**Nama : Fariz Rahman Ramadhan**

**NIM : 1103204046**

**Kelas : Robotic-Class**

# TECHNICAL REPORT MASTERING ROS FOR ROBOTICS PROGRAMMING

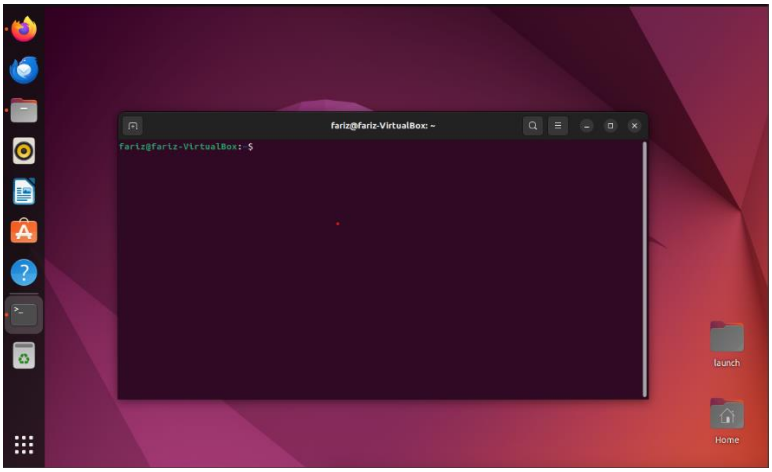
## Pendahuluan

Dalam era teknologi modern, pengembangan robotika semakin menjadi fokus utama dalam memajukan berbagai bidang kehidupan. Robotika tidak hanya memainkan peran penting dalam industri manufaktur, tetapi juga telah merambah ke sektor-sektor seperti kesehatan, layanan, dan penelitian ilmiah. Untuk mencapai tingkat fungsionalitas dan fleksibilitas yang tinggi, penggunaan perangkat lunak yang andal dan efisien menjadi krusial. Di sinilah Robot Operating System (ROS) menjadi suatu solusi yang mendukung pengembangan robotika secara efektif.

ROS, sebuah middleware open-source yang dirancang khusus untuk robotika, menyediakan infrastruktur yang memungkinkan pengembang untuk membuat dan mengelola perangkat lunak untuk robot dengan lebih mudah. Mastering ROS for Robotics Programming menjadi langkah penting untuk mendapatkan pemahaman mendalam tentang kemampuan sistem ini.

## CHAPTER 1
## INTRODUCTION TO ROS

Bab 1, "Pengenalan ROS," merupakan eksplorasi dasar ke dalam dunia Robot Operating System (ROS). Dimulai dengan gambaran pertumbuhan pesat dalam industri robotika dan kebutuhan akan platform pengembangan yang terbuka dan dapat diandalkan, bab ini menyelami asal-usul dan evolusi ROS sejak diperkenalkan oleh Willow Garage pada tahun 2007. Bab ini menjelaskan konsep-konsep kunci dalam ROS, seperti node, topik, dan layanan, membentuk dasar pemahaman arsitektur ROS secara menyeluruh. Aspek praktis, termasuk instalasi dan konfigurasi awal ROS, ditangani untuk memberdayakan pembaca agar dapat memulai perjalanan ROS tanpa hambatan. Selain itu, bab ini menyoroti berbagai manfaat yang diperoleh dari penguasaan ROS, dengan menekankan peran ROS dalam memfasilitasi kolaborasi global di antara pengembang, memanfaatkan ekosistem open-source, dan meningkatkan efisiensi serta kecanggihan solusi robotika. Sebagai hasilnya, Bab 1 bertujuan untuk membekali pembaca dengan pemahaman yang kokoh tentang dasar-dasar ROS, membuka jalan bagi keterlibatan praktis dalam pemrograman robotika dalam kerangka kerja ROS.




Membuka Mesin Virtual Ubuntu
Pertama, pastikan mesin virtual telah terinstal dan dikonfigurasi dengan sistem operasi Ubuntu. Jalankan mesin virtual dan tunggu hingga sistem operasi Ubuntu sepenuhnya dimuat.

Setelah itu menambahkan repository ROS ke daftar repository paket dengan perintah **sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'.** Ini berarti sedang menyesuaikan sistem untuk mendownload dan menginstal ROS dari repository resmi

```
fariz@fariz-VirtualBox:~$ sudo apt install -y ros-noetic-desktop-full
[sudo] password for fariz:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Selanjutnya, melakukan pembaruan pada daftar paket sistem menggunakan sudo apt update agar sistem memiliki informasi paket terbaru. Setelahnya, menginstal ROS Noetic Desktop-Full dengan perintah **sudo apt install -y ros-noetic-desktop-full**. Ini akan menginstal ROS beserta pustaka dan alat pengembangan yang lengkap.

Untuk memastikan dependensi yang dibutuhkan oleh ROS terpenuhi, menginisialisasi rosdep dengan perintah sudo rosdep init dan mengupdate database dengan rosdep update.

