UAS Robotika

Mastering ROS for Robotics Programming: Bab I – Bab VI

BAB I - Pengenalan ROS

Dalam bab ini, kita akan mempelajari dasar-dasar Robot Operating System (ROS), termasuk alasan utama penggunaan ROS, struktur sistem file ROS, hierarki komputasi, dan komunitas ROS.

1.1 Alasan Menggunakan ROS

ROS menjadi pilihan utama dalam pengembangan sistem robotika modern karena beberapa alasan berikut:

- **Terbaru dan Potensial**: ROS adalah platform yang selalu diperbarui dan memiliki potensi besar dalam bidang robotika.
- **Alat dan Fungsi yang Lengkap**: ROS menyediakan berbagai alat dan fungsi yang memudahkan dalam visualisasi, pemrograman, dan simulasi robot.
- **Dukungan untuk Sensor Kelas Atas**: ROS sangat kompatibel dengan berbagai jenis sensor canggih yang digunakan dalam robotika.

1.2 Struktur Sistem File ROS

ROS memiliki struktur file sistem yang terorganisir dalam beberapa level, yaitu:

- **Metapackages**: Kumpulan paket virtual yang biasanya tidak berisi kode atau file, tetapi mengelompokkan paket-paket terkait.
- **Packages**: Elemen sentral dalam ROS yang berisi semua komponen yang diperlukan untuk suatu proyek, seperti kode, pesan, layanan, dan lainnya. Paket-paket ini diatur dalam urutan yang mencakup:
    - **Package Manifest**: Berisi informasi tentang paket, seperti penulis, lisensi, dependensi, dan flag kompilasi.
    - **Messages**: Struktur data yang digunakan untuk komunikasi antar-node.
    - **Services**: Mekanisme untuk permintaan dan respons data antar-node.
    - **Code**: Kode sumber yang mengimplementasikan fungsionalitas paket.
    - **Miscellaneous**: Berkas-berkas lainnya yang mendukung paket.

1.3 Hierarki Komputasi di ROS

Komputasi di ROS diatur dalam beberapa komponen utama, yaitu:

- **Nodes**: Proses individu yang melakukan tugas komputasi spesifik.
- **Master**: Entitas yang menangani registrasi nama dan pencarian proses (nodes) lain.
- **Parameter Server**: Penyimpanan terpusat untuk parameter yang dapat diakses oleh nodes.
- **Messages**: Data yang dikirimkan antar-nodes melalui topik.
- **Topics**: Saluran komunikasi di mana messages dipublikasikan dan disubscrib.
- **Services**: Layanan yang memungkinkan komunikasi sinkron antar-nodes melalui permintaan dan respons.
- **Bags**: Format file untuk menyimpan dan memutar ulang messages.

BAB II – *Getting Started with ROS Programming*

Fase Pertama dalam Pembuatan Robot: Desain dan Pemodelan

Fase awal dalam pembuatan robot mencakup proses desain dan pemodelan. Dalam tahap ini, kita menggunakan alat CAD (*Computer-Aided Design*) seperti Autodesk Fusion 360, SolidWorks, Blender, dan lainnya untuk merancang dan memodelkan robot. Tujuan utama dari pemodelan robot adalah untuk melakukan simulasi, yang memungkinkan kita untuk mengidentifikasi dan memperbaiki cacat kritis dalam desain sebelum melanjutkan ke fase produksi.

1.1 Tujuan Pemodelan dan Simulasi

Simulasi robotik adalah langkah penting karena:

- **Memastikan Fungsi Desain**: Simulasi membantu memastikan bahwa desain robot berfungsi dengan baik sesuai dengan spesifikasi yang diinginkan.
- **Identifikasi Cacat**: Membantu mengidentifikasi dan memperbaiki cacat desain sebelum masuk ke fase produksi yang lebih mahal dan memakan waktu.

