Simulacija robota penjača u Gazebo simulatoru

Crnić, Edi

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Simulacija robota penjača u Gazebo simulatoru

Edi Crnić

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:
doc.dr.sc. Marko Švaco

Student:
Edi Crnić

Zagreb, 2023.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija te navedenu literaturu.

Zahvaljujem se doc.dr.sc. Marku Švaci te mag.ing.mech. Branimiru Ćaranu na ideji, pomoći, strpljenju i motivaciji tokom izrade završnog rada.

Također, zahvaljujem se svojoj obitelji, prijateljima i kolegama na pruženoj podršci tijekom studija.

Edi Crnić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala, autonomni sustavi i računalna inteligencija i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 23 – 6 / 1**	
Ur.broj: **15 - 1703 - 23 -**	

ZAVRŠNI ZADATAK

Student: **Edi Crnić** JMBAG: **0035229150**

Naslov rada na hrvatskom jeziku: **Simulacija robota penjača u Gazebo simulatoru**

Naslov rada na engleskom jeziku: **Simulation of a wall climbing robot in the Gazebo simulator**

Opis zadatka:

Robot penjač je mobilni robot koji ima funkciju vožnje po vertikalnim površinama. Robot sadrži EDF (eng. Electric Ducted Fan) koji služi za generiranje sile adhezije i četiri elektromotora bez četkica s propelerima za kompenzaciju mase robota. Robot ima četiri nezavisno pogonjena i nezavisno zakretna kotača (eng. four-wheel-independent steering and four-wheel-independent driving - 4WIS4WID) koji mu omogućuju gibanje u svim smjerovima bez promjene apsolutne orijentacije tijela robota. Zbog rizika od padova i učinkovitijeg testiranja, u sklopu ovog završnog rada potrebno je napraviti model robota u simulacijskom okruženju Gazebo u sklopu ROS-a (eng. Robot Operating System) kako bi se testirali algoritmi navigacije i lokalizacije mobilnog robota. U sklopu ovog rada potrebno je:

- Stvoriti URDF (eng. Unified Robot Description Format) datoteku u kojoj su učitani modeli mobilnog robota, svi odnosi pokretnih dijelova robota, momenti inercije svih pokretnih dijelova te faktor trenja podloge i kotača.
- Upravljati svim varijablama stanja (kut zakreta svakog kotača, brzina vrtnje svakog kotača i brzina vrtnje potisnih motora) pomoću ROS-a.
- Pronaći adekvatnu građevinsku infrastrukturu (npr. most, vijadukt, nosivi stup) za simulaciju ponašanja robota, implementirati CAD model građevinske infrastrukture u Gazebo simulator te simulirati kretanje robota.
- Raspisati i implementirati zakon inverzne i direktne kinematike zadanog robota penjača.
- Simulirati kretanje robota po odabranoj građevinskoj infrastrukturi u Gazebo simulatoru pomoću ROS-a.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
20.4.2023.	**2. rok (izvanredni):** 12. 7. 2023. **3. rok:** 21. i 22.9. 2023.	**2. rok (izvanredni):** 14. 7. 2023. **3. rok:** 25. 9. – 29. 9. 2023.

Zadatak zadao: Predsjednik Povjerenstva:

Doc. dr. sc. Marko Švaco Prof. dr. sc. Damir Godec

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
x_{wi}^{r}	m	koordinate kotača
y_{wi}^{r}	m	koordinate kotača
v	m/s	linearna brzina robota
v_i	m/s	linearna brzina kotača
ω	rad/s	kutna brzina robota
ω_i	rad/s	kutna brzina kotača
φ	rad	kut kotača

SAŽETAK

Moderna tehnologija omogućuje nam upotrebu dostupnih softvera i alata kako bismo pojeftinili, pojednostavnili i osigurali proces testiranja. Samim time, glavni cilj ovoga rada bio je izvršiti simulaciju robota penjača po horizontalnoj, a tako i po vertikalnoj površini. Prikazan je općeniti uvid u robote penjače, kinematika robota te svi koraci koji prethode simulaciji, tj. koraci koji su nužni za izvođenje simulacije. Za proces simulacije korišten je operativni sustav Ubuntu 20.04 u koji je instaliran Robot Operating System (ROS) koji upravlja robotom u Gazebo simulatoru. Također, prikazan je proces izrade URDF datoteke te svih ostalih datoteka bez kojih simulacija ne bi mogla funkcionirati. Skripta robota kreirana je ručno te pomoću programskog paketa Solidworks 2020.

Ključne riječi: ubuntu, ros, gazebo, robot, rviz, model, urdf, virtualbox, stl, solidworks

SUMMARY

Modern technology allows us to use available software and tools to make testing processes cheaper, simpler, and more secure. Therefore, the main goal of this work was to simulate a climbing robot on both horizontal and vertical surfaces. It provides a general overview of climbing robots, robot kinematics, and all the steps preceding the simulation, i.e., the steps necessary for conducting the simulation. The simulation process was carried out on the Ubuntu 20.04 operating system with the Robot Operating System (ROS) installed, which controls the robot in the Gazebo simulator. Additionally, it illustrates the process of creating URDF files and all other necessary files without which the simulation could not function. The robot script was created manually as well as with the help of the Solidworks 2020 software package.

Keywords: ubuntu, ros, gazebo, robot, rviz, model, urdf, virtualbox, stl, solidworks

1. UVOD

1.1. Mobilni roboti

Mobilni robot je robot kojim upravlja softver koji koristi senzore i ostale tehnologije kako bi bio u mogućnosti prepoznati okolinu po kojoj se kreće i izvršavati radnje koje su mu namijenjene. Fizički robotski elementi (gusjenice, noge, kotači i sl.) zajedno s umjetnom inteligencijom daju funkcionalnu cjelinu koja je potrebna za rad robota. Kako se mobilni roboti koriste za obavljanje zadataka koji su nemogući ili vrlo opasni za ljude (npr. mostovi, zgrade, nuklearne elektrane...), njihova je upotreba sve češća. Jedna od glavnih prednosti mobilnih robota je njihova sposobnost računalnog vida koji služi za otkrivanje okoline u stvarnom vremenu. Također, vrlo su fleksibilni i brzi za implementaciju. Naravno, potrebno je i navesti neke mane mobilnih robota na koje je potrebno obratiti pažnju, primjerice ograničenja veličine tereta, komplikacije s bežičnim vezama između robota i krajnje informacijske točke itd. Mobilni se roboti najčešće kvalificiraju prema okruženju u kojem rade i prema uređaju kojim se kreću, a dijele se na autonomne i vođene mobilne robote.

1.2. Mobilni roboti penjači

Jedan od mnogobrojnih primjera mobilnih robota upravo su roboti penjači, roboti koji imaju funkciju kretanja po vertikalnim površinama. Moguće su brojne izvedbe, primjerice robot koji koristi silu adhezije i četiri elektromotora s propelerima za kompenzaciju mase te četiri nezavisno pogonjena kotača koji mu omogućavaju gibanje u svim smjerovima. Upravo se na prethodno danom primjeru robota penjača bazira ovaj rad. Sljedeća slika prikazuje jednu od mnogobrojnih izvedbi robota penjača.

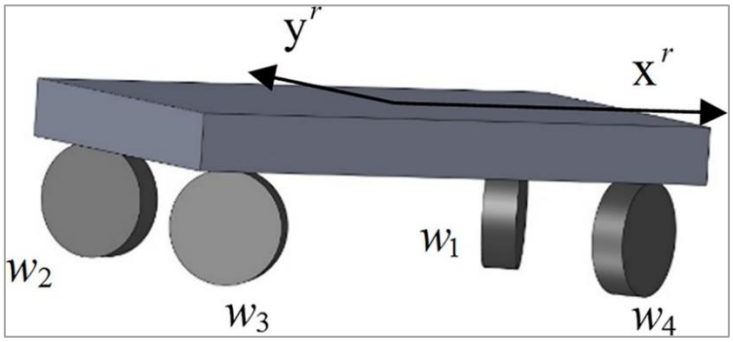
Slika 1. Primjer robota penjača – ICM Climber [1]

Tablica 1. Karakteristike robota ICM Climber

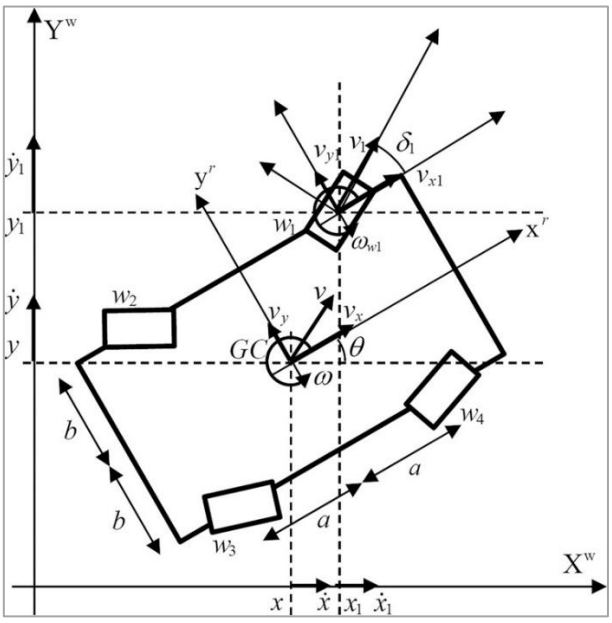
masa	13,61 kg
širina	0,61 m
dužina	0,61 m
visina	0,20 m
brzina	0,08 m/s
snaga	24 Volt DC/110 Volt AC/15 amp

2. ROBOT PENJAČ 4WIS4WID STRUKTURE

Opisat će se pojednostavljeni mobilni robotski sustav s četiri nezavisno pogonjena i nezavisno zakretna kotača (4WIS4WID).



Slika 2. 4WIS4WID sustav robota [2]



Slika 3. Konfiguracija sustava robota u globalnom koordinatnom sustavu [2]

2.1. Kinematički model robota

Prije razvoja kinematičkog modela robota pretpostavljene su tri konfiguracije robota s obzirom na sliku 3:

1) Položaj svih kotača u koordinatnom okviru robota unaprijed je denifiran kao:

$$(x_{wi}^r, y_{wi}^r) \tag{1}$$

$$(x_{w1}^r, y_{w1}^r) = (a, b) \tag{2}$$

$$(x_{w2}^r, y_{w2}^r) = (-a, b) \tag{3}$$

$$(x_{w3}^r, y_{w3}^r) = (-a, -b) \tag{4}$$

$$(x_{w4}^r, y_{w4}^r) = (a, -b) \tag{5}$$

Jednadžba (1) opći je izraz za koordinate kotača *i* u robotskom koordinatnom okviru.

$$v = \sqrt{v_x^2 + v_y^2} \tag{6}$$

$$v_i = \sqrt{v_{xi}^2 + v_{yi}^2} \tag{7}$$

Jednadžbe (6) i (7) izrazi su za linearnu brzinu robota i kotača *i*. Kutna brzina robotskog tijela označena je kao ω, dok je ω_i brzina zakretanja kotača *i*.