```
fariz@fariz-VirtualBox:~$ sudo rosdep init
```

Kemudian, menambahkan konfigurasi lingkungan ROS ke berkas .bashrc menggunakan echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc dan memuat ulang berkas .bashrc dengan source ~/.bashrc. Hal ini memastikan bahwa lingkungan ROS akan terpasang secara otomatis setiap kali terminal dibuka.

Terakhir, menginstal dependensi Python untuk ROS menggunakan sudo apt install -y python3-rosinstall python3-rosinstall-generator python3-wstool build-essential. Ini mencakup alat-alat pengembangan yang diperlukan untuk pengembangan Python di lingkungan ROS.

Sebagai pengecekan akhir, mencetak pesan bahwa instalasi ROS Noetic telah berhasil dan memberikan instruksi untuk mengaktifkan lingkungan dengan menjalankan source

/opt/ros/noetic/setup.bash. Dengan demikian, langkah-langkah tersebut memberikan fondasi yang diperlukan untuk memulai pengembangan aplikasi robotika menggunakan ROS Noetic di sistem Ubuntu.

Why Use ROS:

- Kerangka kerja yang fleksibel dan modular, memudahkan pengembangan sistem robotik kompleks.
- Memungkinkan integrasi berbagai komponen perangkat keras dan perangkat lunak.
- Fokus pada fungsionalitas tingkat tinggi daripada detail tingkat rendah.
- Mendorong kolaborasi melalui platform open-source, mempercepat laju inovasi dalam komunitas robotika.

Basic Elements of ROS Framework:

- Nodes: Unit komputasi individual yang melakukan tugas-tugas tertentu.
- Topics: Fasilitas komunikasi antar nodes dengan mempublikasikan dan berlangganan pesan.
- Messages: Berisi data yang dipertukarkan antar nodes.
- Services: Memungkinkan nodes meminta dan menerima tugas-tugas tertentu.
- Parameter Server: Menyimpan dan mengelola parameter konfigurasi untuk seluruh sistem ROS.

Prerequisites for Programming with ROS:

- Pemahaman dasar tentang sistem operasi berbasis Linux.
- Penguasaan bahasa pemrograman seperti Python atau C++.
- Familiaritas dengan kontrol versi (contoh: Git) dan prinsip dasar robotik.
- Pengetahuan dasar tentang kinematika dan dinamika robotik dapat bermanfaat.

Internal Working of roscore:

- roscore menginisialisasi ROS Master, Parameter Server, dan infrastruktur komunikasi.
- ROS Master memfasilitasi komunikasi antara node-node dengan melacak penerbit, pelanggan, dan layanan.
- Parameter Server mengelola parameter-runtime untuk sistem ROS.
- Menggunakan XML-RPC dan RPCROS untuk infrastruktur komunikasi.
- Menerapkan mekanisme sinkronisasi waktu untuk waktu yang konsisten di antara node-node dalam jaringan ROS.

# CHAPTER 2
# GETTING STARTED WITH ROS PROGRAMMING

Pada bab ini, akan dibahas beberapa topik utama yang esensial untuk memulai pemrograman ROS. Pertama, akan dimulai dengan pembuatan paket ROS, yang merupakan langkah awal yang penting dalam pengembangan aplikasi ROS. Proses ini melibatkan konfigurasi ruang kerja ROS, membuat struktur paket, dan memahami hierarki direktori ROS. Selanjutnya, akan mengeksplorasi cara menambahkan pesan dan layanan kustom ke dalam paket ROS tersebut. Ini akan memberikan pemahaman mendalam tentang cara menyesuaikan komunikasi antar-nodes sesuai dengan kebutuhan aplikasi robotika yang spesifik.

Selanjutnya, akan membahas penggunaan layanan ROS, yang memungkinkan nodes untuk saling berinteraksi dengan cara yang terstruktur. Ini mencakup pembuatan dan implementasi layanan, memungkinkan pembaca memahami cara memanfaatkan mekanisme layanan untuk menyediakan dan meminta fungsionalitas khusus antar-nodes. Disini juga akan menjelajahi pembuatan file launch dalam ROS, yang memfasilitasi pengelolaan konfigurasi dan jalur eksekusi yang kompleks. Terakhir, akan mengulas berbagai aplikasi dari topik, layanan, dan actionlib dalam konteks pengembangan robotika. Ini mencakup cara nodes berkomunikasi melalui topik, memanfaatkan layanan untuk akses ke fitur tertentu, dan mengimplementasikan tindakan (actionlib) untuk tugas-tugas yang memerlukan kontrol tingkat tinggi. Keseluruhan, bab ini bertujuan untuk memberikan pemahaman yang komprehensif tentang penggunaan elemen-elemen dasar ROS dalam pengembangan aplikasi robotika yang praktis dan efektif.

```
fariz@fariz-VirtualBox:~$ roscd
```

Perintah roscd digunakan untuk mengubah direktori kerja saat ini ke direktori paket ROS tertentu. ROS (Robot Operating System) mengadopsi struktur kerja berbasis paket, di mana setiap paket dapat memiliki direktori kerja sendiri yang mengandung kode sumber, file konfigurasi, dan sumber daya lainnya yang terkait dengan paket tersebut. Dengan menggunakan roscd, pengguna dapat dengan cepat berpindah ke direktori kerja paket tertentu tanpa harus mengetik jalur lengkapnya.

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

Perintah catkin_make digunakan dalam lingkungan Catkin untuk membangun (compile) paket-paket ROS. Catkin adalah sistem pembangunan yang digunakan dalam ROS untuk mengelola dan menyusun kode sumber dari berbagai paket dalam satu proyek robotika. Saat bekerja dengan ROS, pembuatan kode sumber dan penyusunan perlu dilakukan untuk menghasilkan eksekutabel, pustaka, dan berkas konfigurasi yang dapat dijalankan di robot atau simulasi.

Proses pembangunan menggunakan catkin_make melibatkan sejumlah langkah, seperti memeriksa dependensi paket, mengonfigurasi proyek, mengkompilasi kode sumber, dan menyusun output hasil kompilasi ke dalam struktur direktori yang tepat.