1.2 Proyek Desain Robot

Dalam bab ini, kita akan membahas proses desain dua jenis robot:

- **Manipulator dengan Tujuh Derajat Kebebasan (DOF)**: Robot yang dirancang untuk memiliki tujuh titik artikulasi atau sendi, memungkinkan fleksibilitas gerak yang tinggi.
- **Robot dengan Penggerak Diferensial**: Robot yang menggunakan dua roda yang digerakkan secara independen untuk navigasi dan manuver.

1.3 Tahapan Lanjutan

Pada bab-bab selanjutnya, kita akan melanjutkan dengan:

- **Simulasi**: Menggunakan perangkat lunak simulasi untuk menguji model robot.
- **Pembangunan Perangkat Keras**: Membangun robot fisik berdasarkan desain yang telah disimulasikan.
- **Interfacing dengan ROS**: Mengintegrasikan robot dengan ROS untuk pengendalian dan pemantauan.

1.4 Menggunakan ROS untuk Desain dan Pemodelan

Jika kita berencana membuat model 3D robot dan mensimulasikannya menggunakan ROS, kita perlu mempelajari beberapa paket ROS yang dapat membantu dalam proses ini. Membuat model robot di ROS penting untuk berbagai alasan:

- **Simulasi dan Kontrol**: Untuk mensimulasikan dan mengontrol robot secara virtual.
- **Visualisasi**: Untuk memvisualisasikan robot dalam lingkungan simulasi.
- **Informasi Struktur dan Kinematika**: Menggunakan alat ROS untuk mendapatkan informasi mengenai struktur dan kinematika robot.

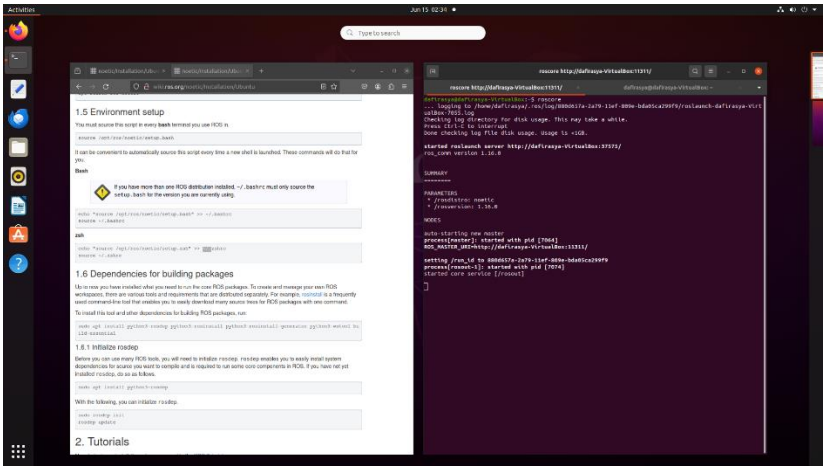
ROS menyediakan beberapa paket penting untuk mendesain dan membuat model robot, di antaranya:

- **urdf**: Digunakan untuk membuat deskripsi model 3D robot dalam format XML.
- **kdl_parser**: Untuk memparsing deskripsi kinematika robot.
- **robot_state_publisher**: Untuk mempublikasikan transformasi keadaan robot berdasarkan model URDF.
- **collada_urdf**: Untuk mengonversi model URDF ke format COLLADA, yang dapat digunakan dalam berbagai aplikasi simulasi.

