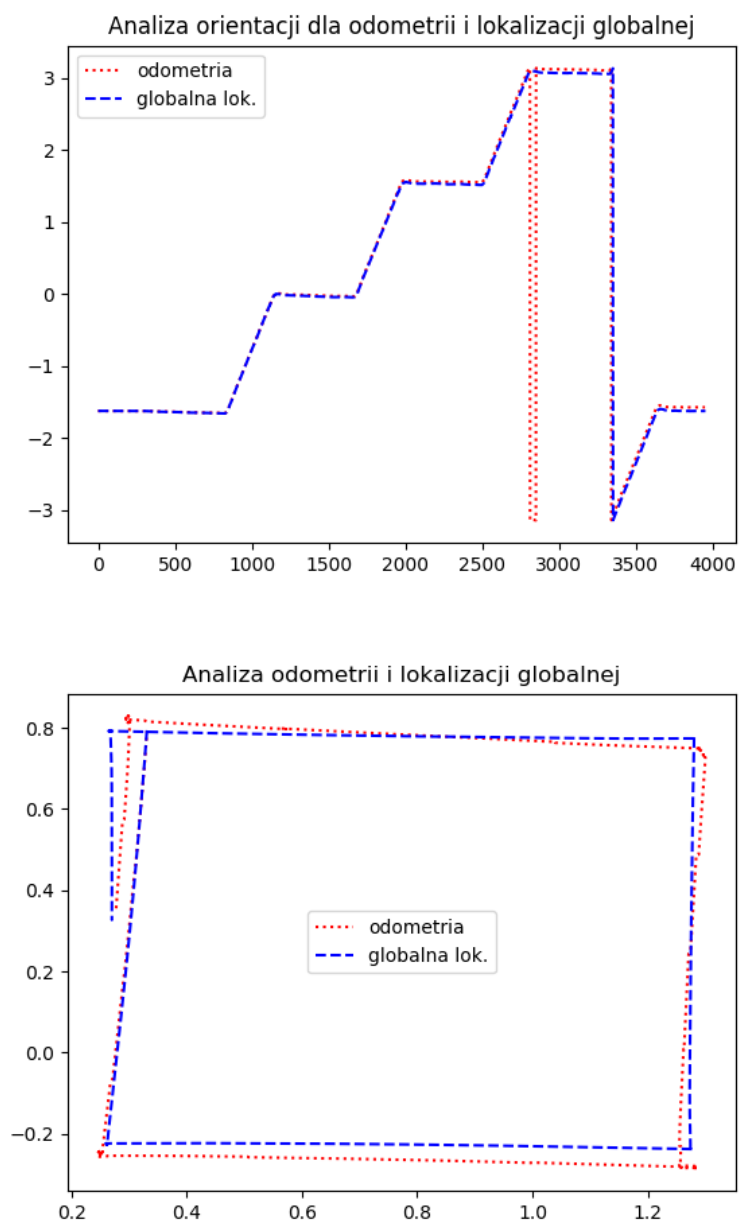
Rys. 2: Obrót i przesunięcie w \mathbb{R}^2

$$\begin{bmatrix} p_{0x} \\ p_{0y} \end{bmatrix} = \begin{bmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{bmatrix} \begin{bmatrix} p_{1x} \\ p_{1y} \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

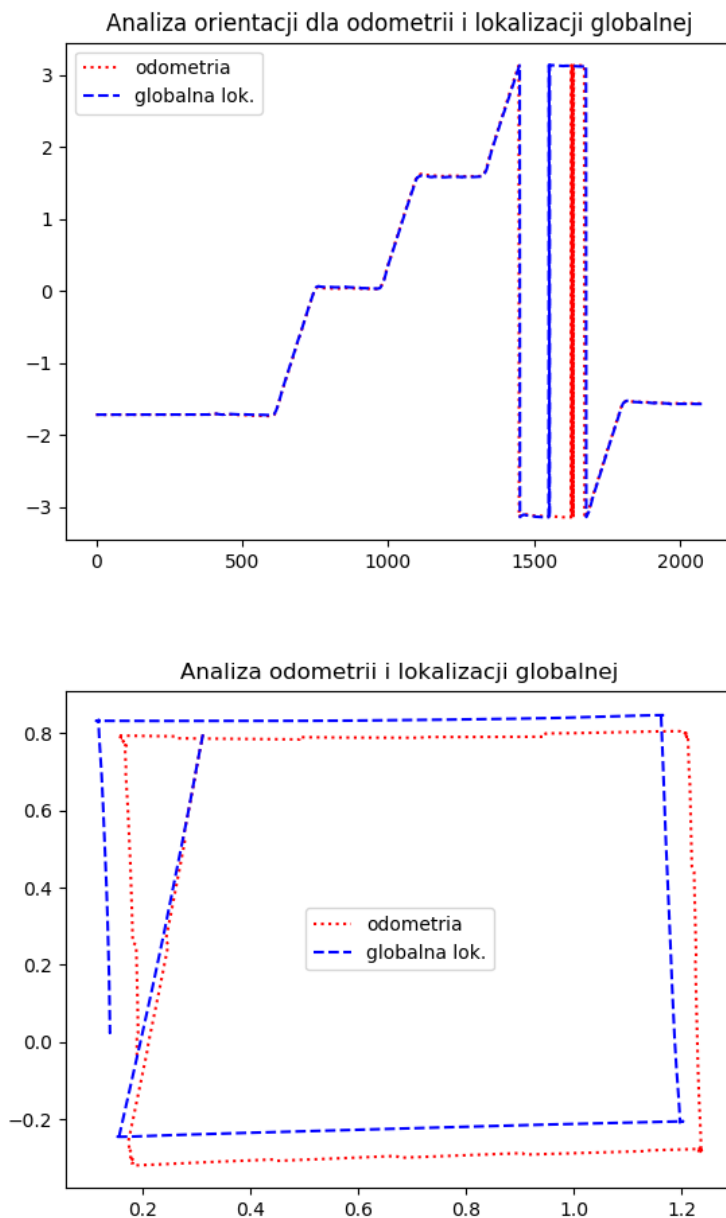
Rys. 3.12: Transformacja między układami współrzędnych na płaszczyźnie