```
fariz@fariz-VirtualBox:~$ rosrun mastering_ros_demo_pkg demo_topic_publisher
```

Perintah rosrun mastering_ros_demo_pkg demo_topic_publisher digunakan untuk menjalankan sebuah ROS node yang disebut demo_topic_publisher dari paket ROS yang disebut mastering_ros_demo_pkg. Dengan menjalankan perintah ini, akan meluncurkan node demo_topic_publisher, yang kemungkinan akan memulai publikasi pesan ke suatu topik tertentu di dalam ROS network. Hal ini menjadi langkah awal dalam menguji atau menjalankan demonstrasi fungsi-fungsi tertentu yang diimplementasikan dalam node.

```
fariz@fariz-VirtualBox:~$ rosrun mastering_ros_demo_pkg demo_service_client
```

Perintah rosrun mastering_ros_demo_pkg demo_service_client digunakan untuk menjalankan ROS node yang disebut demo_service_client dari paket ROS mastering_ros_demo_pkg.
(rosrun mastering_ros_demo_pkg demo_topic_publisher) dapat terkait dengan demonstrasi atau pengujian yang melibatkan dua buah node, satu sebagai publisher (pengirim pesan ke topik) dan yang lain sebagai service client (meminta layanan). demo_topic_publisher mengirimkan pesan ke suatu topik, dan demo_service_client kemungkinan menggunakan informasi dari topik tersebut atau menanggapi perubahan pada topik tersebut dengan meminta layanan dari suatu node. Dengan menjalankan kedua perintah tersebut, dapat dilihat bagaimana dua buah node dapat berkomunikasi satu sama lain dalam konteks ROS.

Dalam kodingan di atas, terdapat dua konteks utama: konteks publisher dan konteks client.

Publisher (demo_topic_publisher):

Node demo_topic_publisher kemungkinan berfungsi sebagai publisher, yang bertanggung jawab untuk mempublikasikan (mengirimkan) pesan ke suatu topik di dalam sistem ROS.
Topik yang menjadi target dapat diidentifikasi dari kode node tersebut. Misalnya, node tersebut mempublikasikan data sensor atau informasi lainnya ke suatu topik tertentu.

Konteks Client (demo_service_client):
Node demo_service_client kemungkinan berfungsi sebagai client layanan (service client), yang berarti node ini akan meminta layanan dari suatu service server dalam sistem ROS.
Service server tersebut dapat mengimplementasikan fungsionalitas tertentu, dan node demo_service_client akan menggunakan layanan ini sesuai dengan kebutuhan aplikasi.
Service server ini terkait dengan data yang dipublikasikan oleh node demo_topic_publisher, atau terkait dengan tugas-tugas lain di dalam konteks aplikasi robotika yang lebih besar.
Secara keseluruhan, dengan menjalankan kedua node ini, dapat dilihat bagaimana node yang berbeda dapat berkomunikasi melalui mekanisme pub/sub (publisher/subscriber) dan service client/server di dalam framework ROS. Publikasi pesan dan pemanggilan layanan adalah dua metode umum di dalam ROS yang memungkinkan nodes berinteraksi dan bekerja bersama.

Protokol Komunikasi yang Didukung oleh ROS:

- ROS mendukung protokol komunikasi publish/subscribe (pub/sub) melalui ROS topics dan client/server melalui ROS services.

- Dalam pub/sub, nodes berkomunikasi asinkron melalui topics, dengan satu node sebagai publisher dan nodes lain sebagai subscribers.
- Dalam client/server, komunikasi bersifat sinkron melalui services, dengan satu node sebagai service server yang menyediakan layanan dan nodes lain sebagai service clients yang memintanya.

Perbedaan antara rosrun dan roslaunch:
- `rosrun` digunakan untuk menjalankan sebuah spesifik ROS node.
- `roslaunch` digunakan untuk meluncurkan sekumpulan nodes atau konfigurasi yang lebih kompleks.
- `roslaunch` memberikan fleksibilitas dan kontrol yang lebih besar, memungkinkan pengaturan parameter, penentuan dependencies, dan manajemen lebih kompleks dalam eksekusi nodes ROS.

Perbedaan antara ROS Topics dan Services dalam Operasionalnya:
- ROS topics menggunakan model pub/sub, di mana pesan dikirimkan asinkron dari publisher ke subscribers.
- ROS services beroperasi dalam model client/server, di mana nodes sebagai clients meminta layanan secara sinkron dari node sebagai service server.