2) Centar mase i težište robota se podudaraju
3) Pretpostavlja se da su masa i radijus svih kotača jednaki

U idealnim uvjetima, tj. uvjetima bez klizanja, odnosi brzine između svakog kotača i tijela robota dobiveni su ograničenjem krutog tijela robota, a predstavljeni su kao:

$$v_{xi} = v_i \cos\delta_i = v_x - x_{wi}^r \omega \tag{8}$$

$$v_{yi} = v_i \sin\delta_i = v_y - y_{wi}^r \omega \tag{9}$$

Zamjenom parametara četiri kotača u jednadžbama (8) i (9) odnos kotača i tijela robota može se opisati kao:

$$P\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = X\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{10}$$

$$P = \begin{bmatrix} 1 & 0 & -b \\ 0 & 1 & a \\ 1 & 0 & -b \\ 0 & 1 & -a \\ 1 & 0 & b \\ 0 & 1 & -a \\ 1 & 0 & b \\ 0 & 1 & a \end{bmatrix} \tag{11}$$

$$P\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = X\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{12}$$

$$X = \begin{bmatrix} \cos\delta_1 & 0 & 0 & 0 \\ \sin\delta_1 & 0 & 0 & 0 \\ 0 & \cos\delta_1 & 0 & 0 \\ 0 & \sin\delta_1 & 0 & 0 \\ 0 & 0 & \cos\delta_1 & 0 \\ 0 & 0 & \sin\delta_1 & 0 \\ 0 & 0 & 0 & \cos\delta_1 \\ 0 & 0 & 0 & \sin\delta_1 \end{bmatrix} \tag{13}$$

Pseudo-inverz matrice *P* glasi:

$$P^* = \begin{bmatrix} \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{b}{K} & \frac{a}{K} & -\frac{b}{K} & -\frac{a}{K} & \frac{b}{K} & -\frac{a}{K} & \frac{b}{K} & \frac{a}{K} \end{bmatrix} \tag{14}$$

$$K = 4a^2 + 4b^2 \tag{15}$$

Pomnožimo li jednadžbu (12) s jednadžbom (14) dobije se odnos između četiri kotača i tijela robota:

$$\begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\cos\delta_1}{4} & \frac{\cos\delta_2}{4} & \frac{\cos\delta_3}{4} & \frac{\cos\delta_4}{4} \\ \frac{\sin\delta_1}{4} & \frac{\sin\delta_2}{4} & \frac{\sin\delta_3}{4} & \frac{\sin\delta_4}{4} \\ W_1 & W_2 & W_3 & W_4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \tag{16}$$

$$W_i = \frac{-y_{wi}^r \cos\delta_i + x_{wi}^r \sin\delta_i}{4(x_{wi}^r)^2 + 4(y_{wi}^r)^2} \tag{17}$$

Kombinirajući tri stanja tijela robota, kut rotacije kotača i kut upravljanja kotača, postoji 11 varijabli stanja koje predstavljaju položaj i držanje robota:

$$q = [x \quad y \quad \theta \quad \varphi_1 \quad \varphi_2 \quad \varphi_3 \quad \varphi_4 \quad \delta_1 \quad \delta_2 \quad \delta_3 \quad \delta_4]^{\mathrm{T}} \tag{18}$$

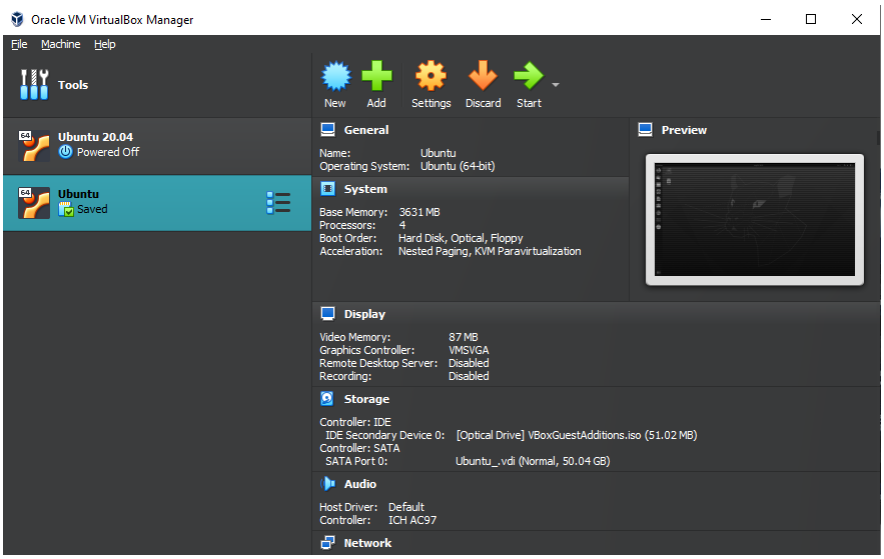
Nakon nekoliko izračuna može se dobiti kinematski model mobilnog 4WIS4WID robota koji predstavlja vektor brzine $\dot{q}$ u globalnom koordinatnom sustavu:

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\varphi}_1 \\ \dot{\varphi}_2 \\ \dot{\varphi}_3 \\ \dot{\varphi}_4 \\ \dot{\delta}_1 \\ \dot{\delta}_2 \\ \dot{\delta}_3 \\ \dot{\delta}_4 \end{bmatrix} = \begin{bmatrix} \frac{\cos(\delta_1+\theta)}{4} & \frac{\cos(\delta_2+\theta)}{4} & \frac{\cos(\delta_3+\theta)}{4} & \frac{\cos(\delta_4+\theta)}{4} & 0 & 0 & 0 & 0 \\ \frac{\sin(\delta_1+\theta)}{4} & \frac{\sin(\delta_2+\theta)}{4} & \frac{\sin(\delta_3+\theta)}{4} & \frac{\sin(\delta_4+\theta)}{4} & 0 & 0 & 0 & 0 \\ W_1 & W_2 & W_3 & W_4 & 0 & 0 & 0 & 0 \\ r^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

3. ROS I GAZEBO

3.1. Oracle VM VirtualBox

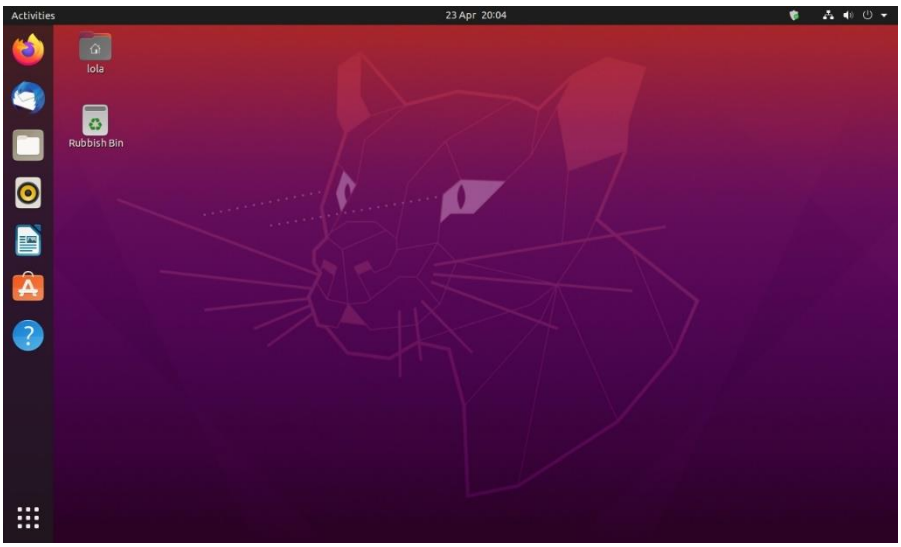
Za pokretanje operacijskog sustava koristi se VirtualBox, jedan od najpoznatijih x86 i AMD64/Intel64 virtualizacijskih proizvoda. Razlog tome je jednostavnije i sigurnije korištenje operacijskog sustava. Sve češće dobiva veći popis značajki i podržanih operacijskig sustava.



Slika 4. Oracle VM VirtualBox Manager [3]

3.2. Ubuntu

Operacijski sustav koji se koristi upravo je Ubuntu, operacijski sustav vrlo sličan Unixu. Kao jezgra operacijskog sustava koristi se Linux. Nastao je kao izvedenica sustava Debian GNU/Linux. Sadrži grafičko korisničko sučelje za osobna računala. Građen je od tzv. distribucija, tj. nekoliko kolekcija softvera.

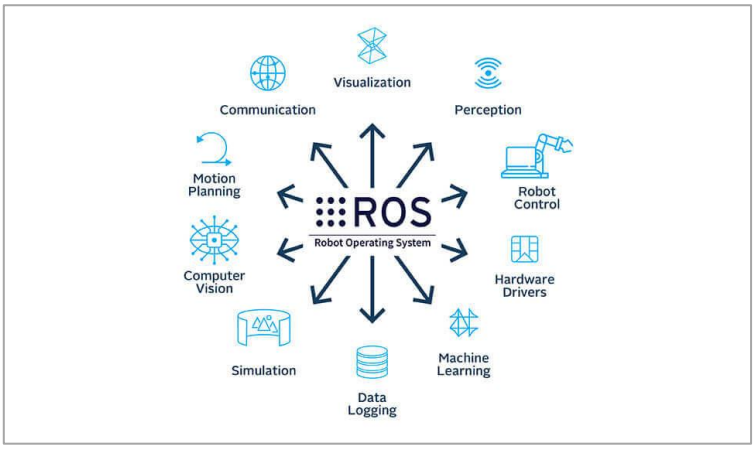


Slika 5. Radna površina sustava Ubuntu 20.04 [4]

3.3. Robot Operating System (ROS)

Robot Operating System (ROS) skup je softverskih alata i biblioteka koji uvelike pomažu pri izradi robotskih aplikacija. ROS nudi standarnu softversku platformu programerima u svim mogućim industrijama koja će ih odvesti do izrade prototipa i proizvodnje. Posjeduje upravljačke programe, najsuvremenije algoritme te snažne razvojne alate. U sklopu ovog rada korišten je ROS Noetic, izdanje ROS1 LTS namijenjeno već navedenom operacijskom sustavu Ubuntu 20.04. Funkcionalnost ROS-a proširena je nizom alata koji omogućuju vizualizaciju i snimanje podataka, jednostavan pristup paketima te stvaranje skripti automatizirajući složene procese konfiguracije i upravljanja. Dodavanjem alata ROS dobiva veće mogućnosti te samim time lakše dolazi do rješenja. Neki od najčešće korištenih alata su:

1) rviz – vizualizator robota, okruženja i podataka
2) rosbag – snimanje i reprodukcija podataka ROS poruka
3) catkin – build sustav
4) rosbash – funkcionalnost bash ljuske
5) roslaunch – pokretanje više ROS čvorova lokalno i udaljeno te postavljanje parametara



Slika 6. Robot Operating System (ROS) [6]

Nadalje se navodi postupak instalacije ROS-a. Nakon pokretanja terminala u Ubuntu potrebno je upisati nekoliko naredbi za ROS instalaciju.