Ruch rzeczywistego robota

Rys. 3.13: Test kwadratu z odometrią, prędkość niska



Rys. 3.14: Test kwadratu z odometrią, prędkość średnia



Rys. 3.15: Test kwadratu z odometrią, prędkość wysoka

Przedstawione przebiegi orientacji dla odometrii i lokalizacji globalnej są ze sobą zgodne. Widoczne rozbieżności wynikają z niewielkich błędów/szumów pomiarowych i faktu zawierania się miary kątowej w zakresie od $-\pi$ do π .

Wyniki pomiarów

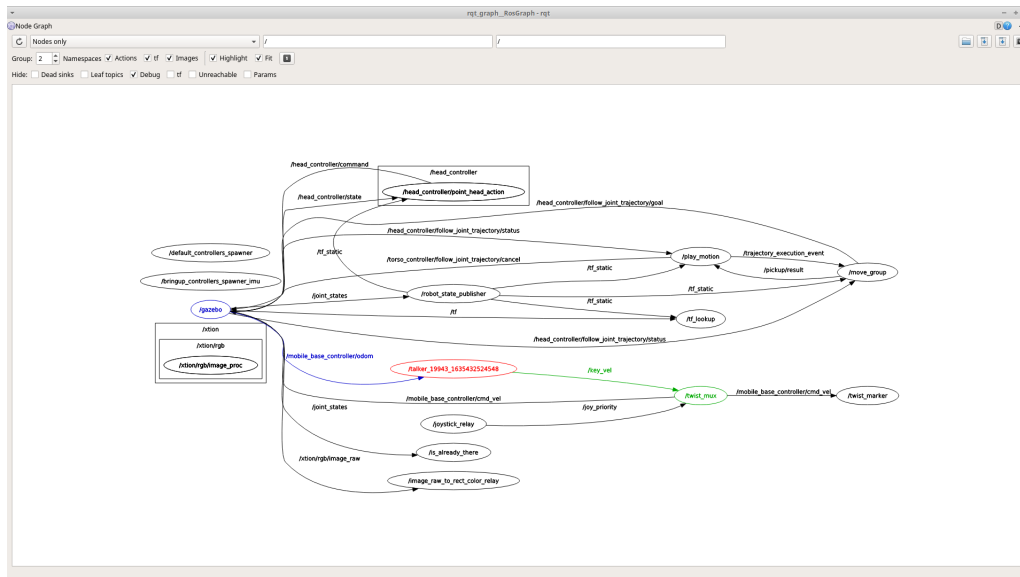
Tab. 3.1: Porównanie błędów pozycji dla różnych prędkości

Prędkość	Błąd x		Błąd y		Błąd theta	
	Skumulowany	Średni	Skumulowany	Średni	Skumulowany	Średni
low	61.17	0.0294	92.95	0.0447	521	0.2506
medium	48.06	0.0122	99.35	0.0252	6327	1.6018
high	276.52	0.0315	308.98	0.0352	19730	2.2498