- Topics digunakan untuk komunikasi asinkron yang memancarkan perubahan status atau data, sedangkan services digunakan untuk permintaan layanan yang bersifat sinkron dan responsif.

Perbedaan antara ROS Services dan Actionlib dalam Operasionalnya:
- Actionlib menyediakan mekanisme yang lebih lanjut daripada services, mendukung tugas-tugas yang kompleks dan dapat dilacak.
- Dalam actionlib, goal (tujuan) didefinisikan, dan server memberikan status yang dapat diperbarui selama eksekusi.
- Actionlib cocok untuk tugas-tugas yang memerlukan umpan balik atau kemajuan yang diperbarui selama eksekusi, seperti navigasi atau manipulasi yang melibatkan pergerakan robot.

# CHAPTER 3
# WORKING WITH ROS WITH 3D MODELING

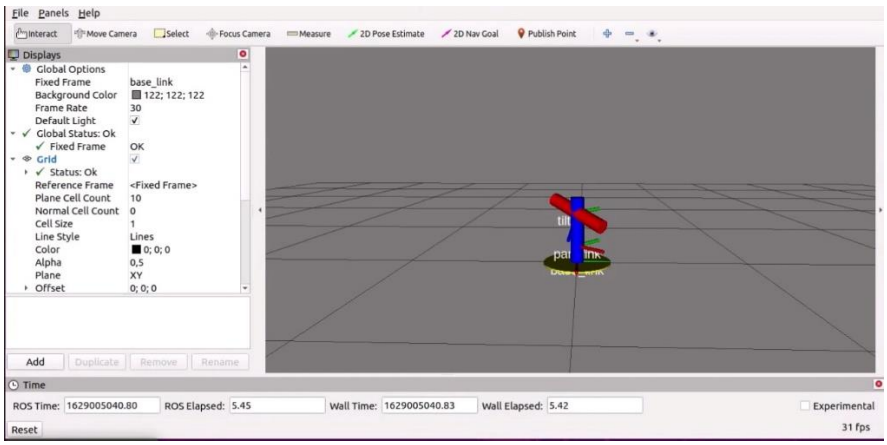
Bab ini dimulai dengan mengulas ROS packages yang berfokus pada pemodelan robot. ROS menyediakan sejumlah paket yang khusus dirancang untuk membantu pengembang dalam memodelkan dan merepresentasikan robot secara tiga dimensi. Ini termasuk alat-alat yang memungkinkan integrasi yang lebih baik antara perangkat keras robot dan simulasi perangkat lunak, menciptakan dasar yang solid untuk pengembangan dan pengujian aplikasi robotika.

Pentingnya Unified Robot Description Format (URDF) sebagai standar dalam pemodelan robot dijelaskan dalam bab ini. membahas cara membuat ROS package yang khusus digunakan untuk mendefinisikan deskripsi robot. Pembahasan selanjutnya fokus pada pembuatan model URDF pertama, yang melibatkan penjelasan mendalam tentang struktur dan konten file URDF. Langkah-langkah ini membantu pembaca memahami bagaimana menggambarkan fisik dan geometri robot secara terinci dalam lingkungan ROS.

Setelah merancang model 3D menggunakan URDF, langkah selanjutnya adalah memvisualisasikannya menggunakan RViz. RViz adalah alat visualisasi di ROS yang memungkinkan untuk melihat dan menganalisis model robot secara interaktif. Untuk melakukan ini, dapat membuat sebuah file peluncuran (launch file) bernama view_demo.launch dan menempatkannya di dalam folder launch. Berikut adalah contoh kode yang dapat dimasukkan ke dalam file tersebut. Navigasikan ke direktori mastering_ros_robot_description_pkg/launch untuk menemukan dan mengelola file tersebut:

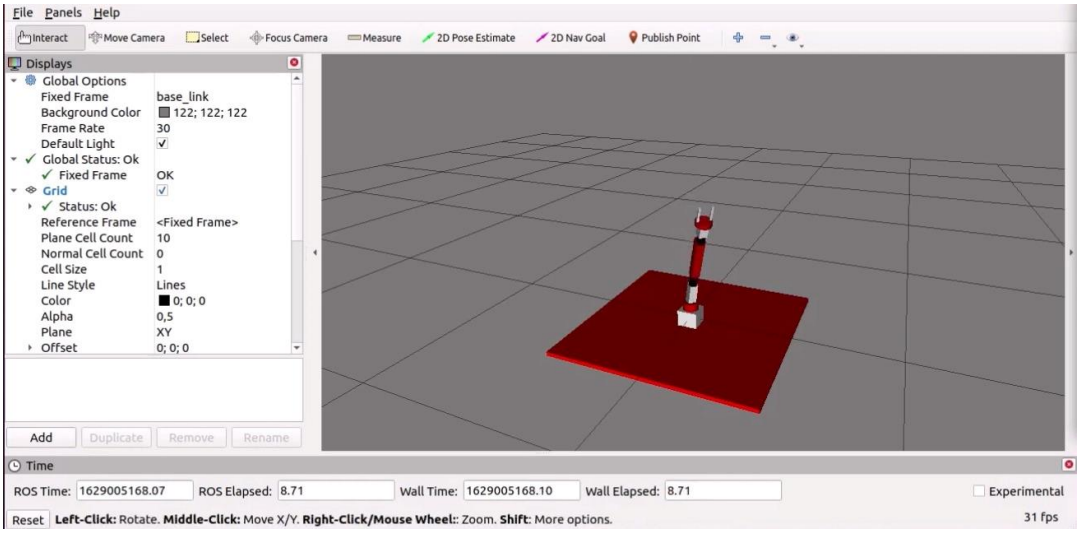
```
fariz@fariz-VirtualBox:~$ roslaunch mastering_ros_robot_description_pkg view_demo.launch
```

Kode roslaunch view_demo.launch yang diberikan memiliki tujuan untuk meluncurkan visualisasi model robot yang telah dirancang menggunakan URDF di RViz, sebuah alat visualisasi di ROS. File ini menyimpan pengaturan tampilan, termasuk konfigurasi frame referensi, tampilan topik, dan elemen visualisasi lainnya.
Jadi, secara keseluruhan, kodingan ini digunakan untuk memuat model robot dari file URDF dan meluncurkan RViz dengan konfigurasi tampilan yang telah ditentukan, sehingga pengguna dapat dengan mudah memvisualisasikan dan menganalisis model robot tersebut di lingkungan RViz.



Bab ini juga menyajikan pendekatan alternatif menggunakan XML Macros (Xacro) untuk pemodelan robot. Xacro memberikan fleksibilitas yang signifikan dalam menyusun dan mengelola deskripsi robot dengan menyederhanakan dan memodularisasi kode URDF. Pembaca dipandu melalui proses mengonversi kode Xacro menjadi URDF dan kemudian diperkenalkan pada pembuatan deskripsi robot untuk manipulator dengan tujuh derajat kebebasan (DOF) dan model robot untuk differential drive mobile robot. Ini memberikan dasar yang kokoh untuk memahami dan menerapkan pemodelan robot yang efektif menggunakan ROS.

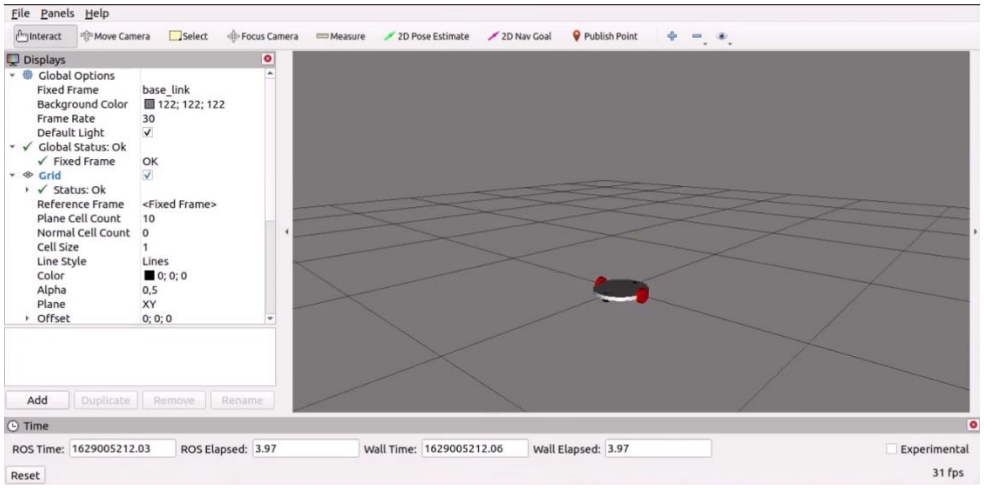
```
fariz@fariz-VirtualBox:~$ roslaunch mastering_ros_robot_description_pkg view_arm.launch
```




Kode roslaunch mastering_ros_robot_description_pkg view_arm.launch juga digunakan untuk meluncurkan visualisasi model robot arm yang telah dirancang menggunakan URDF di RViz.

Selanjutnya memasukan kodingan dibawah untuk membuat model robot, untuk diferensial mobile robot