1) sources.list setup

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc
/apt/sources.list.d/ros-latest.list'
```

2) keys setup

```
sudo apt install curl # if you haven't already installed curl

curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | s
udo apt-key add -
```

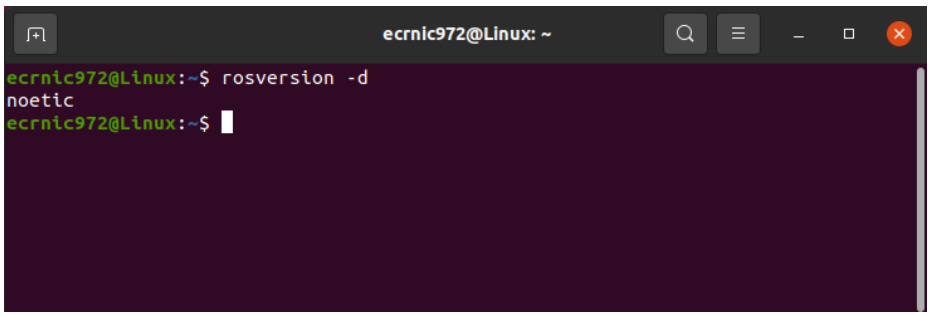
3) installation

```
sudo apt update

sudo apt install ros-noetic-desktop-desktop
```

4) environment setup

```
source /opt/ros/noetic/setup.bash

echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc

source ~/.bashrc
```

Kako bismo provjerili da je ROS pravilno instaliran, jedno od mogućih rješenja je zatražiti ispis verzije.



Slika 7. ROS verzija

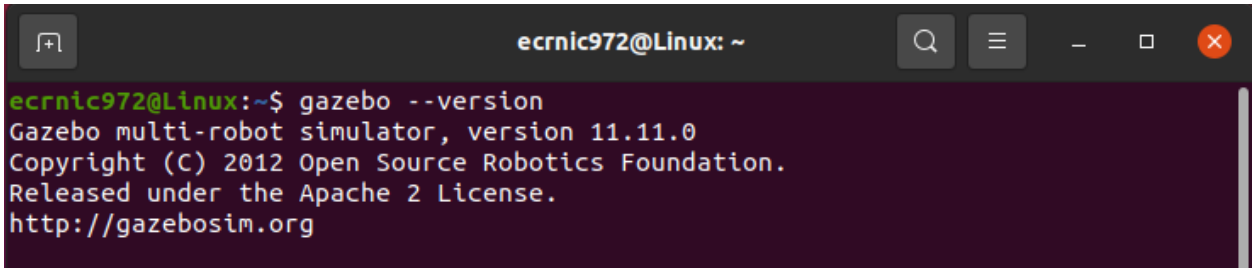
3.4. Gazebo simulator

Gazebo je 3D robotski simulator otvorenog koda s integriranim motorom ODE, OpenGL renderiranjem i kodom podrške za simulaciju senzora i kontrolu aktuatora. Omogućuje pristup simulaciji s kompletnim paketom alata razvojnih usluga i biblioteka u oblaku koji čine simulaciju lakšom, realističan prikaz okruženja uz kvalitetno osvjetljenje itd.

Nakon instalacije svih potrebnih paketa i potrebne nadogradnje, instalacija Gazeba vrši se upisom sljedećeg koda u Ubuntu terminal:

```
sudo apt-get install gazebo11

# For developers that work on top of Gazebo, one extra package

sudo apt-get install libgazebo11-dev
```

Da bismo provjerili je li Gazebo u potpunosti instaliran potrebno je upisati sljedeći kod u Ubuntu terminal.



Slika 8. Gazebo verzija

3.5. Workspace i Catkin package

Workspace, tj. radni prostor obično se odnosi na strukturu direktorija gdje su organizirane datoteke, modeli, dodatci te razne konfiguracijske datoteke povezane sa simulacijom. Postavljanje radnog prostora dobra je praksa zbog nekoliko glavnih razloga kao što su organizacija, dijeljenje,

ponovljivost, prilagodba i sl. Postavljanje radnog prostora za Gazebo simulacije uključuje stvaranje strukture direktorija koja odgovara korisničkim potrebama. Pri tome se uključuju mape za modele, datoteke, skripte i još mnogo toga. Na taj način pojednostavljuje se proces razvoja simulacije te je rad učinkovitiji. Za kreiranje radnog prostora potrebno je provesti nekoliko jednostavnih operacija u Ubuntu terminalu.

Prvi je korak kreirati datoteku robot_ws koja će predstavljati radni prostor te datoteku src (source) unutar robot_ws datoteke:

```
mkdir -p/robot_ws/src
```

Da bismo uspješno kreirali radni prostor potrebno je navigirati do robot_ws datoteke i provesti nekoliko uzastopnih naredbi:

```
cd robot_ws/

catkin_make

source /robot_ws/devel/setup.bash
```

U prethodno kreiranoj src datoteci potrebno je kreirati nekoliko paketa:

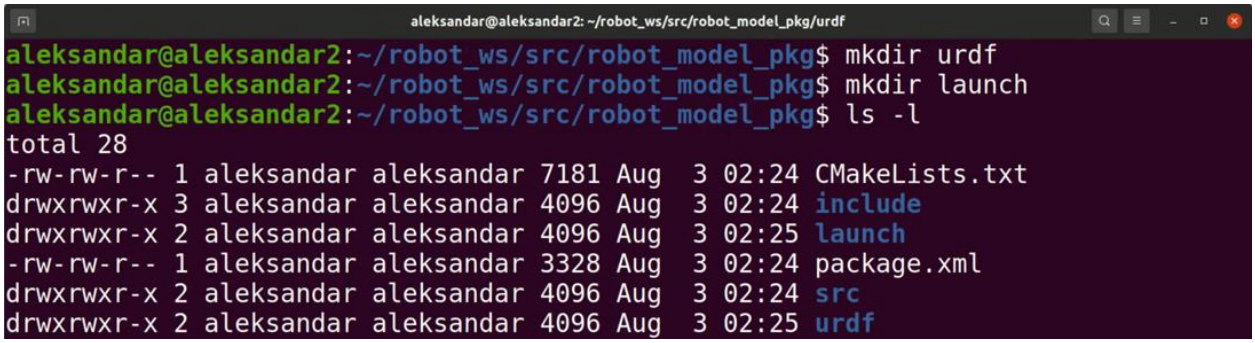
```
cd src

catkin_create_pkg robot_model_pkg roscpp tf2 geometry_msgs urdf rviz joint_state_publ
isher_gui
```

Slika 9. Kreiranje datoteka i paketa [14]

Sljedeći je korak kreirati dvije nove datoteke unutar prethodno kreirane datoteke pod nazivom robot_model_pkg:

```
cd robot_model_pkg
```

```
mkdir urdf

mkdir launch
```



Slika 10. Kreiranje urdf i launch datoteke

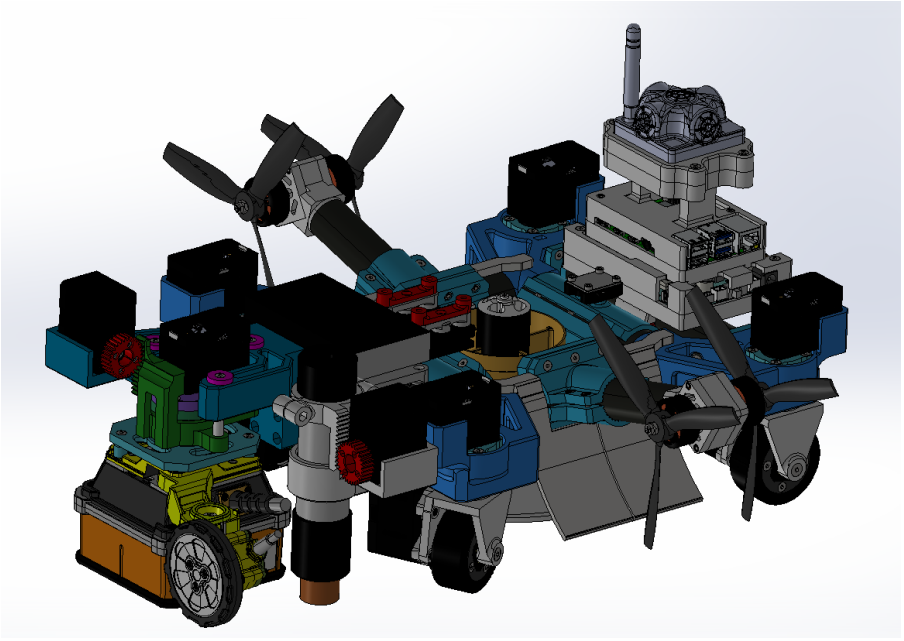
4. SIMULACIJSKI MODEL ROBOTA

Unified robot description format (URDF) vrsta je datoteke proširivog označnog jezika (XML) koja uključuje fizički opis modela, tj. robota. U URDF-u opisane su informacije o zglobovima, motorima, masi, momentima inercije, odnosi pokretnih dijelova i sl. Nakon definirane URDF datoteke ista se pokreće pomoću već navedenog ROS-a. Iz URDF-a moguće je direktno očitati za što je robot sposoban i prije upravljanja tim robotom. Također, URDF datoteka često igra ključnu ulogu u procesu donošenja odluka raznih tvrtki koje kupuju robota. URDF datoteke najčešće se koriste u simulatorima za testiranje ponašanja robota prije implementacije u stvarni svijet što je upravo i slučaj u ovome radu. Sljedeći kod primjer je stvaranja valjka duljine 0.6m i polumjera 0.2m.