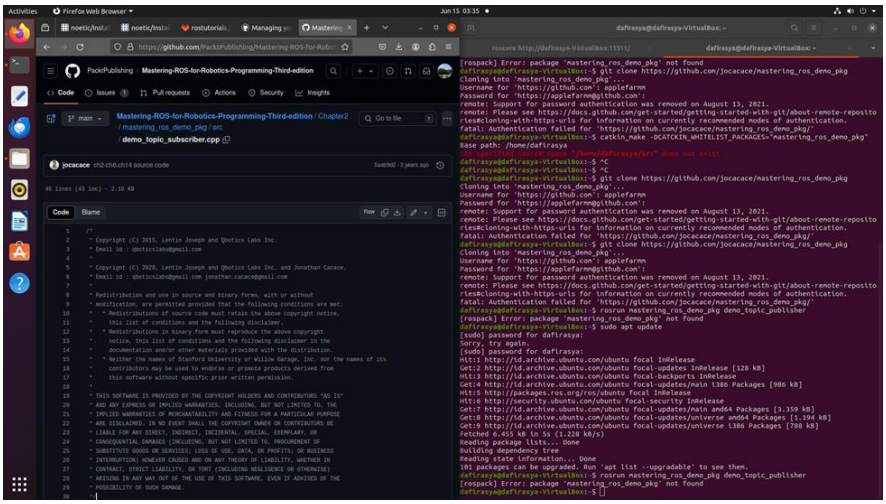
Paket-paket ini membantu kita dalam membuat deskripsi model 3D robot yang sesuai dengan karakteristik perangkat keras sebenarnya, memungkinkan simulasi yang lebih akurat dan pengendalian yang efektif.

*note: terdapat masalah (untuk seluruh bab ini) yang dialami yaitu ada error dalam “mastering_ros_demo_pkg” karena paket tersebut “*is not found*”.

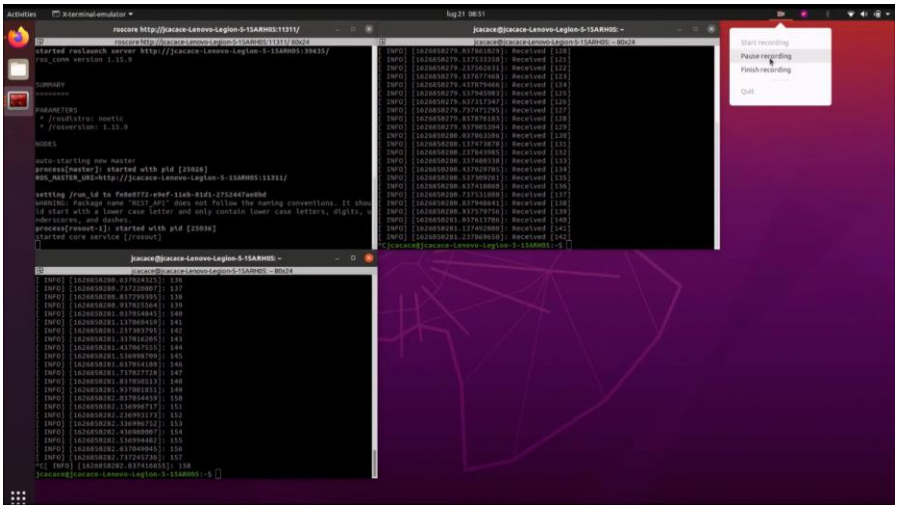
Berikut merupakan setelah mengikuti bagian topic publisher dan subscriber*:



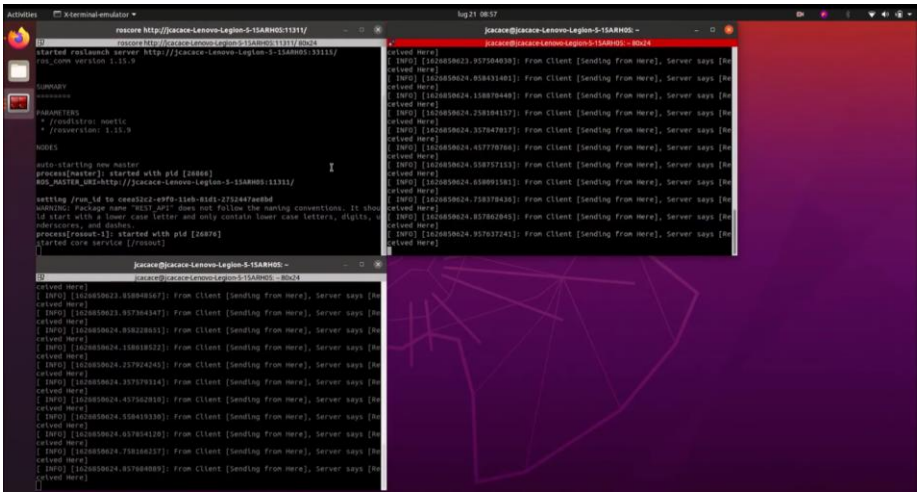
Hasil error saya seperti ini:



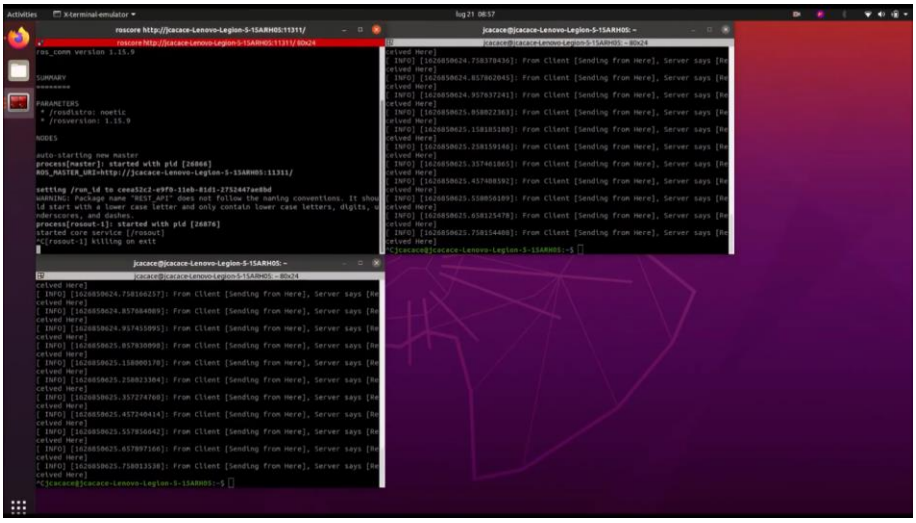
Untuk contoh hasil akhirnya akan terlihat seperti ini;



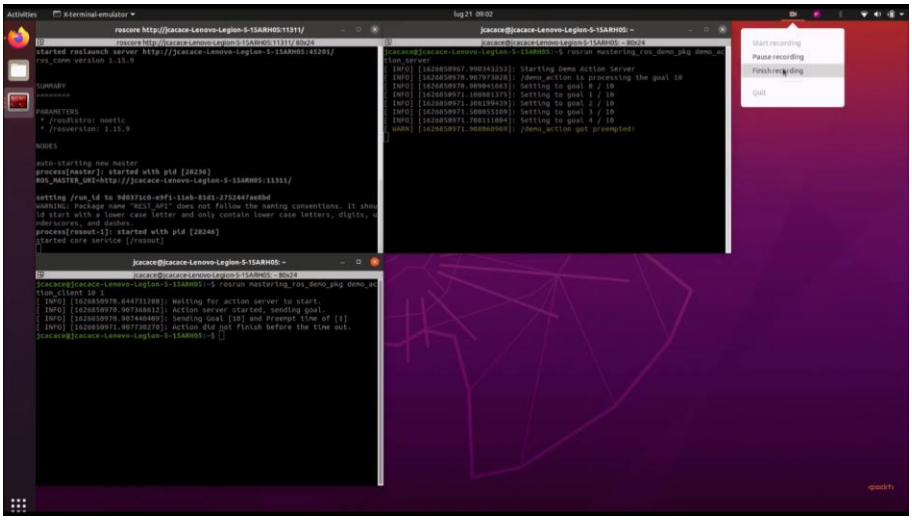
Ini merupakan contoh setelah mengikuti bagian msg publisher dan subscriber*:



Ini merupakan contoh setelah mengikuti bagian service server dan client*:



Ini merupakan contoh setelah mengikuti bagian action server dan client*:



BAB III - *Working with ROS for 3D Modelling*

Dalam bab ini, kita akan mempelajari cara menggunakan ROS (*Robot Operating System*) untuk pemodelan 3D robot. Fokus utama adalah memahami dan menggunakan *Unified Robot Description Format* (URDF) serta alternatifnya, yaitu Xacro.