```
fariz@fariz-VirtualBox:~$ roslaunch mastering_ros_robot_description_pkg view_mobile_robot.launch
```




Packages Used for Robot Modeling in ROS:

- urdf: Menyediakan alat dan fungsi untuk mendefinisikan model robot menggunakan URDF (Unified Robot Description Format).
- xacro: Memungkinkan representasi robot yang lebih modular dan reusable dengan menggunakan XML Macros (Xacro).
- joint_state_publisher: Digunakan untuk mengirimkan informasi status joint ke simulator atau robot fisik.
- robot_state_publisher: Menyediakan transformasi antara frame referensi robot dan dunia.

Important URDF Tags for Robot Modeling:

- <link>: Mendefinisikan link dalam robot.
- <joint>: Mendefinisikan hubungan antara dua link melalui suatu joint.
- <material>: Menyertakan informasi tentang sifat-sifat material dan visual link.
- <collision>: Mendefinisikan geometri untuk keperluan deteksi tabrakan.
- <inertial>: Menyertakan informasi inersia yang digunakan untuk perhitungan dinamika.

Reasons for Using Xacro over URDF:

- Modularitas: Xacro memungkinkan representasi robot yang modular dan reusable.
- Keterbacaan: Kode Xacro lebih ringkas dan lebih mudah dibaca daripada XML URDF.
- Kemudahan Penyuntingan: Xacro mempermudah penyesuaian dan penyuntingan model robot tanpa perlu mengubah seluruh file URDF.

Function of Joint State Publisher and Robot State Publisher Packages:

- joint_state_publisher: Mengirimkan informasi tentang status joint ke simulator atau robot fisik, memungkinkan pengguna untuk memperbarui dan mengamati posisi joint.
- robot_state_publisher: Menyediakan transformasi antara sistem koordinat (frame) referensi robot dan dunia, memungkinkan visualisasi model robot dalam lingkungan 3D seperti RViz.

Function of Transmission Tag in URDF:

- <transmission>: Digunakan untuk mendefinisikan cara pergerakan energi melalui joint, termasuk jenis transmisi seperti gearbox atau differential drive.
- Memungkinkan pengguna untuk merinci bagaimana gaya dan torsi dihubungkan antara joint, menggambarkan dengan lebih akurat sifat transmisi energi di dalam model robot.

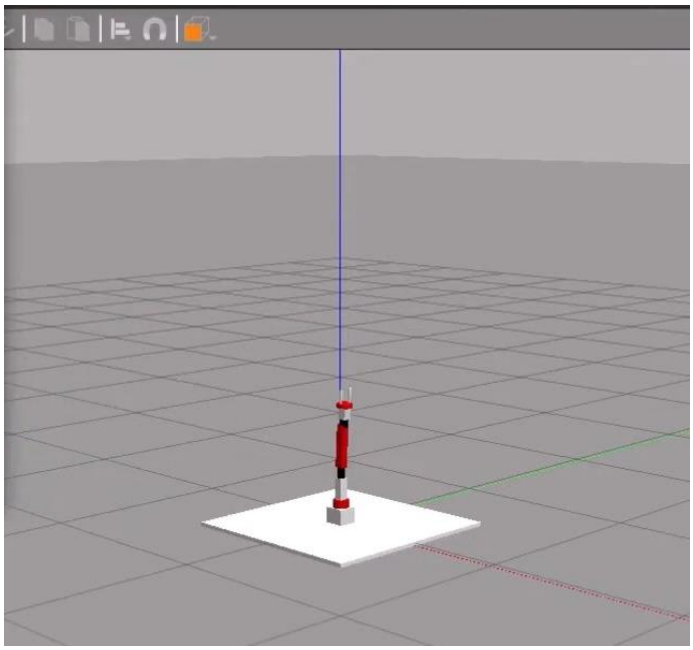
# CHAPTER 4

# SIMULATING ROS USING ROS AND GAZEBO

Pada Bab 4 ini, memahami esensi simulasi robotik dan peran Gazebo sebagai lingkungan simulasi utama di ROS. memulai dengan memahami konsep dasar simulasi robotik dan pentingnya Gazebo sebagai simulasi realistik yang mendukung pengembangan dan pengujian robotika. Selanjutnya, pembaca akan dipandu melalui langkah-langkah simulasi model lengan robotik menggunakan Gazebo. Ini mencakup pembuatan model simulasi, mengintegrasikannya dengan ROS, dan memberikan kemampuan untuk simulasi pergerakan serta respons sensor.

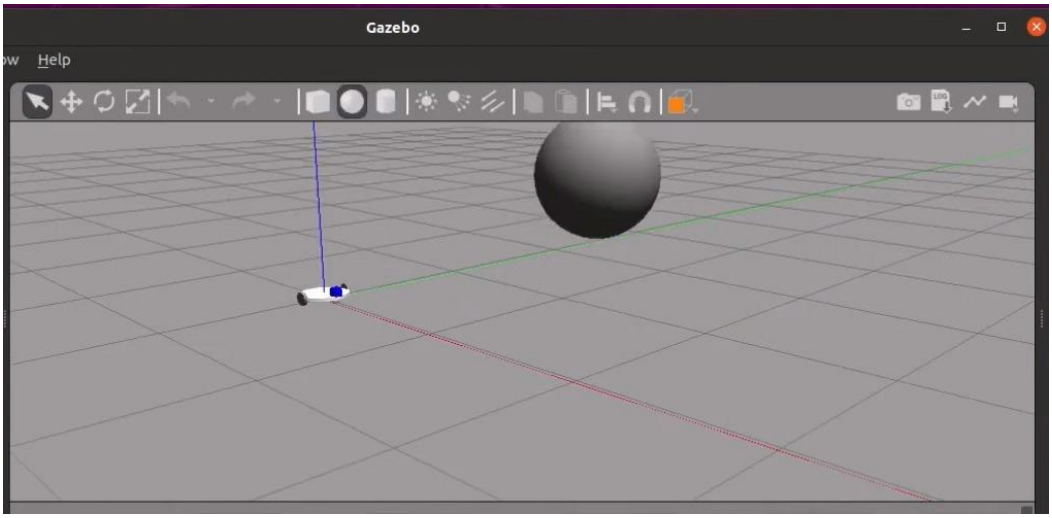
Setelah memahami dasar-dasar simulasi, bab ini akan memperluas wawasan dengan melalui simulasi robotik yang lebih kompleks. Ini mencakup simulasi lengan robotik dengan sensor kedalaman, memungkinkan pemodelan lebih lanjut dalam konteks persepsi robot. Selanjutnya, akan menjelajahi cara menggerakkan joint robot menggunakan ROS controllers di Gazebo, membuka pintu bagi pengembangan kontroler robot yang lebih canggih. Tak hanya itu, pembaca juga akan mempelajari simulasi robot roda berdiferensiasi dan kemudian melihat bagaimana mengendalikan robot beroda tersebut secara jarak jauh (teleoperasi) dalam lingkungan simulasi Gazebo. Dengan bab ini, pembaca akan mendapatkan pemahaman yang mendalam tentang bagaimana melakukan simulasi robotik yang efektif menggunakan ROS dan Gazebo, membuka peluang eksplorasi dan pengembangan lebih lanjut dalam pengembangan robotika.