```
<?xml version="1.0"?>
<robot name="myfirst">
   <link name="base_link">
     <visual>
       <geometry>
         <cylinder length="0.6" radius="0.2"/>
       </geometry>
     </visual>
   </link>
</robot>
```

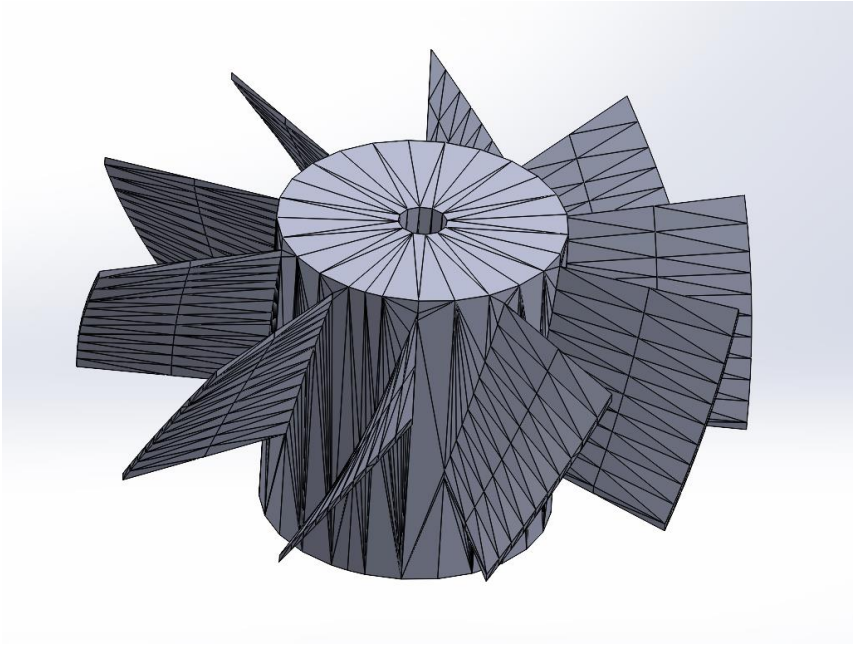
4.1. Solidworks .stl export

Za potrebe ovog rada dostupan je 3-D model robota penjača što je ujedno i prvi korak stvaranja URDF datoteke.



Slika 11. 3-D model robota penjača

Potrebno je sve dijelove koji će utjecati na kretanje robota (primjerice adhezijski propeler, propeleri za kompenzaciju mase, kotači itd.) spremiti kao STL datoteku koja će biti povezana s URDF datotekom. Također, ostali dio robota predstavljat će *base_link*, odnosno dio robota koji tipično predstavlja okvir robota. *Base_link* bitan je koncept u robotici jer definira početnu točku za pozicioniranje i orijentaciju drugih dijelova robota u odnosu na njega. Sljedeća slika prikazuje jednu od potrebnih STL datoteka.



Slika 12. Propeler za silu adhezije u STL formatu [12]

4.2. Solidworks to URDF Exporter

Jedan od mogućih načina kreiranja URDF datoteke upravo je direktno preko Solidworks plugina Solidworks to URDF. Taj je način idealan za jednostavnije robote, no kako zadani robot penjač sadrži veći broj zglobova i ostalih dijelova URDF datoteka kreirana je postepeno ručnim putem.

4.3. Opis robota putem URDF datoteke

Sljedeći kod definicija je robota pod nazivom "robot" s raznim vezama i zglobovima. To uključuje vezu glavnog tijela ("base_link"), aktivne veze kotača, veze propelera, veze kotača i veze adhezijskog propelera. Za spajanje ovih djelova koriste se zglobovi koji predstavljaju rotacije kotača i propelera. Kod specificira inercijska svojstva, vizualizaciju, detalje sudara i karakteristike zglobova za svaku komponentu, pridonoseći fizičkoj strukturi robota i ponašanju u simulacijama. Sve potrebne informacije o robotu kao što su masa, momenti inercije, centar mase i sl. preuzeti su iz 3-D modela robota.

Kako bismo pobliže objasnili navedeni kod izdvojit ćemo dio koda i detaljnije ga opisati, primjerice dio koji se odnosi na zadnji desni kotač robota.

```
<link
   name="rr_wheel_link">
   <inertial>
     <origin
       xyz="0 -0.00153 0"
       rpy="0 0 0" />
     <mass
       value="0.04390" />
     <inertia
       ixx="9408.90E-09"
       ixy="6.98E-09"
       ixz="-0.14E-09"
       iyy="14079.84E-09"
       iyz="-0.54E-09"
       izz="9436.08E-09" />
   </inertial>
   <visual>
     <origin
       xyz="0 0 0"
       rpy="0 0 0" />
```

```
      <geometry>
        <mesh
          filename="package://test1/meshes/rr_wheel.STL" scale="0.001 0.001
0.001"/>
      </geometry>
      <material
        name="">
        <color
          rgba="0.75294 0.75294 0.75294 1" />
      </material>
    </visual>
    <collision>
      <origin
        xyz="0 0 0"
        rpy="0 0 0" />
      <geometry>
        <mesh
          filename="package://test1/meshes/rr_wheel.STL" scale="0.001 0.001
0.001"/>
      </geometry>
    </collision>
  </link>
  <joint
    name="rr_wheel_joint"
    type="continuous">
    <origin
      xyz="0 0 -0.03525"
      rpy="3.1416 0 1.5708" />
    <parent
      link="rr_active_wheel_link" />
    <child
      link="rr_wheel_link" />
    <axis
      xyz="0 -1 0" />
    <limit
      effort="100"
      velocity="100" />
    <dynamics
      damping="0.1"
```

```
      friction="0.1" />
  </joint>
</robot>
```

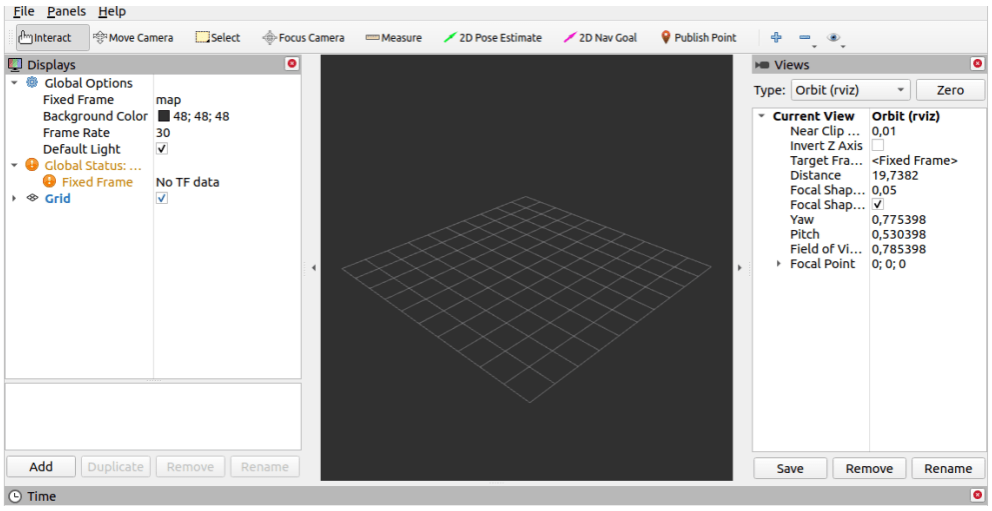
Kao što se vidi u prethodnom dijelu koda, definiran je link pod nazivom rr_wheel_link. Centar njegove mase je pomaknut za 1.53 mm suprotno od osi Y. Masa kotača iznosi 44.9 g. Nakon toga definirane su vizualne i kolizijske karakteristike navedenog linka. Nadalje, definiran je joint (zglob) pod nazivom rr_wheel_joint kontinuiranog tipa zato što nema nikakvih ograničenja u smislu vrtnje. Navedeni zglob definiran je u odnosu na roditeljski link pod nazivom rr_active_wheel_link što znači da su njegove koordinate određene u odnosu na roditeljski link. Drugim riječima, zglob je pomaknut za 35.25 mm suprotno od osi Z. Kako bi model kotača bio dobro orijentiran u odnosu na ostatak robota primjenjene su rotacije za 180 stupnjeva (3.1416 rad) oko osi X te 90 stupnjeva (1.5708 rad) oko osi Z. Također, definiran je faktor trenja između podloge i kotača koji iznosi 0.1. Na isti princip definirani su ostali dijelovi robota i njihove međusobne veze.

4.4. RViz

Rviz predstavlja integralnu komponentu ROS-a kao alat za 3-D vizualizaciju koji ima golemu važnost u robotici i automatizaciji. Služi kao dinamička platforma za vizualizaciju složenih sustava, pomaže u razumijevanju, otklanjanju pogrešaka i poboljšaju robotskih komponenti. Rviz omogućuje 3-D okruženje koje podržava različite vrste podataka. Na taj način moguće je steći uvid u ponašanje i funkcionalnost robotskih sustava. Jedna od velikih značajki RViz-a je sposobnost predstavljanja modela unutar prostora uz vizualizaciju planiranih putanja u stvarnom vremenu. Također, pomaže pri otklanjanju problema tijekom razvoja.

Instalacija RViza poprilično je jednostavna, a vrši se upisivanjem sljedećih naredbi u Ubuntu terminal:

```
sudo apt-get install ros-noetic-rviz

source /opt/ros/indigo/setup.bash

roscore &

rosrun rviz rviz
```



Slika 13. RViz [15]

Kako bismo uspješno vizualizirali model robota u RViz-u potrebno je u prethodno kreiranu launch datoteku upisati sljedeći kod:

```
<launch>
  <arg
    name="model" />
  <param
    name="robot_description"
    textfile="$(find robot_model_pkg)/urdf/robot.urdf" />
  <node
    name="joint_state_publisher_gui"
    pkg="joint_state_publisher_gui"
    type="joint_state_publisher_gui" />
  <node
    name="robot_state_publisher"
    pkg="robot_state_publisher"
    type="robot_state_publisher" />
  <node
    name="rviz"
    pkg="rviz"
    type="rviz"
    args="-d $(find robot_model_pkg)/urdf.rviz" />
</launch>
```

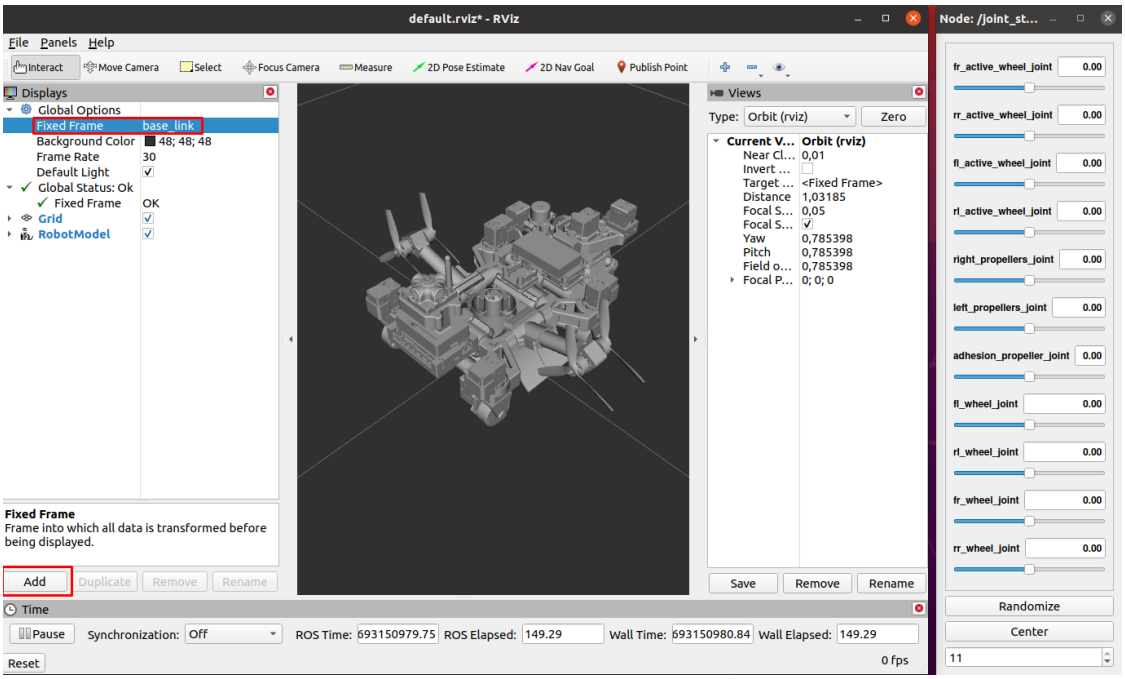
Navedena datoteka za pokretanje predstavlja ROS okruženje za vizualizaciju kinematičkih i geometrijskih informacija robota pomoću RViz-a. Učitava URDF opis robota, pokreće GUI

zajedničkog stanja za ručno kontroliranje zglobova, pokreće izdavač stanja robota za pružanje kinematičkih informacija i inicijalizira RViz konfiguracijskom datotekom za vizualizaciju.

Da bismo započeli vizualizaciju robota potrebno je pokrenuti launch datoteku:

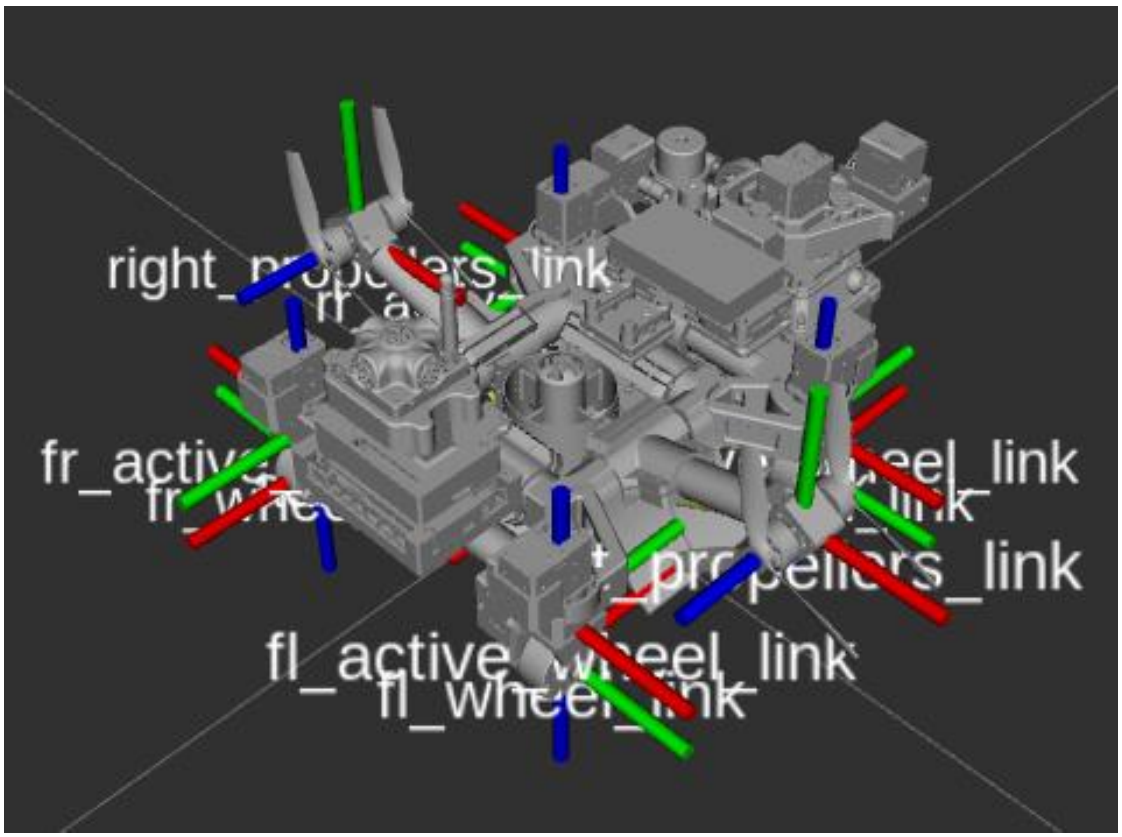
```
roslaunch robot_model_pkg robot.launch
```

Sljedeći je korak Fixed Frame definirati kao base_link te dodati model robota, a postupak je prikazan na sljedećoj slici.



Slika 14. Model robota u RViz-u

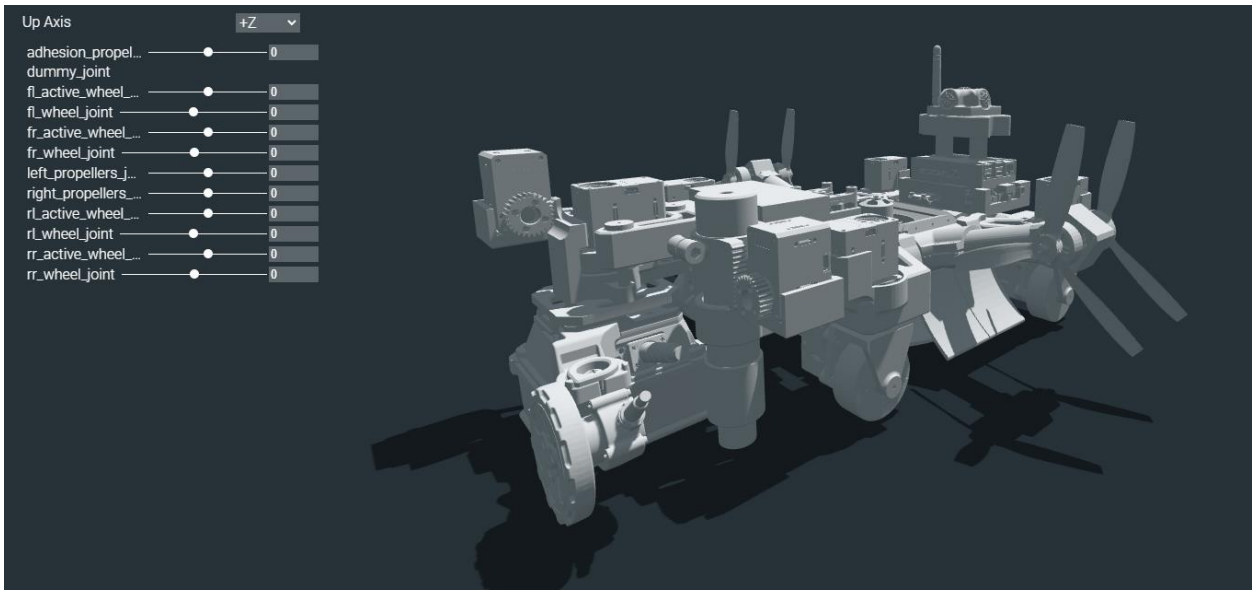
Također, moguće je prikazati koordinatni sustav svakog pomičnog dijela robota kojim je moguće upravljati pomogu joint_state_publisher node-a.



Slika 15. Koordinatni sustavi zglobova

4.5. Web Based URDF Viewer

Kako se radi o već definiranom modelu robota iz kojeg je bilo potrebno izvući STL datoteke, inicijalna URDF datoteka bila je testirana na operativnom sustavu Windows pomoću Web Based URDF Viewer-a pomoću kojeg je također moguće testirati svaki zglob robota.



Slika 16. Web Based URDF Viewer

4.6. Robot spawn

Kako bismo simulirali model robota u realnom okruženju potrebno je isti učitati u Gazebo simulator. Prvi je korak navigirati do mape u kojoj se nalaze svi podatci o robotu te kreirati novu .launch datoteku koje će biti zaslužna za učitavanje robota u Gazebo okruženje.

```
ecrnic972@Linux:~/robot_ws$ cd src
ecrnic972@Linux:~/robot_ws/src$ cd test1
ecrnic972@Linux:~/robot_ws/src/test1$ cd launch
ecrnic972@Linux:~/robot_ws/src/test1/launch$ gedit spawn_model.launch
```

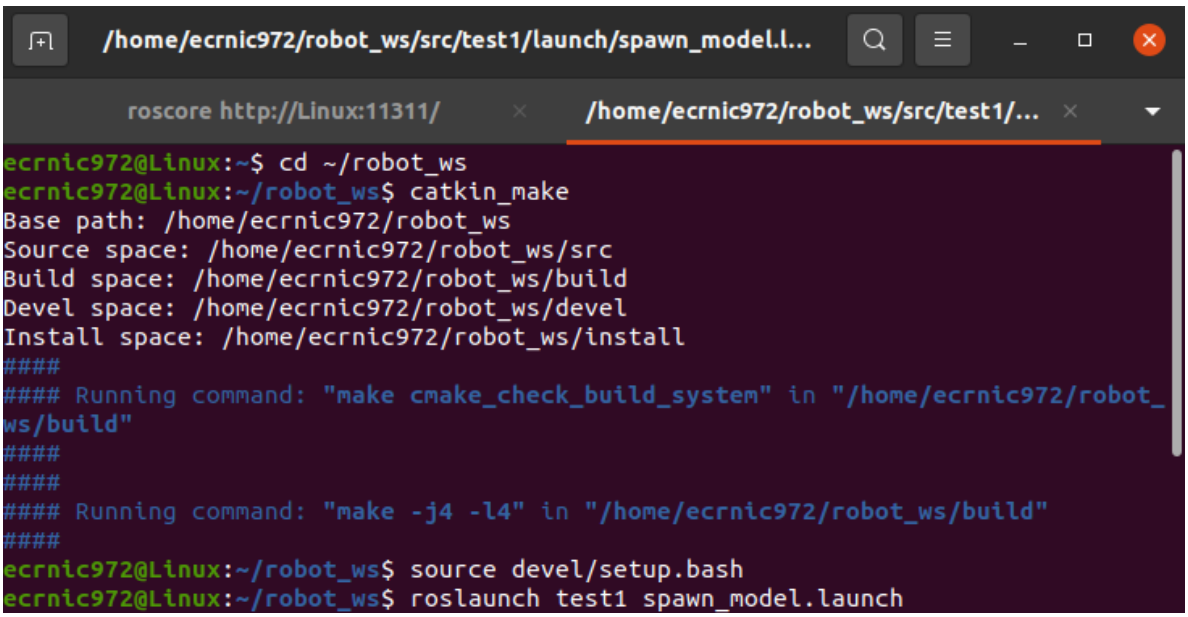
Slika 17. Kreiranje .launch datoteke

Nakon izvršene naredbe prikazuje se tekstualni okvir u koji je potrebno upisati sljedeći kod kako bi otvorili Gazebo simulator s praznim okruženje iz gazebo_ros paketa te kako bi učitali robot u to okruženje:

```
<?xml version="1.0"?>
<launch>
    <include file="$(find gazebo_ros)/launch/empty_world.launch">
        <arg name="gui" value="true"/>
        <arg name="paused" value="true"/>
    </include>

    <param name="robot_description" textfile="$(find robot)/urdf/robot.urdf" />

    <node name="spawn_urdf" pkg="gazebo_ros" type="spawn_model" respawn="false"
args="-param robot_description -urdf -model hobo" />
</launch>
```

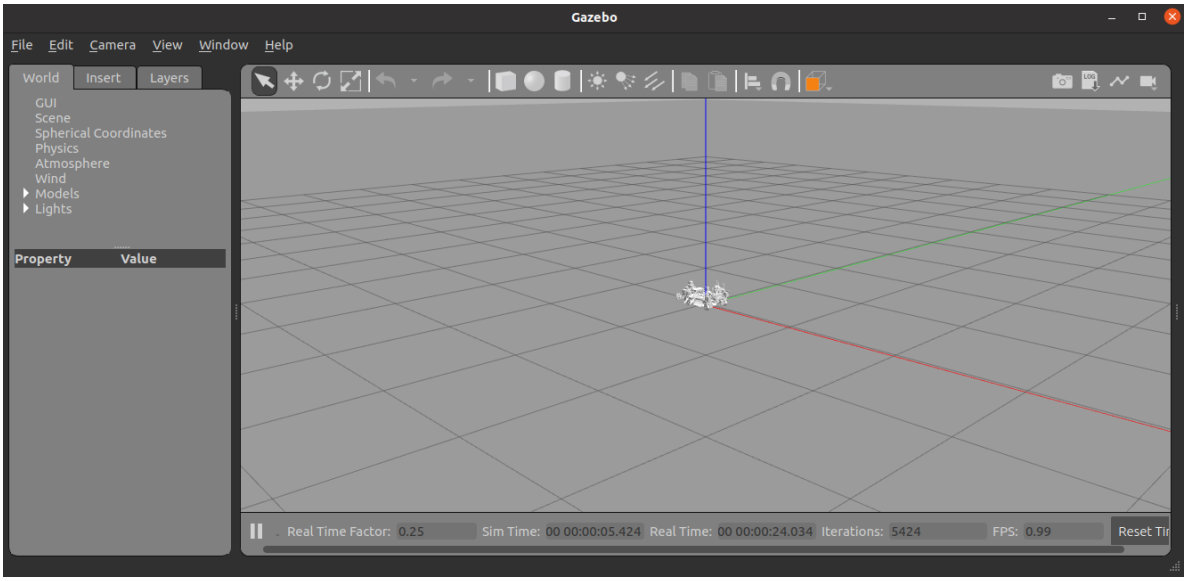
Sljedeći je korak unijeti nekoliko naredbi u Ubuntu terminal kako bismo pokrenuli Gazebo simulator s učitanim modelom robota i okruženjem.

```
cd -/robot_ws

catkin_make

source devel/setup.bash

roslaunch test1 spawn_model.launch
```



Slika 18. Pokretanje Gazebo simulatora

Nakon izvršavanja naredbi prikazuje se Gazebo simulator s učitanim modelom robota i praznim okruženjem.



Slika 19. Gazebo simulator s učitanim modelom robota

5. SIMULACIJA HORIZONTALNIH KRETNJI

5.1. Gazebo plugins

Gazebo dodaci omogućuju URDF modelima veću funkcionalnost i mogu povezati ROS poruke i servisne pozive za izlaz senzora i ulaz motora. Model robota testiran je pomoću dva Gazebo plugina, Thruster plugin te Skid Steering Drive plugin. Kako bismo plugine implementirali u

simulaciju potrebno ih je dodati unutar <robot> elementa u sklopu URDF datoteke. U nastavku je prikazan dio koda koji se odnosi na Skid Steering Drive plugin:

```
<gazebo>
  <plugin name="skid_steer_drive_controller"
filename="libgazebo_ros_skid_steer_drive.so">
    <updateRate>100.0</updateRate>
    <robotNamespace>/</robotNamespace>
    <leftFrontJoint>fl_wheel_joint</leftFrontJoint>
    <rightFrontJoint>fr_wheel_joint</rightFrontJoint>
    <leftRearJoint>rl_wheel_joint</leftRearJoint>
    <rightRearJoint>rr_wheel_joint</rightRearJoint>
    <wheelSeparation>0.4</wheelSeparation>
    <wheelDiameter>0.215</wheelDiameter>
    <robotBaseFrame>base_link</robotBaseFrame>
    <torque>20</torque>
    <topicName>cmd_vel</topicName>
    <broadcastTF>false</broadcastTF>
  </plugin>
</gazebo>
```

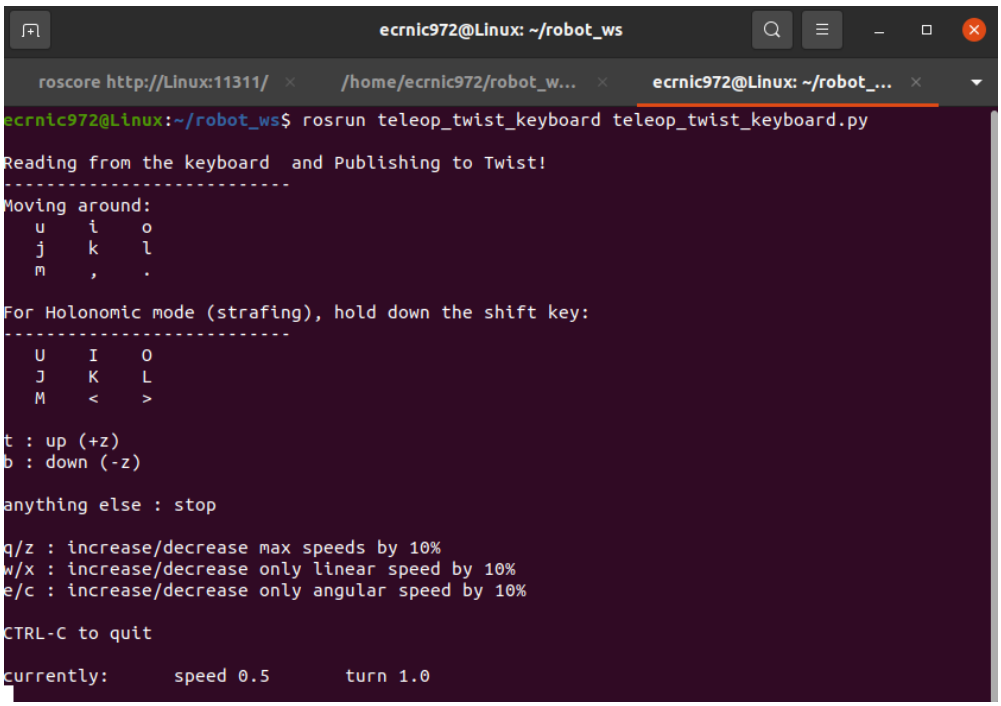
5.2. Teleop_twist_keyboard

Teleop_twist_keyboard je ROS paket za daljinsko upravljanje robotima pomoću računalne tipkovnice. Pojednostavljuje proces slanja Twist poruka za kontrolu kretanja robota, što ga čini vrijednim alatom za razvoj robota, testiranje i otklanjanje pogrešaka. Upravo je taj ROS paket korišten za prvotno testiranje robota po ravnoj podlozi. Instalacija se vrši na jednostavan način:

```
sudo apt-get install ros-noetic-teleop-twist-keyboard
```

Kako bismo provjerili je li paket dobro instaliran potrebno ga je pokrenuti:

```
rosrun teleop_twist_keyboard teleop_twist_keyboard.py
```



Slika 20. Teleop_twist_keyboard paket [18]

Bitno je napomenuti kako je potrebno u Gazebo okruženju pokrenuti simulaciju da bi paket postao dostupan.

5.3. Thruster_controller

Da bismo pokrenuli propeler za adheziju najprije je potrebno učitati još jedan plugin u URDF datoteku.

```
<plugin name="example" filename="libusv_gazebo_thrust_plugin.so">
      <cmdTimeout>1.0</cmdTimeout>
      <thruster>
        <linkName>left_propellers_link</linkName>
        <propJointName>left_propellers_joint</propJointName>
        <engineJointName>left_propellers_joint</engineJointName>
        <cmdTopic>left_thrust_cmd</cmdTopic>
        <angleTopic>left_thrust_angle</angleTopic>
        <enableAngle>false</enableAngle>
        <mappingType>1</mappingType>
        <maxCmd>1.0</maxCmd>
        <maxForceFwd>250.0</maxForceFwd>
        <maxForceRev>-100.0</maxForceRev>
        <maxAngle>1.57</maxAngle>
```

```
      </thruster>
</plugin>
```

Kako bismo pokrenuli propelere za pogon potrebno je kreirati mapu u kojoj će biti smještena python skripta za pokretanje propelera.

```
cd -/robot_ws/src

catkin_create_pkg thruster_controller roscpp
```

Sljedeći je korak kreirati .py datoteku u koju upisujemo skriptu za pokretanje propelera.

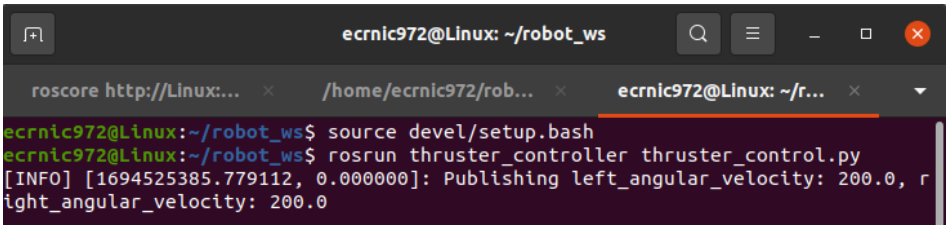
```
cd -/robot_ws/src/thruster_controller/src

gedit thruster_control.py
```

U kreiranu datoteku upisujemo sljedeću skriptu.

