

Technical Report UAS Robotika
Mastering ROS



Oleh:

Nama : Arfiq Rimeldo

NIM : 1103202102

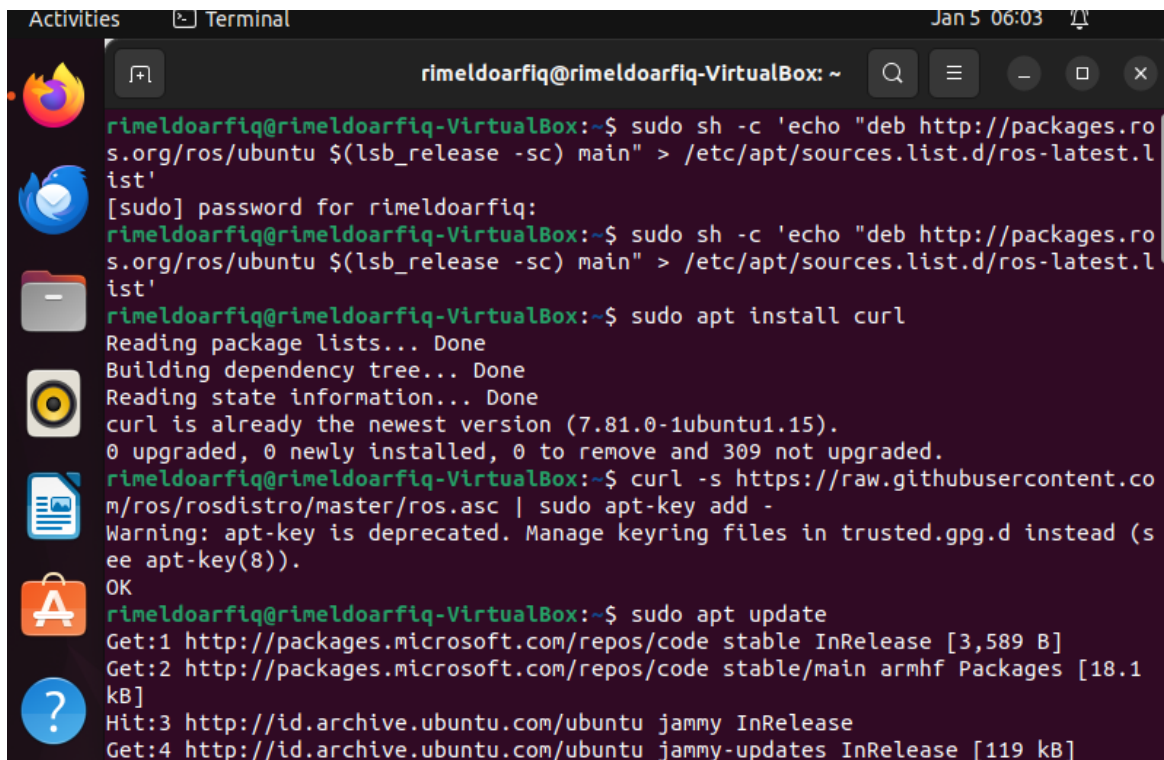
PROGRAM STUDI S1 TEKNIK KOMPUTER
FAKULTAS TEKNIK ELEKTRO
UNIVERSITAS TELKOM
2024

Chapter 1

Robot Operating System (ROS) adalah kerangka kerja yang fleksibel yang menyediakan berbagai alat dan pustaka untuk menulis perangkat lunak robot. Ini menawarkan beberapa fitur kuat untuk membantu pengembang dalam tugas seperti pertukaran pesan, komputasi terdistribusi, penggunaan kode, dan implementasi algoritma terkini untuk aplikasi robotik. Di sini VirtualBox dipersiapkan terlebih dahulu untuk menjalankan Linux. Kemudian dipersiapkan juga Ubuntu 22.04 baru dilanjutkan dengan menginstall ROS2.

Langkah-langkah pada chapter 1:

1. Melakukan instalasi ROS Noetic menggunakan panduan pada <https://wiki.ros.org/noetic/Installation/Ubuntu> untuk memudahkan dalam proses penginstalan.
2. Setelah install selesai dilanjutkan dengan melakukan setup pada source dan key. Dari gambar di atas dapat dilihat proses penambahan repository ROS (Robot Operating System) ke daftar sumber perangkat lunak di sistem operasi berbasis Ubuntu dan proses



```
rimeldoarfiq@rimeldoarfiq-VirtualBox: ~  
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'  
[sudo] password for rimeldoarfiq:  
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'  
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ sudo apt install curl  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
curl is already the newest version (7.81.0-1ubuntu1.15).  
0 upgraded, 0 newly installed, 0 to remove and 309 not upgraded.  
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -  
Warning: apt-key is deprecated. Manage keyring files in trusted.gpg.d instead (see apt-key(8)).  
OK  
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ sudo apt update  
Get:1 http://packages.microsoft.com/repos/code stable InRelease [3,589 B]  
Get:2 http://packages.microsoft.com/repos/code stable/main armhf Packages [18.1 kB]  
Hit:3 http://id.archive.ubuntu.com/ubuntu jammy InRelease  
Get:4 http://id.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
```

mengunduh kunci publik (public key) dari repository ROS dan menambahkannya ke dalam keyring APT di sistem operasi Ubuntu. Kemudian melakukan update untuk mengupdate ROS.