1.1 Pengenalan URDF

URDF (*Unified Robot Description Format*) adalah format file yang digunakan untuk mendeskripsikan model 3D robot dalam ROS. File URDF berisi informasi tentang geometri, kinematika, dan properti fisik robot. Berikut adalah langkah-langkah umum dalam menggunakan URDF:

- **Membuat Model Robot**: Menulis file URDF yang mendeskripsikan robot.
- **Visualisasi di RVIZ**: Menggunakan RVIZ, aplikasi visualisasi di ROS, untuk melihat dan memeriksa model robot yang dibuat.

1.2 Tantangan dan Kendala

Sayangnya, akses ke GitHub untuk mendapatkan contoh dan kode URDF mungkin terbatas atau tidak tersedia. Namun, kita masih bisa merujuk pada buku dan sumber daya lain yang menyediakan kode dan panduan untuk membuat model robot.

1.3 Kekurangan URDF

Meskipun URDF adalah alat yang kuat, ada beberapa keterbatasan, antara lain:

- **Kemudahan Penggunaan**: URDF bisa jadi kompleks dan sulit digunakan untuk pemula.
- **Kemampuan untuk Digunakan Kembali**: URDF kurang fleksibel dalam hal penggunaan ulang kode.
- **Modularitas**: Kurang mendukung pembuatan model yang modular.
- **Kemudahan Pemrograman**: Penulisan dan pemeliharaan file URDF bisa menjadi rumit.

1.4 Pengenalan Xacro

Untuk mengatasi beberapa keterbatasan URDF, ROS menyediakan format alternatif yaitu Xacro (XML Macros). Xacro membantu menyederhanakan penulisan file URDF dengan menggunakan makro, sehingga lebih mudah untuk digunakan dan diprogram. Keunggulan Xacro meliputi:

- **Sederhana dan Mudah Digunakan**: Mempermudah penulisan dan pemeliharaan model robot.
- **Modularitas**: Memungkinkan pembuatan model yang lebih modular dan terorganisir.
- **Kemampuan untuk Digunakan Kembali**: Xacro memungkinkan penggunaan ulang kode yang lebih baik.

Setelah menulis file Xacro, kita dapat mengonversinya menjadi URDF jika diperlukan, sehingga tetap kompatibel dengan alat-alat lain di ROS.