```
#!/usr/bin/env python


#!/usr/bin/env python3

import rospy
from std_msgs.msg import Float64

class ThrusterControlNode:
    def __init__(self):
        rospy.init_node('thruster_control_node')

        # Define the joint publishers for left and right thrusters
        self.left_thruster_pub =
rospy.Publisher('/left_propellers_joint/command', Float64, queue_size=1)
        self.right_thruster_pub =
rospy.Publisher('/right_propellers_joint/command', Float64, queue_size=1)

    def run(self):
        # Set a desired angular velocity for the propellers (in radians per
second)
```

```
        left_angular_velocity = 200.0  # Adjust this value to control the left
propeller's angular velocity
        right_angular_velocity = 200.0  # Adjust this value to control the right
propeller's angular velocity

        rospy.loginfo(f"Publishing left_angular_velocity:
{left_angular_velocity}, right_angular_velocity: {right_angular_velocity}")

        # Publish the angular velocity commands to the joint positions
        while not rospy.is_shutdown():
            self.left_thruster_pub.publish(left_angular_velocity)
            self.right_thruster_pub.publish(right_angular_velocity)
            rospy.sleep(0.1)  # Adjust the sleep duration to control the update
rate

if __name__ == '__main__':
    try:
        thruster_control_node = ThrusterControlNode()
        thruster_control_node.run()
    except rospy.ROSInterruptException:
        pass
```

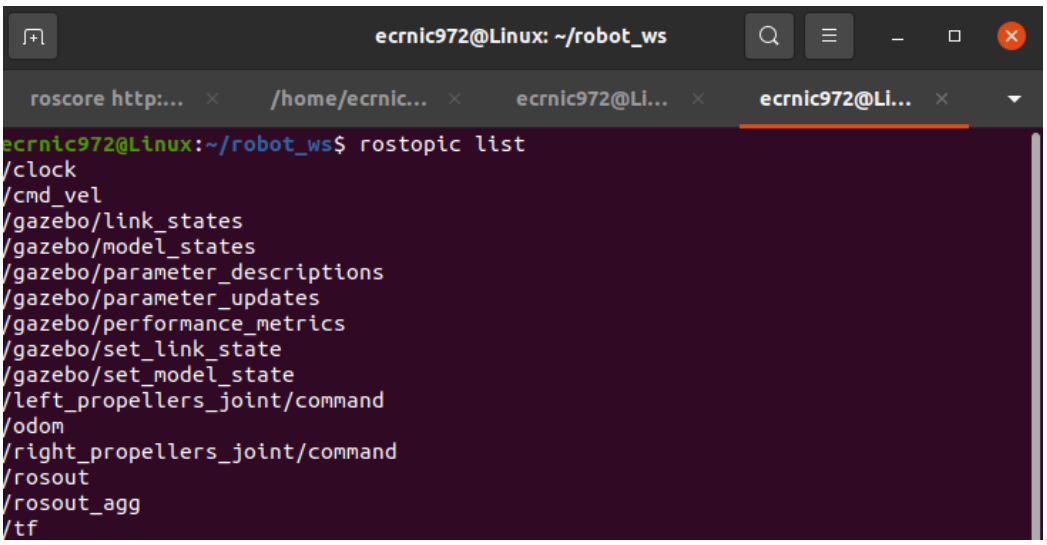
Kako bismo pokrenuli kreiranu skriptu, potrebno je učitati model robota u Gazebo simulator te u sljedeći terminal upisati naredbu za pokretanje paketa.



Slika 21. Pokretanje thruster_control.py skripte

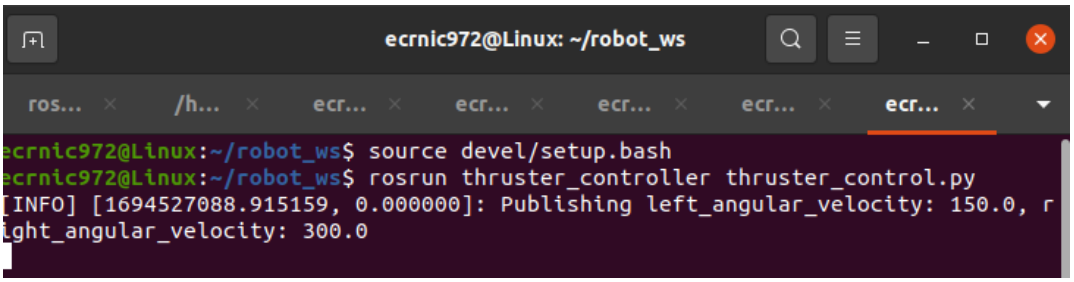
Na slici 21. vidljivo je kako python skripta izdaje zadane vrijednosti kutne brzine lijevog i desnog propelera.

Upisivanjem sljedeće naredbe u terminal moguće je prikazati kako je ROS učitao komande za upravljanje propelerima.

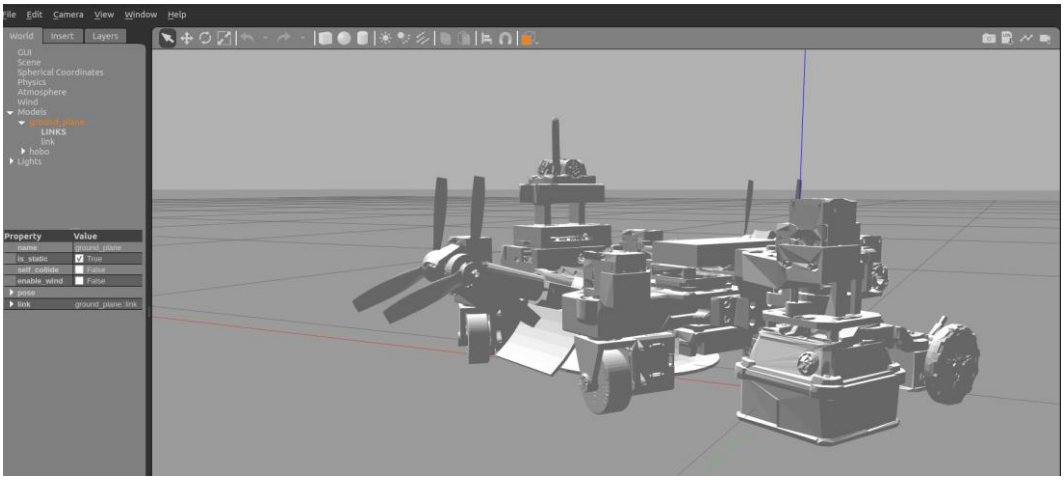


Slika 22. Rostopic list

Ukoliko smanjimo kutnu brzinu desnog propelera, a povećamo kutnu brzinu lijevog može se vidjeti kako robot zakreće.



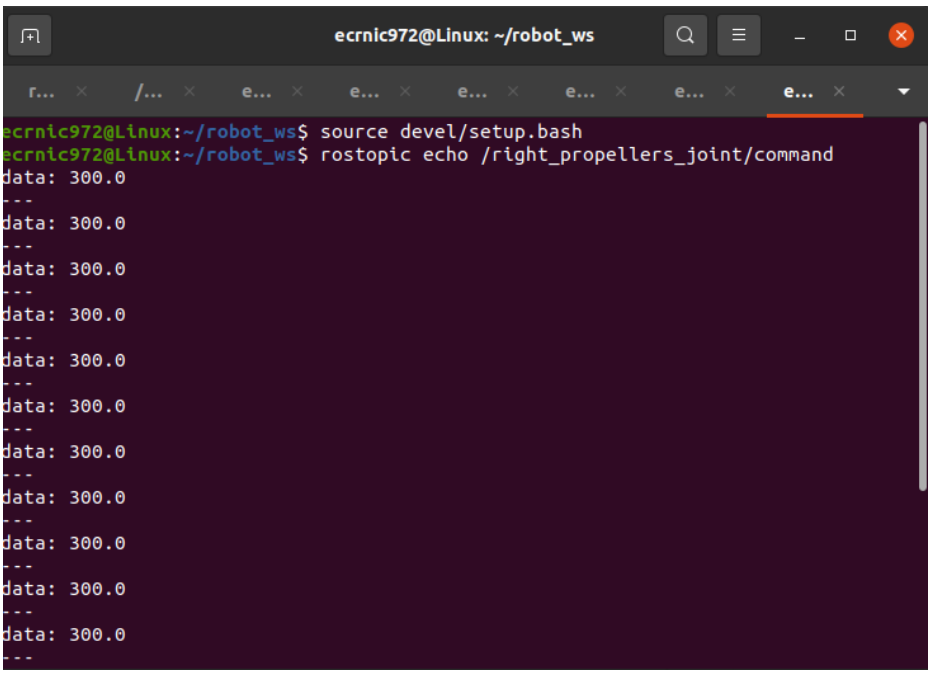
Slika 23. Mjenjanje brzine propelera thruster propelera



Slika 24. Horizontalno pozicioniranje robota

```
rostopic echo /right_propellers_joint/command
```

Također, moguće je provjeriti da se u nekom vremenskom intervalu šalju naredbe o kutnoj brzini, primjerice, desnog propelera.



Slika 25. Rostopic echo desnog propelera

6. SIMULACIJA VERTIKALNIH KRETNJI

6.1. Gazebo plugins

Da bismo pokrenuli propeler za adheziju najprije je potrebno učitati još jedan plugin u URDF datoteku.

```
</plugin>
    <plugin name="example" filename="libusv_gazebo_thrust_plugin.so">
      <cmdTimeout>1.0</cmdTimeout>
      <thruster>
        <linkName>adhesion_propeller_link</linkName>
        <propJointName>adhesion_propeller_joint</propJointName>
        <engineJointName>adhesion_propeller_joint</engineJointName>
        <cmdTopic> thrust_cmd</cmdTopic>
        <angleTopic> thrust_angle</angleTopic>
        <enableAngle>false</enableAngle>
        <mappingType>1</mappingType>
        <maxCmd>1.0</maxCmd>
        <maxForceFwd>250.0</maxForceFwd>
        <maxForceRev>-100.0</maxForceRev>
```

```
        <maxAngle>1.57</maxAngle>
      </thruster>
</plugin>
```

6.2. Adhesion_controller

Kako bismo pokrenuli propeler za adheziju potrebno je kreirati mapu u kojoj će biti smještena python skripta za pokretanje propelera.

```
cd -/robot_ws/src

catkin_create_pkg adhesion_controller roscpp
```

Sljedeći je korak kreirati .py datoteku u koju upisujemo skriptu za pokretanje propelera.