3. Setelah semua berhasil diupdate maka dapat dilanjutkan dengan proses instalasi ROS Noetic.

```
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ sudo apt install ros-noetic-desktop-full
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

4. Setelah ROS Noetic terinstall maka langkah selanjutnya adalah untuk melakukan setup environment

```
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ Ssource /opt/ros/noetic/setup.bash
```

Setelah environment berhasil disetup, ROS Noetic dapat diprogram.

5. Terakhir mencoba menggunakan command 'roscore'

```
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ roscore
```

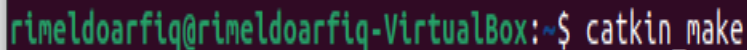
.Kegunaan dari command 'roscore' tersebut adalah:

- a) **Master Node:** **roscore** adalah master node pada sistem ROS. Ini berarti bahwa ini adalah entitas pusat yang mengelola pendaftaran (registration) dan koneksi antara komponen-komponen ROS lainnya.
- b) **Parameter Server:** **roscore** menyediakan parameter server, yang memungkinkan berbagai node ROS untuk menyimpan dan mengambil parameter-parameter yang diperlukan selama waktu eksekusi.
- c) **ROS Communication:** **roscore** menyediakan layanan untuk komunikasi antara node-node di dalam sistem ROS. Ini termasuk layanan publikasi dan langganan, serta topik-topik yang digunakan untuk pertukaran pesan antar node.
- d) **Name Service:** **roscore** menyediakan layanan pencarian nama (name service), yang memungkinkan node untuk menemukan satu sama lain dan berkomunikasi secara dinamis.
- e) **Time Server:** **roscore** berfungsi sebagai sumber waktu untuk sinkronisasi waktu di seluruh sistem ROS, memastikan koordinasi yang tepat antara node-node yang beroperasi secara bersamaan.

Chapter 2

Langkah pertama dalam chapter 2 ini yaitu mengenai ROS package. ROS package yaitu organisasi dari pekerjaan ROS. Package ini dapat berisi kode sumber, pustaka, konfigurasi, data, dan file-file lain yang diperlukan untuk mengimplementasikan suatu fungsi atau fitur tertentu dalam ekosistem ROS. Setiap ROS package memiliki struktur direktori yang ditentukan, dan mereka memungkinkan pengembang untuk mengorganisasi dan membagi kode mereka dengan cara yang terstruktur. Langkah-langkah yang dilakukan antara lain:

1. Membuat workspace ROS
`mkdir -p catkin_ws/src`
`cd catkin_ws/src`
2. Membuat ROS package baru
`cd catkin_ws/src`
`catkin_create_pkg`
3. Menambahkan file dan kode sumber
4. Membangun workspace menggunakan catkin
5. Build package
`cd catkin_ws`
`catkin_make`



```
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ catkin_make
```

Selain ROS Package, chapter 2 ini juga membahas mengenai nodes. Nodes adalah entitas dasar yang menjalankan tugas tertentu di dalam sistem robotika. Node adalah program yang dirancang untuk berkomunikasi dengan node lainnya melalui model komunikasi publish-subscribe, layanan (service), atau mekanisme lain yang disediakan oleh ROS. Beberapa karakteristik dari nodes di ROS:

1. **Independen:** Setiap node dijalankan sebagai proses independen. Ini berarti bahwa satu node dapat dijalankan atau dimatikan tanpa mempengaruhi node lainnya.
2. **Komunikasi:** Nodes berkomunikasi satu sama lain melalui topik (topics), layanan (services), atau menggunakan mekanisme komunikasi ROS lainnya. Mereka dapat mengirim dan menerima pesan untuk berbagi informasi.
3. **Spesifik Fungsi:** Setiap node biasanya ditujukan untuk melaksanakan fungsi atau tugas tertentu dalam sistem robotika. Misalnya, sebuah node mungkin bertanggung jawab untuk membaca data dari sensor, sedangkan node lainnya dapat mengendalikan aktuator.
4. **Modularitas:** Desain sistem di ROS cenderung modular, di mana setiap tugas atau fungsionalitas dapat diimplementasikan dalam satu atau beberapa nodes terpisah. Ini mempermudah pengembangan, pemeliharaan, dan integrasi komponen-komponen sistem.
5. **Language Independence:** ROS mendukung berbagai bahasa pemrograman seperti C++, Python, dan lainnya. Ini memungkinkan pengembang untuk menulis nodes dalam bahasa yang paling nyaman bagi mereka.

Langkah-langkah yang digunakan dalam pembuatan nodes ini antara lain:

1. Persiapkan workspace lagi seperti tadi.
2. Buat ROS package
3. Mulai membuat node
4. Menggunakan command 'roslaunch mastering_ros_demo_pkg demo_topic_publisher'