```
fariz@fariz-VirtualBox:~$ roslaunch seven_dof_arm_gazebo seven_dof_arm_gazebo_control.launch
```



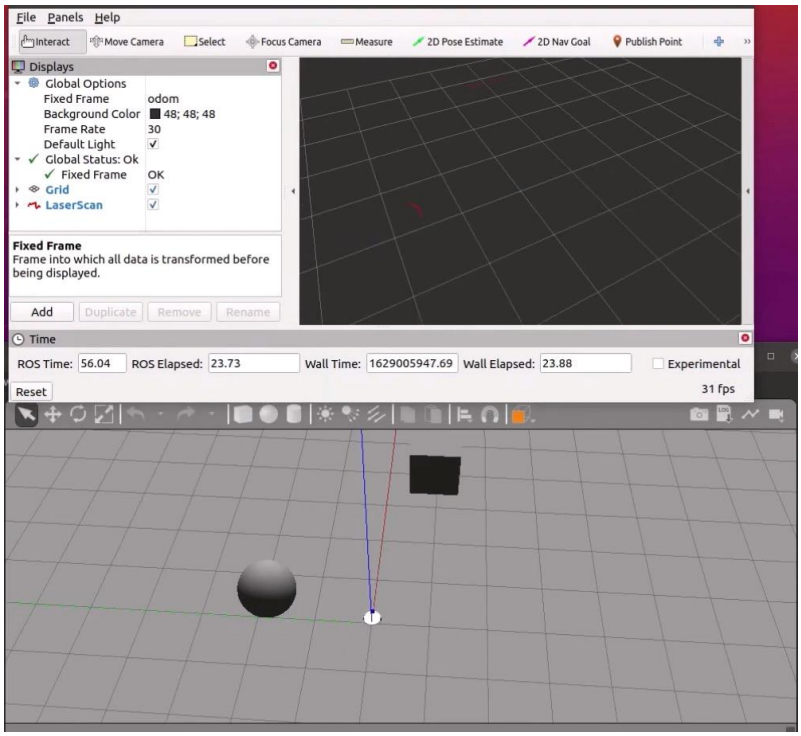
Perintah diatas akan menampilkan robot gazebo, dan memunculkan robot seperti gambar tersebut.

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




Perintah diatas yaitu roslaunch diff_wheeled_robot_gazebo diff_wheeled_gazebo_full.launch, dimasukan untuk menampilkan robot wheelez gazebo seperti diatas