```
cd -/robot_ws/src/adhesion_controller/src

gedit adhesion_controller.py
```

U kreiranu datoteku upisujemo sljedeću skriptu.

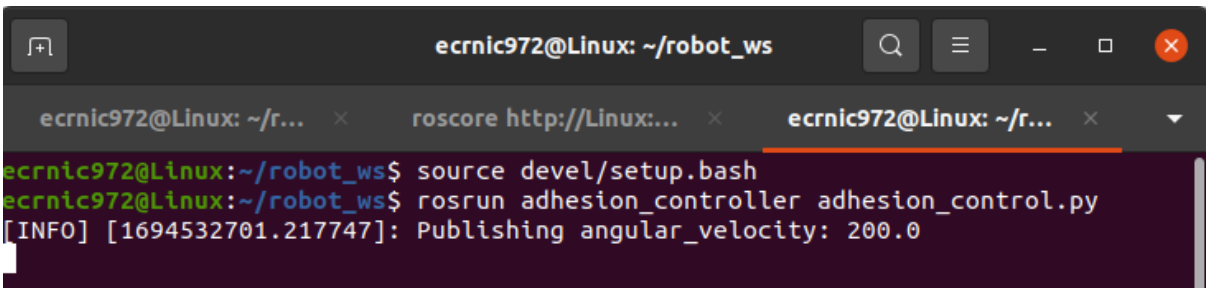
```
#!/usr/bin/env python3

import rospy
from std_msgs.msg import Float64

class ThrusterControlNode:
    def __init__(self):
        rospy.init_node('thruster_control_node')

        # Define the joint publishers for left and right thrusters
        self.thruster_pub = rospy.Publisher('/adhesion_propeller_joint/command',
Float64, queue_size=1)

    def run(self):
        # Set a desired angular velocity for the propeller (in radians per
second)
        angular_velocity = 200.0  # Adjust this value to control the left
propeller's angular velocity

        rospy.loginfo(f"Publishing angular_velocity: {angular_velocity}")
```

```
        # Publish the angular velocity commands to the joint positions
        while not rospy.is_shutdown():
            self.thruster_pub.publish(angular_velocity)
            rospy.sleep(0.1)  # Adjust the sleep duration to control the update
rate

if __name__ == '__main__':
    try:
        thruster_control_node = ThrusterControlNode()
        thruster_control_node.run()
    except rospy.ROSInterruptException:
        pass
```

Kako bismo pokrenuli kreiranu skriptu, potrebno je učitati model robota u Gazebo simulator koristeći tf2 proširenje kako bismo bili u mogućnosti učitati model robota pod određenim kutem. U nastavku je prikazana modificirana .launch datoteka u kojoj je potrebno namjestiti vrijednosti rotacije.

```
<?xml version="1.0"?>
<launch>
    <!-- Include Gazebo world launch -->
    <include file="$(find gazebo_ros)/launch/empty_world.launch">
        <arg name="gui" value="true"/>
        <arg name="paused" value="true"/>
    </include>

    <!-- Load robot description -->
    <param name="robot_description" textfile="$(find test1)/urdf/test1.urdf" />

    <!-- Spawn the robot model in Gazebo -->
    <node name="spawn_urdf" pkg="gazebo_ros" type="spawn_model" respawn="false"
args="-param robot_description -urdf -model hobo" />

    <!-- Rotate the robot model -->
    <node name="rotate_robot" pkg="tf2_ros" type="static_transform_publisher"
args="0 0 0 0 0 0 /hobo/base_link /hobo/base_link_rotated 100" />
</launch>
```

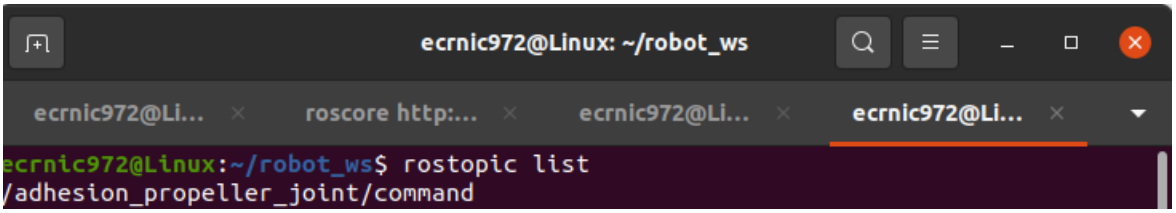
Kako bismo pokrenuli kreiranu .py skriptu, potrebno je učitati model robota u Gazebo simulator te u sljedeći terminal upisati naredbu za pokretanje paketa.



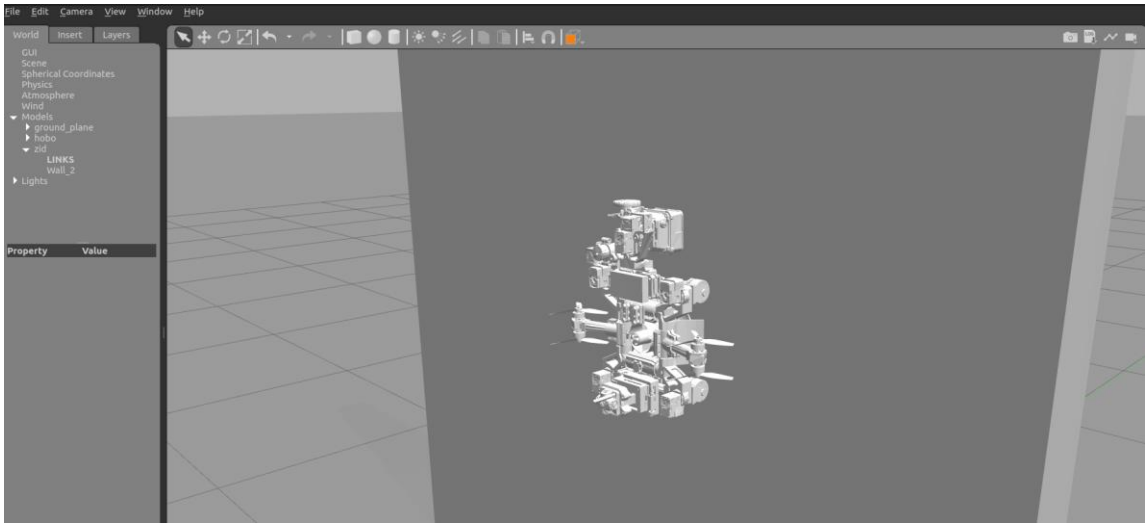
Slika 26. Pokretanje adhesion_control.py skripte

Na slici 21. vidljivo je kako python skripta izdaje zadane vrijednosti kutne brzine adhezijskog propelera.

Upisivanjem sljedeće naredbe u terminal moguće je prikazati kako je ROS učitao komande za upravljanje propelerom.



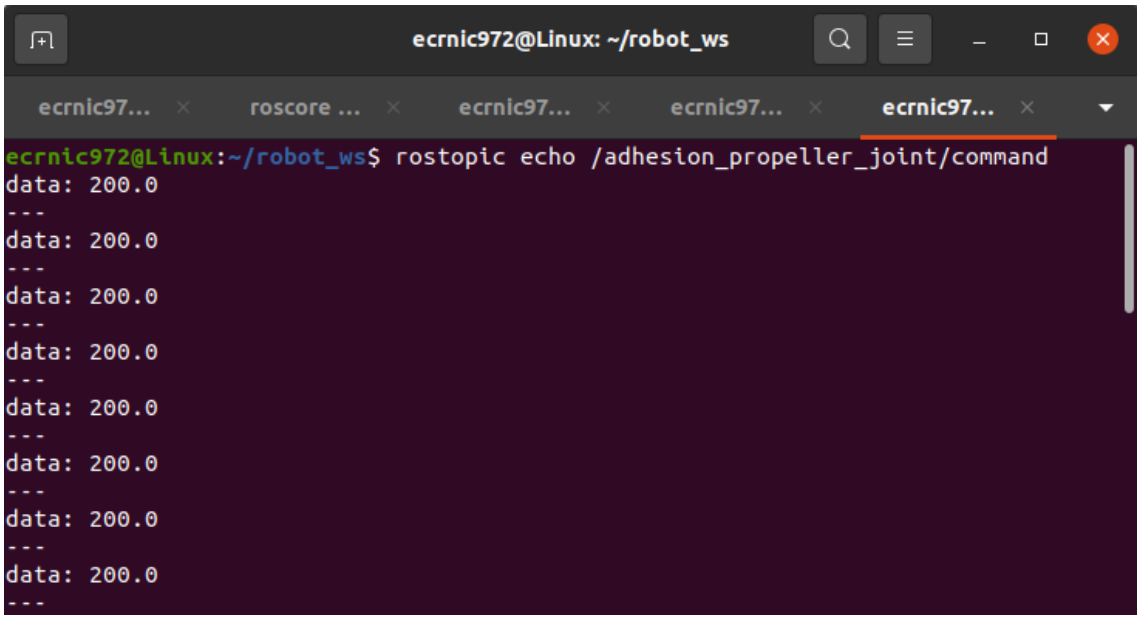
Slika 27. Rostopic list



Slika 28. Vertikalno pozicioniranje robota

Također, moguće je provjeriti da se u nekom vremenskom intervalu šalju naredbe o kutnoj brzini adhezijskog propelera.

```
rostopic echo /adhesion_propeller_joint/command
```



Slika 29. Rostopic echo adhezijskog propelera

7. ZAKLJUČAK

Rad naglašava značajnu ulogu moderne tehnologije u revolucioniranju procesa testiranja korištenjem dostupnih softvera i alata. Primarni fokus bio je na simulaciji sposobnosti robota penjača na vodoravnim i okomitim površinama, što je zahtijevalo razumijevanje samog robota, njegove kinematike i bitnih koraka do simulacije. Simulacija je izvedena na Ubuntu 20.04 operativnom sustavu koristeći Robot Operating System (ROS) za kontrolu robota unutar Gazebo simulatora. Proces je uključivao stvaranje URDF datoteke i drugih bitnih datoteka ključnih za simulaciju. Pružajući uvid u tehnike i tehnologije rad prikazuje napredak robotike i automatizacije omogućujući inženjerima da razviju i testiraju robote na isplativ, jednostavan i siguran način. Također, rad predstavlja korelaciju tehnologije s robotikom uz nove mogućnosti za razvoj robota i robotskih aplikacija.

LITERATURA

[1] https://www.techtarget.com/iotagenda/definition/mobile-robot-mobile-robotics

[2] https://www.researchgate.net/publication/273310520_Kinematics_dynamics_and_control_design_of_4WIS4WID_mobile_robots

[3] https://www.virtualbox.org/

[4] https://ubuntu.com/

[5] https://hr.wikipedia.org/wiki/Ubuntu

[6] https://en.wikipedia.org/wiki/Robot_Operating_System

[7] https://gazebosim.org/home

[8] https://formant.io/urdf

[9] http://wiki.ros.org/urdf

[10] https://www.ros.org/blog/why-ros/

[11] http://wiki.ros.org/ROS/Tutorials

[12] http://wiki.ros.org/sw_urdf_exporter

[13] https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/joe.2014.0241

[14] http://wiki.ros.org/catkin/Tutorials/create_a_workspace

[15] http://wiki.ros.org/rviz

[16] https://gkjohnson.github.io/urdf-loaders/javascript/example/bundle/index.html

[17] https://classic.gazebosim.org/tutorials?tut=ros_gzplugins

[18] http://wiki.ros.org/teleop_twist_keyboard

[19] https://github.com/ros-teleop/teleop_twist_keyboard

[20] http://docs.ros.org/en/melodic/api/usv_gazebo_plugins/html/classgazebo_1_1Thruster.html

PRILOZI

I. https://github.com/ecrnic972/4WID4WIS_robot_codes/blob/main/robot.urdf

II. https://github.com/ecrnic972/4WID4WIS_robot_codes/blob/main/thruster_control.py

III. https://github.com/ecrnic972/4WID4WIS_robot_codes/blob/main/adhesion_control.py