```
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ roslaunch mastering_ros_demo_pkg demo_topic_publisher
```

Digunakan untuk menjalankan node atau program dengan nama **demo_topic_publisher** dari ROS package yang disebut **mastering_ros_demo_pkg**

5. Menggunakan command 'roslaunch mastering_ros_demo_pkg demo_service_client'

```
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ roslaunch mastering_ros_demo_pkg demo_service_client
```

Digunakan untuk menjalankan node atau program yang disebut **demo_service_client** dari ROS package dengan nama **mastering_ros_demo_pkg**.

Chapter 3

ROS dapat digunakan dalam konteks pengembangan aplikasi robotika yang melibatkan pemodelan dan pemrosesan data 3D, diantaranya:

1. Pengolahan Data Sensor 3D: ROS mendukung integrasi dengan sensor-sensor 3D seperti lidar, kinect, atau kamera stereovision. Data yang dihasilkan dari sensor-sensor ini dapat diakses, diolah, dan dimanipulasi menggunakan nodes ROS. Sebagai contoh, ROS memiliki paket-paket seperti `pcl_ros` (Point Cloud Library for ROS) yang memungkinkan pemrosesan data titik awan (point cloud) dalam aplikasi 3D modeling.
2. Penggunaan Simulator 3D: Ada simulator 3D yang kompatibel dengan ROS, seperti Gazebo. Gazebo memungkinkan simulasi robotika di lingkungan 3D yang realistis. Pengguna dapat mengembangkan dan menguji perangkat lunak robot di simulator ini sebelum menerapkannya pada perangkat keras fisik.
3. Paket untuk Manipulasi 3D: Beberapa paket ROS mendukung manipulasi objek atau model 3D. Sebagai contoh, ROS memiliki paket seperti `moveit` yang dirancang untuk perencanaan pergerakan robot dan manipulasi objek di ruang 3D.
4. Pemodelan Objek 3D: Anda dapat menggunakan alat pemodelan 3D, seperti Blender atau CAD tools, dan mengintegrasikannya dengan ROS untuk mengembangkan dan merencanakan aplikasi robotika yang melibatkan objek atau lingkungan 3D.
5. Komunikasi dengan Software 3D Modeling: ROS dapat berkomunikasi dengan perangkat lunak pemodelan 3D atau aplikasi lainnya. Ini memungkinkan pengembang untuk memanfaatkan data yang dihasilkan oleh perangkat lunak pemodelan 3D atau membuat algoritma pengolahan data tambahan.

Pada chapter 3 kali ini ada langkah-langkah yang digunakan dalam 3D modeling menggunakan ROS, yaitu:

1. Membuat workspace
2. Membuat ROS Package
3. Membuat urdf file

URDF (Unified Robot Description Format) adalah format file XML yang digunakan untuk mendeskripsikan model robot dalam Robot Operating System (ROS). File URDF memberikan informasi yang diperlukan tentang struktur mekanis, kinematika, dan visualisasi robot.

4. Source workspace yang telah dibuat tadi.
5. Memulai membuat model robot menggunakan RViz

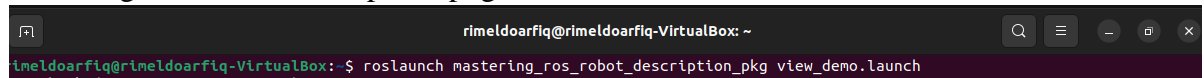
RViz (Robot Visualization) adalah alat visualisasi yang sangat berguna dalam Robot Operating System (ROS). RViz memungkinkan pengguna untuk memvisualisasikan data dari berbagai sensor, model robot, serta informasi lainnya dalam lingkungan robotika. Ini membantu pengembang dan insinyur dalam memahami dan memeriksa kinerja robot. Beberapa fitur dari RViz antara lain:

- a) Visualisasi Robot Model: RViz memungkinkan pengguna untuk memuat dan menampilkan model robot dalam format URDF. Dengan cara ini,

pengguna dapat melihat secara real-time posisi, orientasi, dan konfigurasi sendi dari robot.

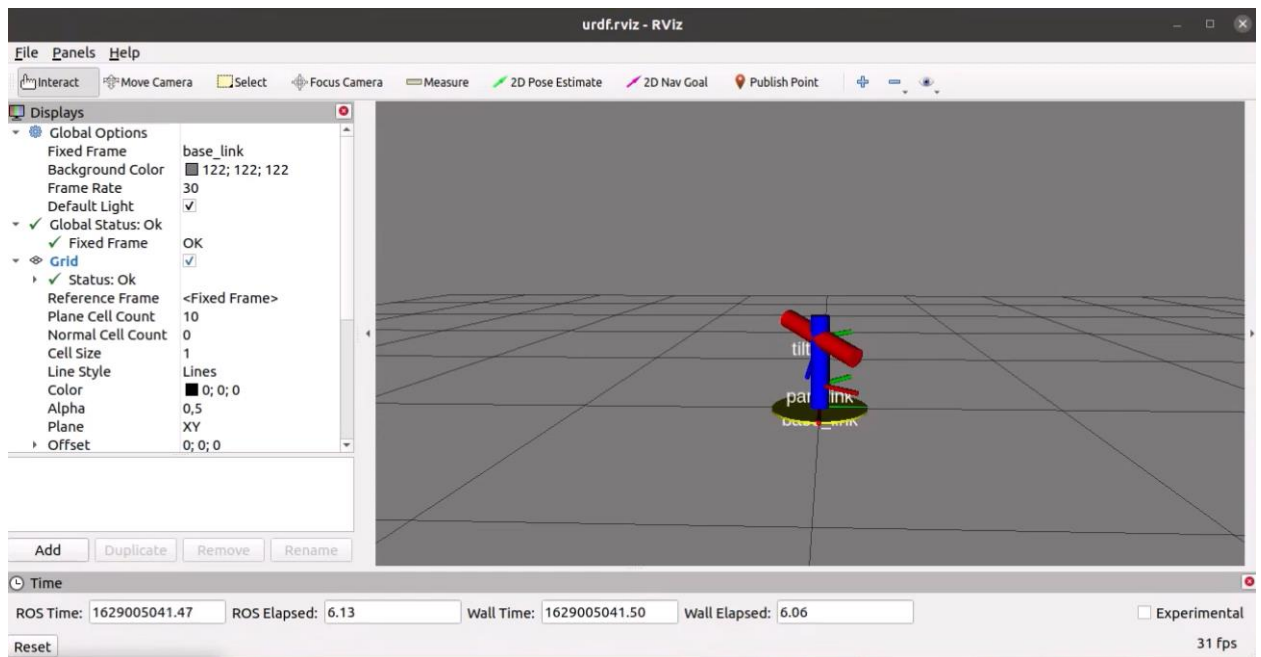
- b) Visualisasi Sensor Data: RViz mendukung berbagai jenis sensor, seperti data dari lidar, kamera, dan data titik awan (point cloud). Hal ini memungkinkan pengguna untuk memeriksa dan menganalisis data sensor yang dihasilkan oleh robot.
- c) Visualisasi Trajectory dan Pergerakan Robot: RViz memungkinkan visualisasi trajectory pergerakan robot dan memungkinkan pengguna untuk merencanakan dan memeriksa pergerakan robot dalam simulasi.
- d) Pemetaan dan Pemahaman Lingkungan: Dengan menggunakan RViz, pengguna dapat memvisualisasikan pemetaan (mapping) lingkungan yang dibuat oleh robot menggunakan sensor seperti lidar atau kinect. Ini membantu untuk memahami dan memeriksa kualitas pemetaan robot.
- e) Debugging dan Analisis: RViz menyediakan alat untuk memecahkan masalah dan menganalisis kinerja robot, membantu dalam proses debugging selama pengembangan perangkat lunak.
- f) Konfigurasi Interaktif: RViz memungkinkan pengguna untuk mengonfigurasi tampilan dengan cara yang interaktif. Ini termasuk menyesuaikan properti visualisasi, memilih bagian robot yang ingin dilihat, atau mengatur tampilan data sensor.
- g) Simulasi di Gazebo: RViz dapat diintegrasikan dengan simulator Gazebo untuk menyediakan visualisasi dari simulasi robot yang lebih kompleks, termasuk lingkungan 3D yang realistis.

6. Memulai dengan menggunakan command 'roslaunch mastering_ros_robot_description_pkg view_demo.launch'