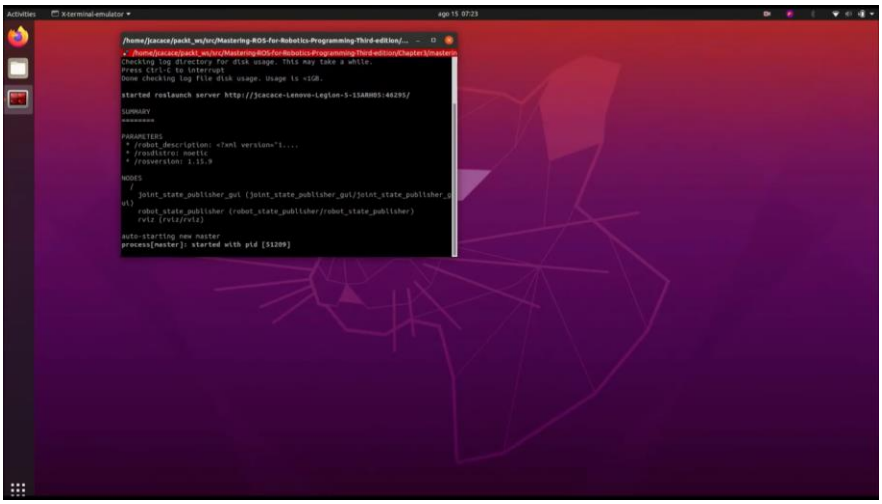
1.5 Implementasi dan Visualisasi

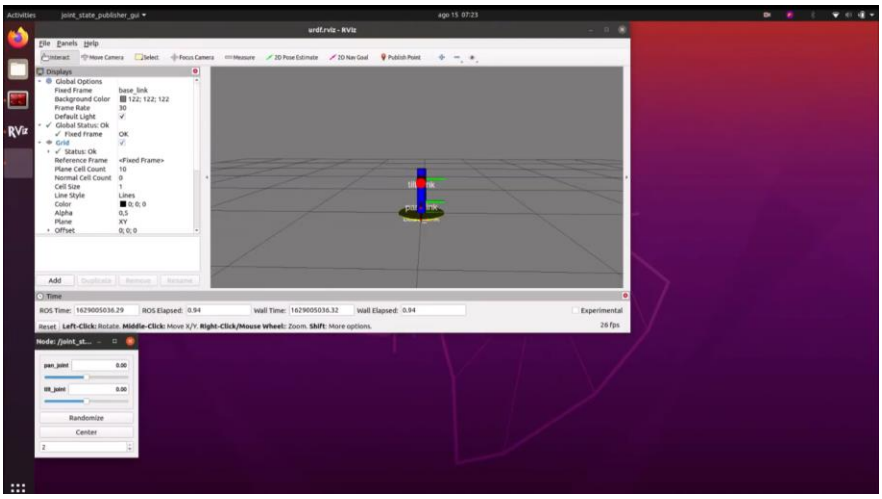
Langkah-langkah untuk menggunakan Xacro dalam pemodelan robot meliputi:

- **Membuat File Xacro**: Menulis deskripsi model robot dalam format Xacro.
- **Mengonversi ke URDF**: Menggunakan perintah ROS untuk mengonversi file Xacro menjadi URDF.
- **Visualisasi di RVIZ**: Memvisualisasikan model robot di RVIZ untuk memastikan desain dan fungsi yang diinginkan.

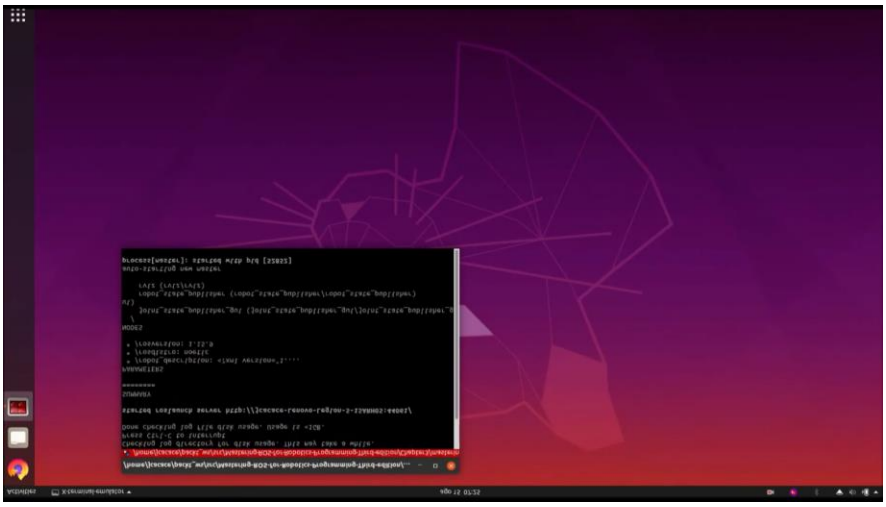
Untuk hasil contoh dari bab ini bisa dilihat berikut ini:

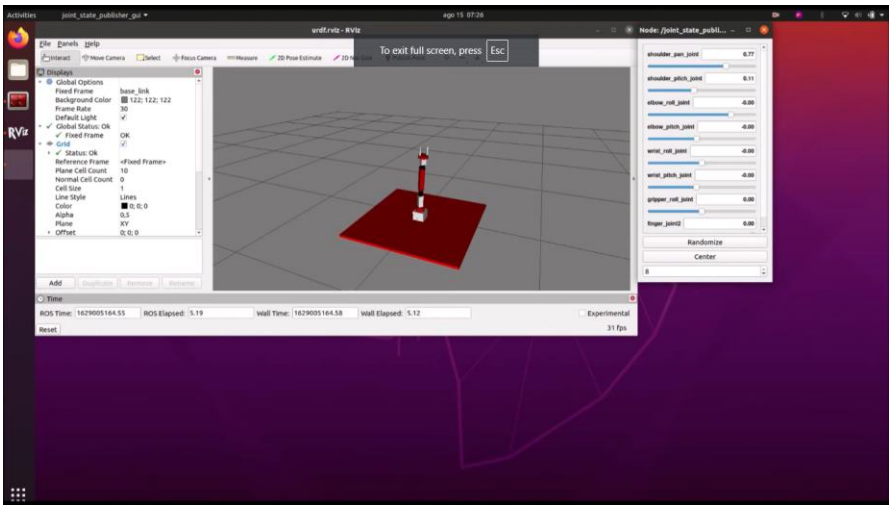
(1)*Visualizing the 3D Robot model in Rviz*



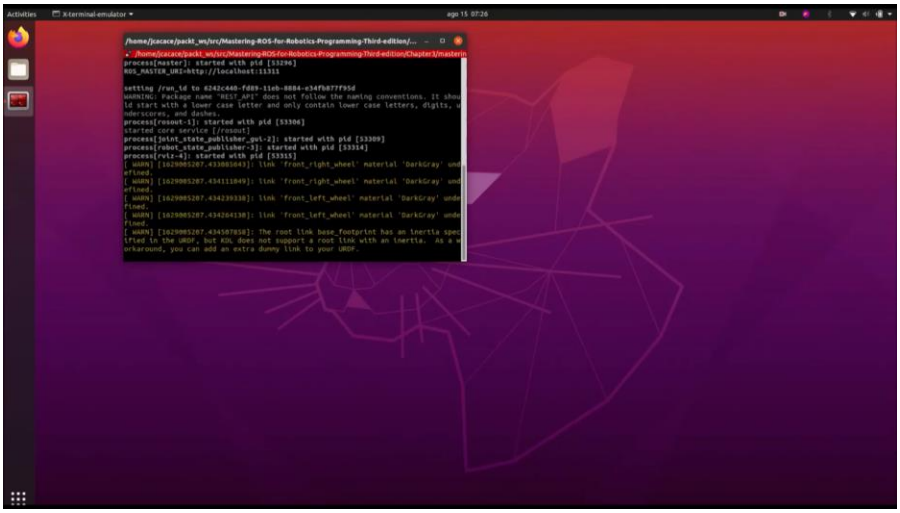


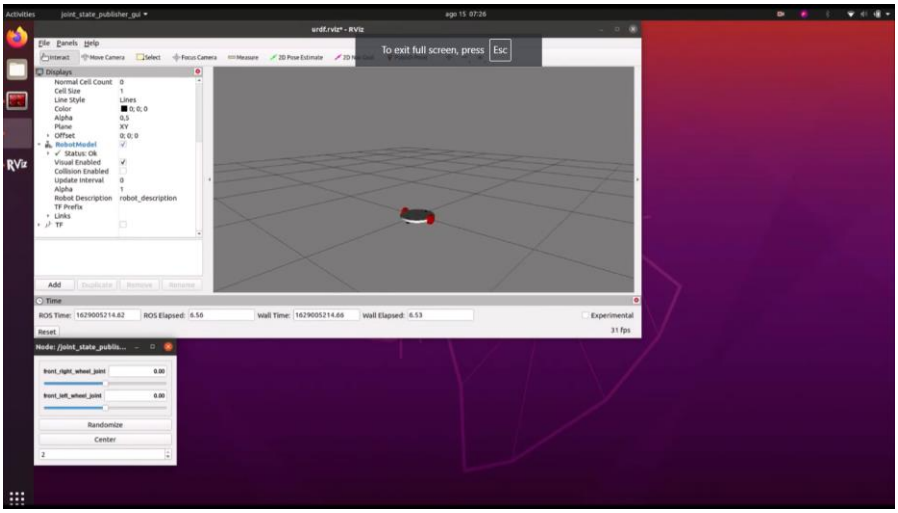
(2) Explaining the XARCO model of the seven-DOF arm