```
fariz@fariz-VirtualBox:~$ roslaunch diff_robot_control keyboard_teleop.launch
```




Dan terakhir masukan perintah yaitu roslaunch diff_robot_control keyboard_teleop.launch.

Tujuan Simulasi Robot:

- Pengembangan dan Pengujian: Simulasi robot memberikan lingkungan terkendali untuk pengembangan dan pengujian algoritma dan perangkat lunak robotik tanpa memerlukan robot fisik.

- Efisiensi Biaya: Simulasi membantu mengurangi biaya yang terkait dengan perangkat keras dan potensi kerusakan selama pengujian, memungkinkan pengembangan berulang dalam ruang virtual.
- Validasi Algoritma: Simulasi memungkinkan validasi dan penyempurnaan algoritma robotik, memungkinkan pengembang mengoptimalkan kinerja sebelum diimplementasikan pada robot nyata.

Menambahkan Sensor ke Simulasi Gazebo:

- Plugin Gazebo: Sensor dapat ditambahkan ke Gazebo menggunakan plugin, yaitu perangkat lunak tambahan yang memperluas fungsionalitas Gazebo.
- Model URDF: Sensor biasanya ditambahkan ke model URDF robot, mendefinisikan propertinya dan hubungannya dalam simulasi.
- Paket ROS: ROS menyediakan paket yang memudahkan integrasi sensor ke dalam lingkungan simulasi Gazebo.

Jenis Kontroler ROS dan Antarmuka Perangkat Keras:

- Kontroler Sendi: Bertanggung jawab untuk mengontrol posisi, kecepatan, atau usaha sendi dalam sistem robotik.
- Kontroler Usaha: Mengontrol langsung torsi atau gaya yang diterapkan pada sendi.
- Kontroler Kecepatan: Mengatur kecepatan sendi untuk mencapai pergerakan robot yang diinginkan.
- Kontroler Posisi: Mengontrol posisi sendi untuk manipulasi robot yang tepat.
- Antarmuka Perangkat Keras: ROS mendukung berbagai antarmuka perangkat keras yang memungkinkan komunikasi dengan aktuator, sensor, dan perangkat robotik lainnya.

Menggerakkan Robot Bergerak dalam Simulasi Gazebo:

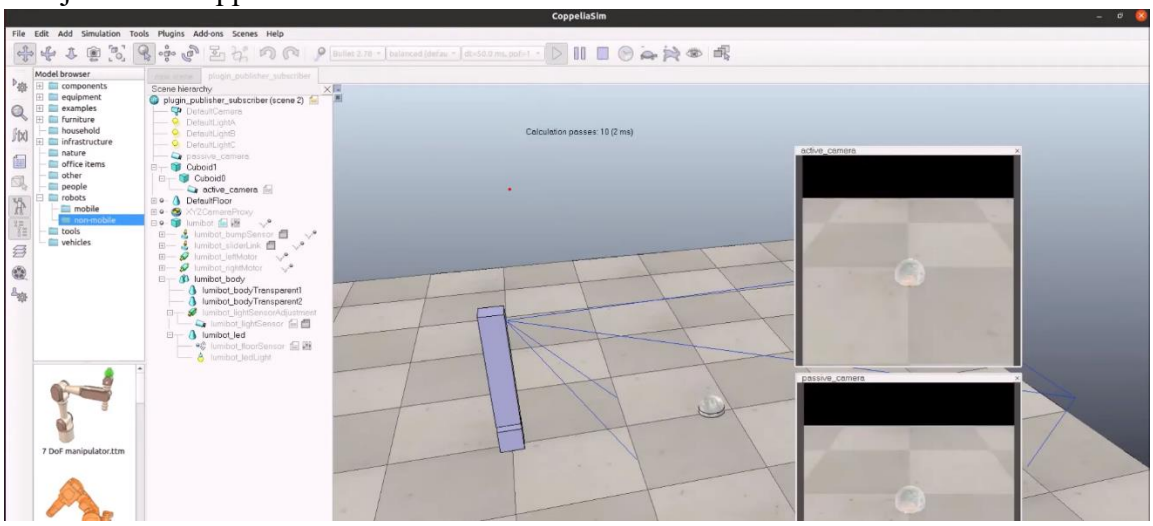
- Kontroler ROS: Memanfaatkan kontroler ROS, seperti kontroler differential drive, untuk memberikan perintah kecepatan roda pada robot bergerak.
- Teleoperasi: Menggunakan node atau antarmuka teleoperasi dalam ROS untuk mengendalikan pergerakan robot secara jarak jauh dalam simulasi.
- Plugin Gazebo: Menggunakan plugin Gazebo untuk mensimulasikan dan mengendalikan dinamika robot bergerak, memungkinkan gerakan dan interaksi yang realistis dalam lingkungan virtual.

# CHAPTER 5
# SIMULATING ROBOTS USING ROS, COPPELIASIM, AND WEBOTS

Bab ini akan membahas simulasi robot menggunakan beberapa lingkungan simulasi, yaitu CoppeliaSim dan Webots, yang diintegrasikan dengan ROS. Langkah-langkah kunci yang akan dibahas melibatkan. Melibatkan proses konfigurasi dan integrasi CoppeliaSim dengan ROS. Ini termasuk langkah-langkah pengaturan yang diperlukan untuk menjembatani komunikasi antara ROS dan lingkungan simulasi CoppeliaSim. Mendemonstrasikan cara mensimulasikan model lengan robotik menggunakan CoppeliaSim dan mengintegrasikannya dengan ROS. Ini mencakup pembuatan model, pengontrolan pergerakan, dan pertukaran data dengan ROS.
Menyajikan langkah-langkah untuk mengkonfigurasi dan mengintegrasikan Webots dengan ROS. Pembahasan akan mencakup proses pengaturan agar ROS dapat berkomunikasi dengan simulasi yang berjalan di lingkungan Webots.