```
rimeldoarfiq@rimeldoarfiq-VirtualBox: ~  
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ roslaunch mastering_ros_robot_description_pkg view_demo.launch
```

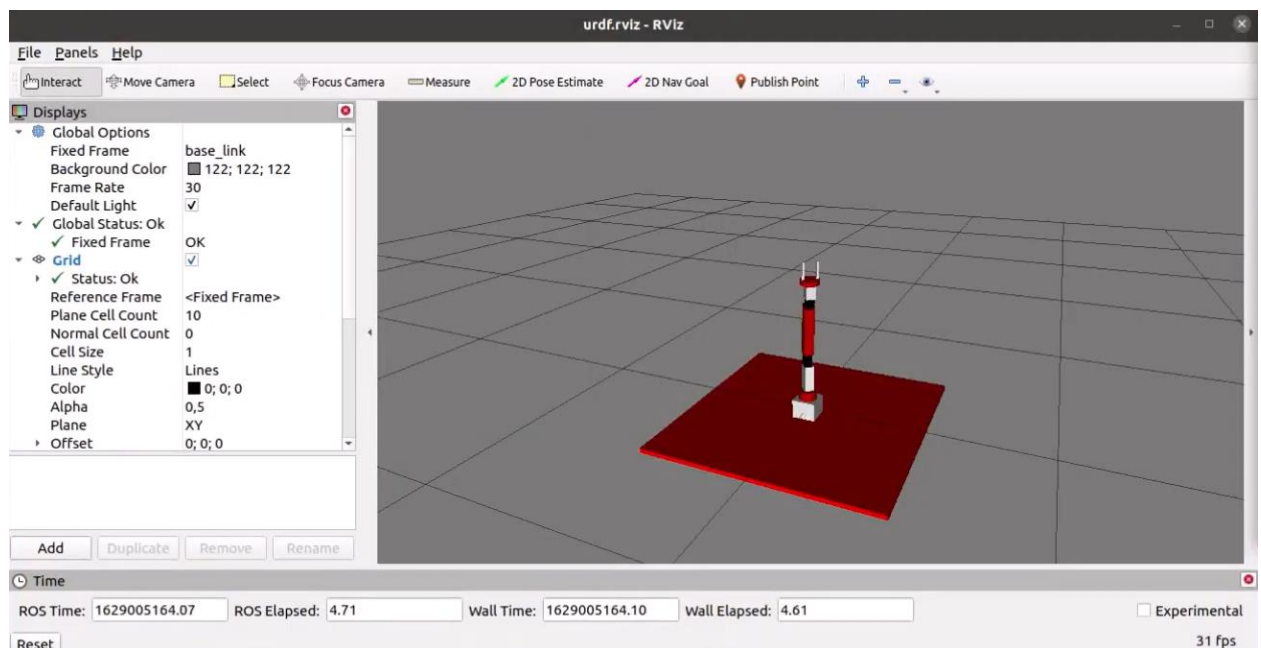
7. Setelah berhasil dilaunch maka akan muncul tampilan seperti gambar berikut:



8. Langkah selanjutnya menggunakan command 'roslaunch mastering_ros_robot_description_pkg view_arm.launch'

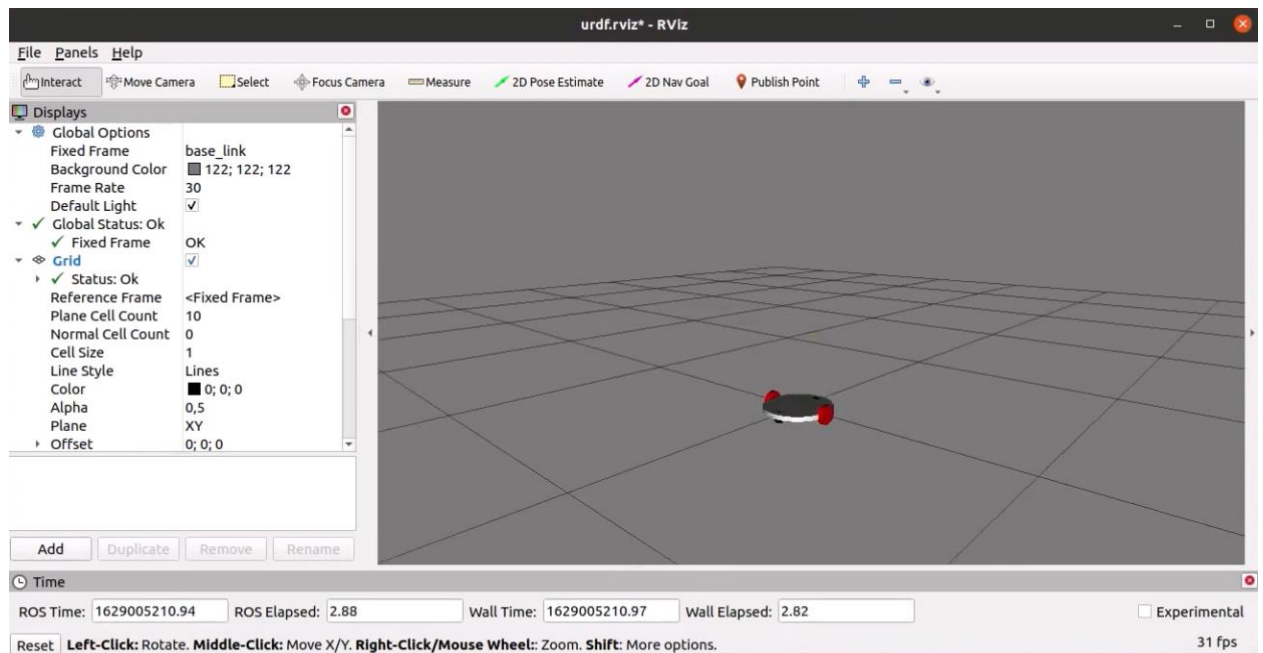
```
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ roslaunch mastering_ros_robot_description_pkg view_arm.launch
```

9. Setelah command berhasil maka akan muncul tampilan:



10. Langkah terakhir menggunakan command 'roslaunch mastering_ros_robot_description_pkg view_mobile_robot.launch'

11. Setelah command dimasukkan maka akan muncul tampilan:



Chapter 4

Simulasi robot menggunakan ROS (Robot Operating System) dan Gazebo merupakan pendekatan umum dan sangat efektif dalam pengembangan robotika. Gazebo adalah simulator fisika 3D yang menyediakan lingkungan simulasi realistis untuk sistem robotik. Ketika diintegrasikan dengan ROS, pengembang dapat mensimulasikan robot, menguji algoritma, dan memvalidasi strategi kontrol dalam lingkungan virtual sebelum mengimplementasikannya pada robot fisik. Gambaran umum mengenai simulasi ROS dan Gazebo antara lain:

1. Integrasi ROS dan Gazebo:
ROS dan Gazebo sering digunakan bersama untuk menciptakan lingkungan simulasi yang komprehensif. ROS menyediakan middleware untuk komunikasi antar komponen sistem robot, sedangkan Gazebo menangani simulasi fisika robot dan lingkungannya.
2. Membuat Model URDF:
URDF (Unified Robot Description Format) umum digunakan untuk mendeskripsikan model robot dalam ROS. Struktur fisik robot, konfigurasi sendi, dan representasi visualnya dijelaskan dalam file URDF.
File URDF dapat dibuat secara manual atau dihasilkan menggunakan alat seperti Xacro, yang memungkinkan deskripsi URDF yang terparameterisasi.
3. Integrasi Plugin Gazebo:
Plugin Gazebo dapat digunakan untuk mensimulasikan sensor, aktuator, dan komponen lain dari robot. Sebagai contoh, plugin sensor kamera dalam Gazebo dapat mensimulasikan output kamera robot.
Paket ROS sering menyediakan plugin Gazebo untuk memudahkan integrasi komponen-komponen ROS ke dalam lingkungan simulasi.
4. File Launch untuk Simulasi:
File launch dalam ROS digunakan untuk menyiapkan dan mengonfigurasi berbagai node dan komponen yang diperlukan untuk simulasi.
Sebuah file launch mungkin mencakup konfigurasi untuk menyematkan model robot di Gazebo, memuat pengontrol yang diperlukan, dan menjalankan node untuk komunikasi antara ROS dan Gazebo.
5. Konfigurasi Pengontrol:
Pengontrol untuk sendi dan komponen robot dikonfigurasi untuk mensimulasikan pergerakan dan dinamika yang realistis.
ROS Control adalah paket yang umumnya digunakan untuk mengonfigurasi dan mengontrol sendi dan aktuator robot dalam simulasi.
6. Simulasi Sensor:
Sensor seperti kamera, lidar, dan IMU umumnya disimulasikan di Gazebo. Plugin Gazebo dapat digunakan untuk meniru output sensor dan menerbitkan data ke topik ROS.
Data sensor dapat divisualisasikan di RViz atau digunakan oleh node ROS untuk algoritma persepsi atau kontrol.
7. Topik dan Layanan ROS:

Selama simulasi, node-node ROS berkomunikasi satu sama lain menggunakan topik dan layanan, sama seperti dalam skenario dunia nyata.