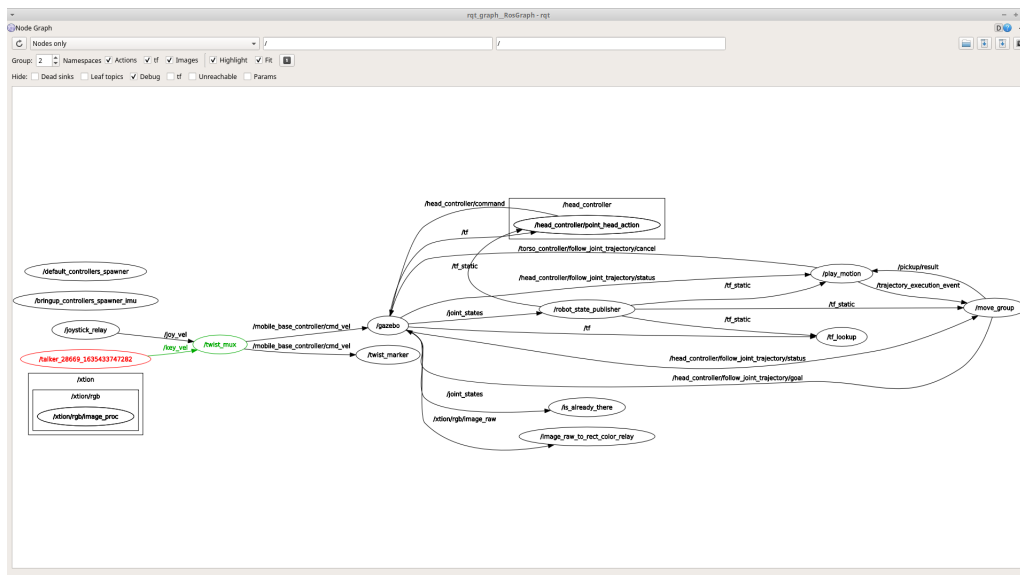
W przypadku zbyt szybkich ruchów robota tracimy dokładność pozycjonowania go w przestrzeni. Ograniczone przyspieszenie robota wpływa szczególnie na niedokładność ruchu obrotowego, przez co dokładne wykonanie wyznaczonej trajektorii może być awykonalne.

3.0.7. Struktura komunikacji

W celu zobrazowania struktury sterującej robotem *Tiago* w ramach ROS (*Robot Operating System*) załączmy poniżej grafy struktur komunikacji tzn. zależności pomiędzy poszczególnymi węzłami i tematami.



Rys. 3.16: Sterowanie z odometrią - struktura komunikacji



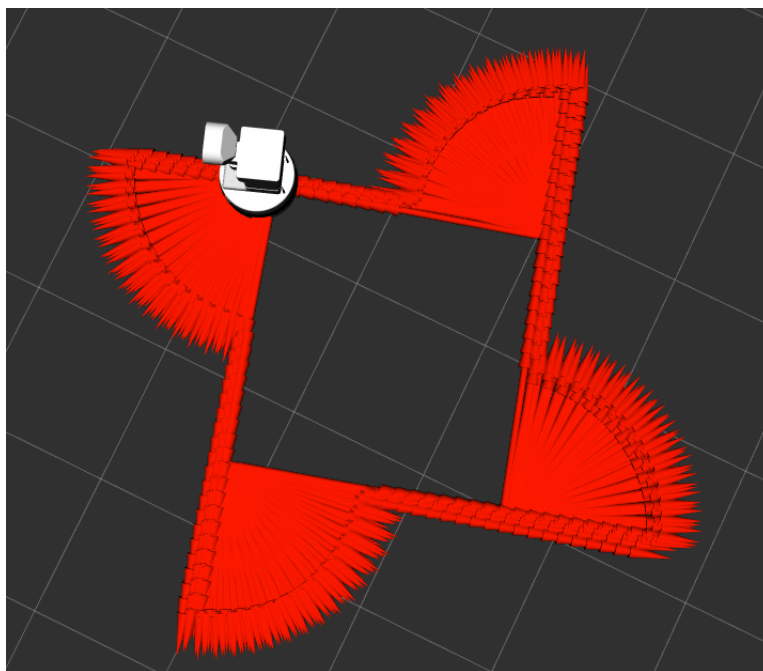
Rys. 3.17: Sterowanie wyłącznie prędkościowe - struktura komunikacji

4. Podsumowanie

Testy przeprowadzone dla sterowania prędkościowego wykazały zgodność prędkości zadanej z prędkością wykonaną. Mimo to ten rodzaj sterowania musi zakładać całkowity brak oddziaływań zewnętrznych na robota. Najmniejszy poślizg, czy też nierówna powierzchnia, mogą spowodować, że robot będzie "przekonany", że dojechał do celu - w rzeczywistości może być daleko od niego.

Odometria jest bardziej skomplikowana od samego sterowania prędkościowego. Korzystamy w niej z danych z enkoderów zamontowanych na wałach silników, które informują nas o dystansie przebytych przez robota oraz o kącie, o jaki wykonał obrót. Nierówności terenu, ściany, poślizgi, czy też manualne podniesienie i postawienie robota w innym miejscu powodują zagubienie informacji o lokalizacji. Odometria robota bazuje na fizycznych wielkościach takich jak rozstaw i średnica kół, czy też rozdzielczość enkoderów. Drobne niedokładności w wartościach tych parametrów powodują akumulację błędów. Zaobserwowaliśmy to podczas testów z prawdziwym robotem.

Na poniższej grafice widać ścieżkę wykreśloną przez robota. Po wykonaniu kilku, symulowanych przejazdów w ramach testu kwadratu widać już wyraźnie, że kolejne ścieżki nie pokrywają się idealnie.



Rys. 4.1: Akumulacja błędów odometrii