```
fariz@fariz-VirtualBox:~$ cd dev
fariz@fariz-VirtualBox:~/dev$ cd coppeliaSim
fariz@fariz-VirtualBox:~/dev/coppeliaSim$ ./coppeliaSim.sh
```

Menjalankan Coppeliasim




Setelah itu Memandu dalam menulis kontroler pertama untuk digunakan dalam simulasi. Ini mencakup pemahaman dasar tentang bagaimana mengontrol perilaku robotik dan implementasi kontroler sederhana. Menjelaskan proses menulis node teleoperasi menggunakan paket webots_ros. Node ini memungkinkan pengguna untuk mengendalikan robot secara jarak jauh dalam simulasi Webots menggunakan perintah dari ROS. Bab ini bertujuan memberikan pemahaman yang mendalam tentang bagaimana mensimulasikan robot menggunakan CoppeliaSim dan Webots, sambil mengintegrasikan kemampuan tersebut dengan ROS. Dengan demikian, pembaca akan memiliki dasar yang kuat untuk eksplorasi lebih lanjut dalam pengembangan dan pengujian robotik.

Komunikasi antara CoppeliaSim dan ROS:

- Plugin Antarmuka ROS: CoppeliaSim menyediakan plugin antarmuka ROS yang berfungsi sebagai jembatan antara simulasi dan ROS. Plugin ini memungkinkan komunikasi melalui topik, layanan, dan aksi ROS.
- Jembatan ROS: Jembatan ROS memungkinkan CoppeliaSim untuk memublikasikan dan berlangganan topik ROS, memfasilitasi pertukaran data antara lingkungan simulasi dan ROS.

Mengontrol Simulasi CoppeliaSim dengan ROS:

- Layanan dan Topik ROS: ROS menyediakan layanan dan topik yang memungkinkan pengguna mengirim perintah dan menerima informasi dari CoppeliaSim. Misalnya, mengontrol posisi atau kecepatan joint dapat dicapai dengan memublikasikan ke topik tertentu atau memanggil layanan ROS.

Mengimpor Model Robot Baru di CoppeliaSim dan Mengintegrasikannya dengan ROS:

- Impor URDF: CoppeliaSim mendukung impor model robot yang didefinisikan dalam format URDF. Pengguna dapat mengimpor model URDF langsung ke CoppeliaSim untuk simulasi.
- Integrasi ROS: Setelah model robot diimpor, plugin antarmuka ROS memungkinkan integrasi dengan ROS. Ini melibatkan konfigurasi penerbit dan pelanggan ROS untuk berkomunikasi dengan model robot yang diimpor.

Webots sebagai Perangkat Lunak Mandiri:

- Ya: Webots dapat digunakan sebagai perangkat lunak mandiri untuk simulasi dan pengembangan robot. Ini menyediakan lingkungan yang ramah pengguna dengan antarmuka grafis untuk merancang, mensimulasikan, dan menguji robot.

Komunikasi ROS dan Webots:

- Kontroler ROS untuk Webots: Webots menyediakan kontroler ROS yang memungkinkan komunikasi antara ROS dan Webots. Ini memungkinkan pengguna mengontrol dan memantau simulasi Webots menggunakan perintah dan topik ROS.
- Paket rosbridge_suite: Paket rosbridge_suite dapat digunakan untuk membangun komunikasi antara ROS dan Webots. Ini berfungsi sebagai server WebSocket, memungkinkan komunikasi dua arah antara kedua sistem.