Pengembang dapat menguji dan mendebug node-node dan algoritma ROS mereka dalam lingkungan simulasi sebelum mengimplementasikannya pada perangkat keras robot.

8. Visualisasi Hasil Simulasi:

RViz, alat visualisasi dalam ROS, dapat digunakan untuk memonitor dan visualisasi keadaan robot, data sensor, dan informasi relevan lainnya selama simulasi.

9. Debugging dan Pengujian:

Lingkungan simulasi menyediakan lingkungan yang terkendali dan dapat diulang untuk debugging dan pengujian sistem robotik. Pengembang dapat bereksperimen dengan berbagai skenario dan kasus ekstrim untuk memvalidasi keandalan algoritma mereka.

10. Pengujian Berbasis Skenario:

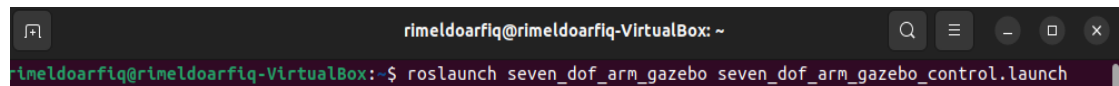
Gazebo memungkinkan pengembang untuk mensimulasikan berbagai skenario, termasuk lingkungan kompleks, hambatan dinamis, dan kondisi yang sulit. Ini membantu dalam menguji kemampuan robot mengatasi berbagai situasi.

11. Penyetelan Parameter:

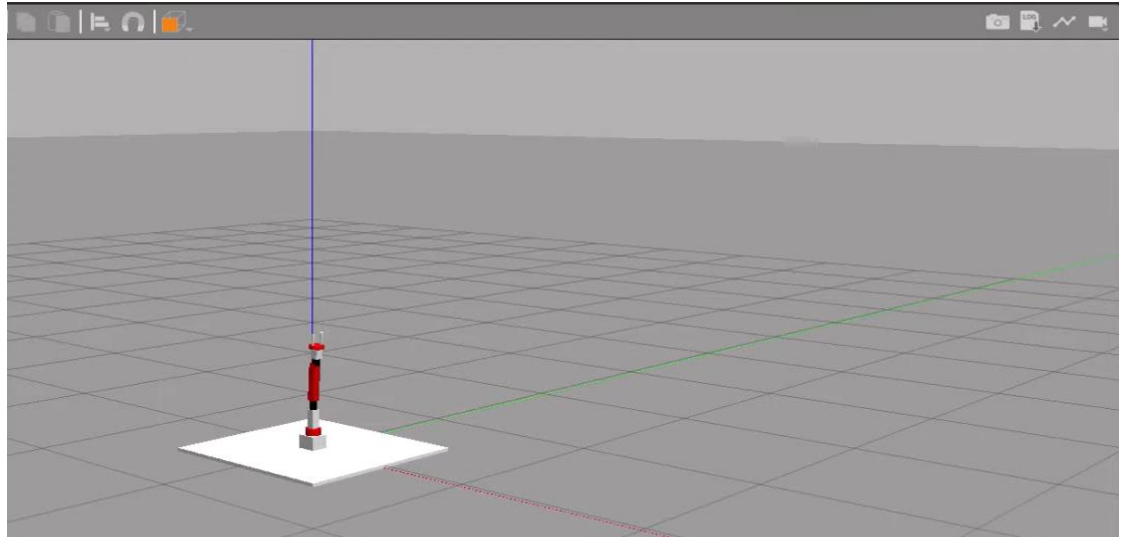
Simulasi memberikan platform untuk menyetel dan mengoptimalkan parameter kontrol tanpa risiko merusak perangkat keras fisik.

Langkah-langkah yang dilakuakn dalam chapter 4 ini antara lain:

1. Buat workspace pada ROS
2. Mempersiapkan package ROS untuk robot
3. Mempersiapkan model URDF
4. Mulai menjalankan gazebo
5. Mulai dengan menggunakan command 'roslaunch seven_dof_arm_gazebo seven_dof_arm_world.launch'



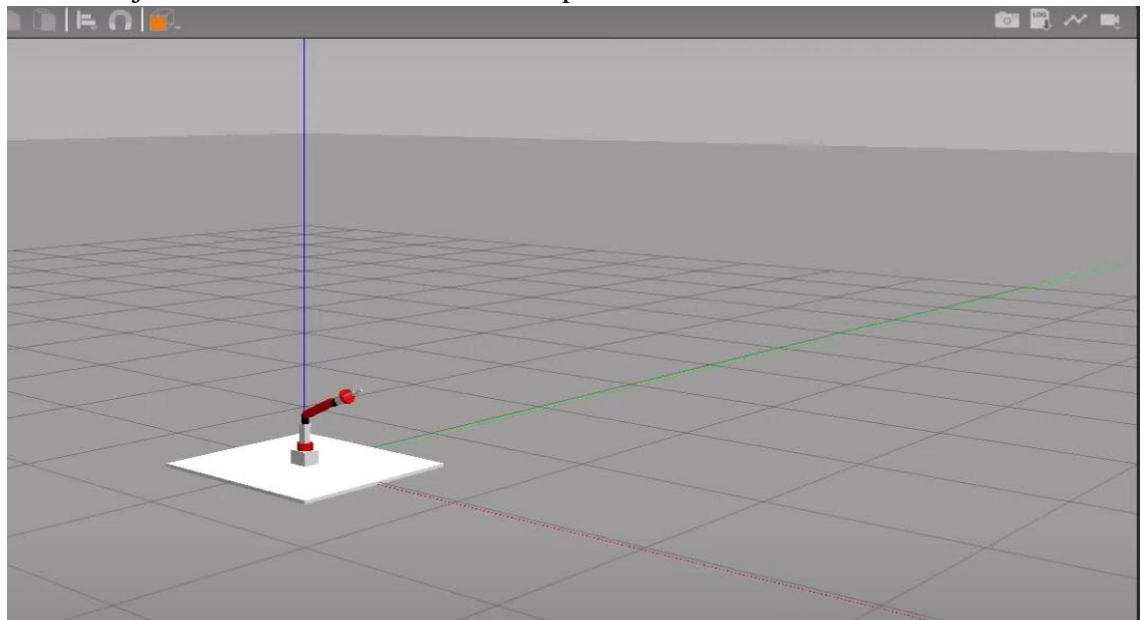
6. Setelah command dijalankan maka akan muncul tampilan seperti gambar berikut:



7. Selanjutnya menggunakan command 'rostopic pub /seven_dof_arm/joint4_position_controller/command std_msgs/Float64 "data: 1.0"'

```
rimeldoarfiq@rimeldoarfiq-VirtualBox: ~
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ rostopic pub /seven_dof_arm/joint4_position_controller/command std_msgs/Float64 "data: 1.0"
```

8. Setelah dijalankan maka akan muncul tampilan:

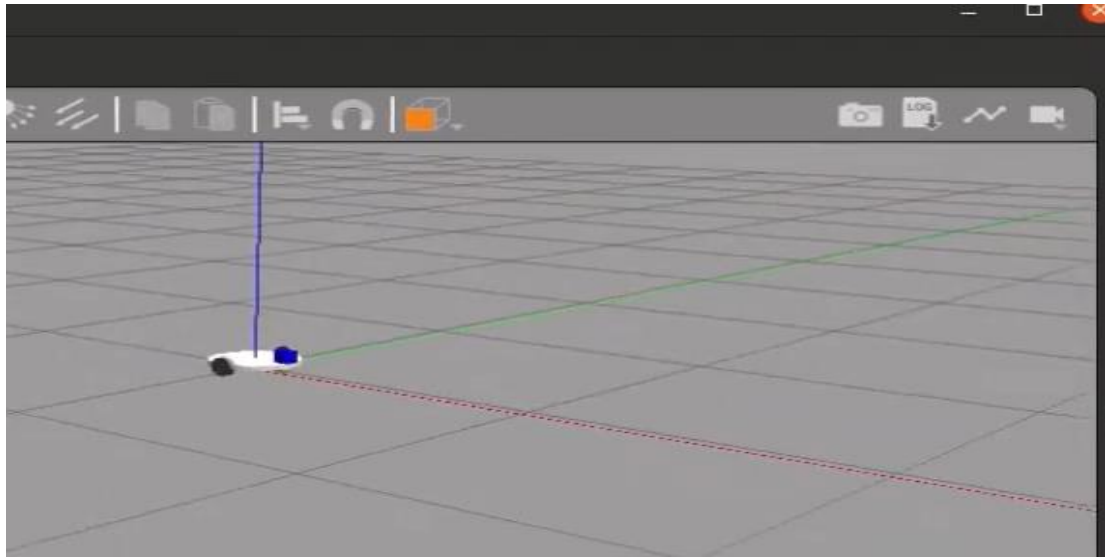


Gambar di atas dapat dilihat bahwa robot dapat bergerak secara otomatis.

9. Command selanjutnya yaitu 'roslaunch diff_wheeled_robot_gazebo diff_wheeled_gazebo.launch'

```
rimeldoarfiq@rimeldoarfiq-VirtualBox: ~
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ roslaunch diff_wheeled_robot_gazebo diff_wheeled_gazebo_full.launch
```

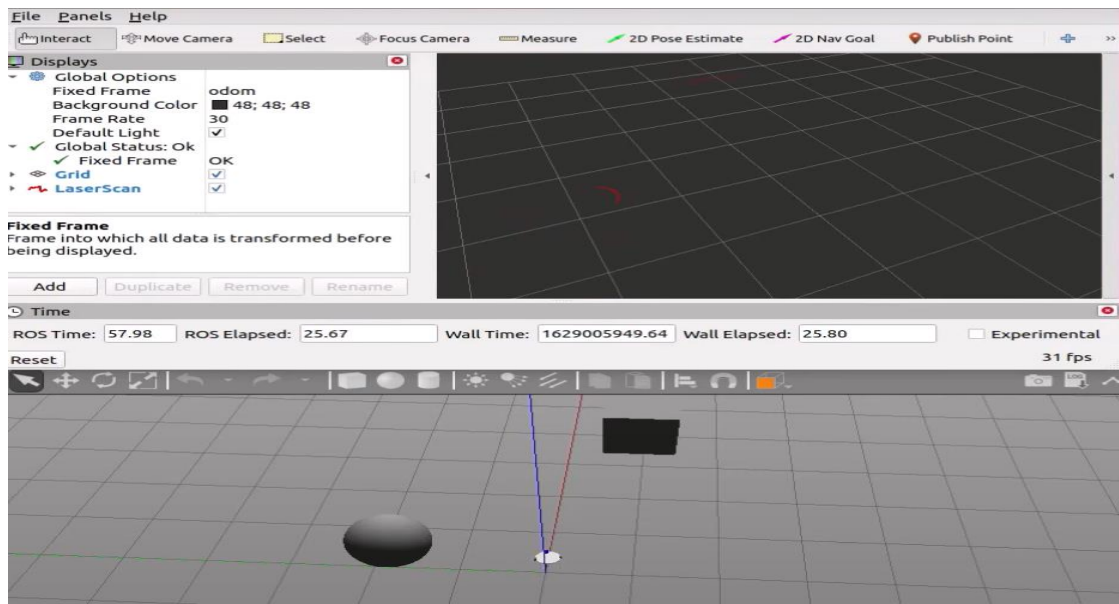
10. Maka akan muncul tampilan:



11. Command terakhir yang dilakukan pada chapter 5 yaitu 'roslaunch diff_wheeled_robot_control keyboard_teleop.launch'

```
rimeldoarfiq@rimeldoarfiq-VirtualBox: ~
rimeldoarfiq@rimeldoarfiq-VirtualBox:~$ roslaunch diff_wheeled_robot_control keyboard_teleop.launch
```

12. Maka akan muncul tampilan:



Chapter 5

Mensimulasikan robot menggunakan ROS (Robot Operating System), CoppeliaSim, dan Webots adalah pendekatan yang umum dilakukan dalam pengembangan robotika. Simulasi memungkinkan pengembang untuk menguji dan memvalidasi algoritma, kontrol, dan desain robot tanpa harus mengandalkan perangkat keras fisik. Langkah-langkahnya antara lain:

1. Unduh CoppeliaSim (<https://www.coppeliarobotics.com/downloads>) dan unduh versi yang sesuai dengan sistem operasi
2. Menjalankan CoppeliaSim menggunakan command './coppeliaSim.sh'

3. Maka akan muncul tampilan seperti gambar dibawah yang menandakan CoppeliaSim berhasil dijalankan dan siap digunakan