(3)Creating a Robot model for the differential drive mobile





BAB IV - *Simulating Robots Using ROS & Gazebo*

Dalam bab ini, kita akan mempelajari beberapa konsep lanjutan dalam Robot Operating System (ROS), yaitu ROS pluginlib, nodelets, dan plugin Gazebo. Kita akan mengeksplorasi fungsi dan aplikasi masing-masing konsep, serta melihat contoh untuk mendemonstrasikan cara kerjanya.

1.1 ROS Pluginlib

Pluginlib adalah kerangka kerja di ROS yang memungkinkan pembuatan dan penggunaan plugin. Plugin adalah modul yang dapat dimuat secara dinamis, memungkinkan fleksibilitas dan modularitas dalam pengembangan perangkat lunak ROS. Dengan pluginlib, kita dapat membuat komponen perangkat lunak yang bisa diperluas tanpa perlu mengubah kode dasar aplikasi.

1.2 ROS Nodelets

Nodelets adalah bentuk khusus dari ROS node yang memungkinkan beberapa node berjalan dalam satu proses. Keuntungan utama dari nodelets adalah mengurangi overhead komunikasi antarproses, yang bisa meningkatkan performa aplikasi ROS, terutama dalam kasus yang memerlukan komunikasi data yang cepat dan sering.

- **Keuntungan Nodelets**:
    - **Efisiensi Komunikasi**: Menghindari overhead komunikasi antarproses dengan berbagi memori antar-node.
    - **Penggunaan Sumber Daya yang Lebih Baik**: Mengurangi penggunaan CPU dan memori dibandingkan dengan menjalankan node terpisah.

1.3 Plugin Gazebo

Plugin Gazebo digunakan untuk memperluas fungsionalitas simulator Gazebo, yang sering digunakan bersama ROS untuk mensimulasikan robot dan lingkungan mereka. Plugin ini memungkinkan kita untuk menambahkan perilaku khusus, sensor, atau aktuator ke dalam simulasi Gazebo.

- **Penggunaan Plugin Gazebo**:
    - **Simulasi Sensor dan Aktuator**: Menambahkan dan mengkonfigurasi berbagai sensor dan aktuator pada model robot di dalam Gazebo.
    - **Perilaku Kustom**: Menerapkan logika kustom dan perilaku dinamis pada elemen simulasi.

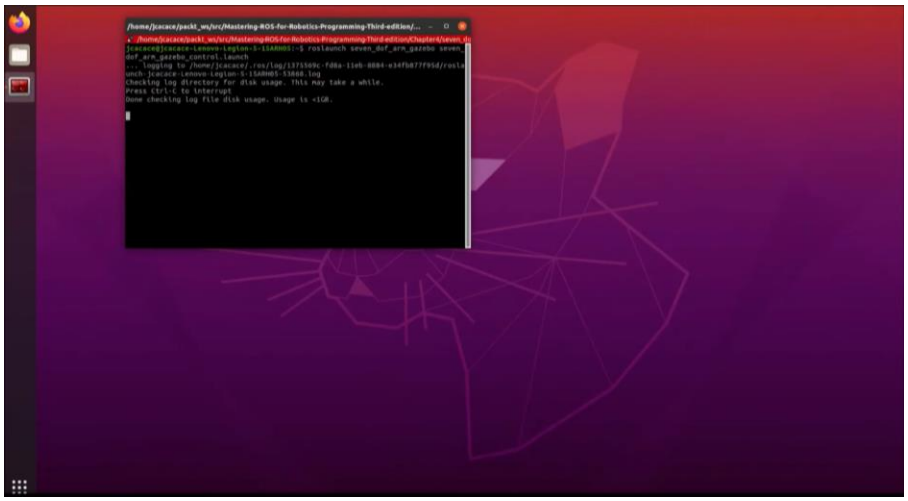
1.4 Implementasi dan Contoh

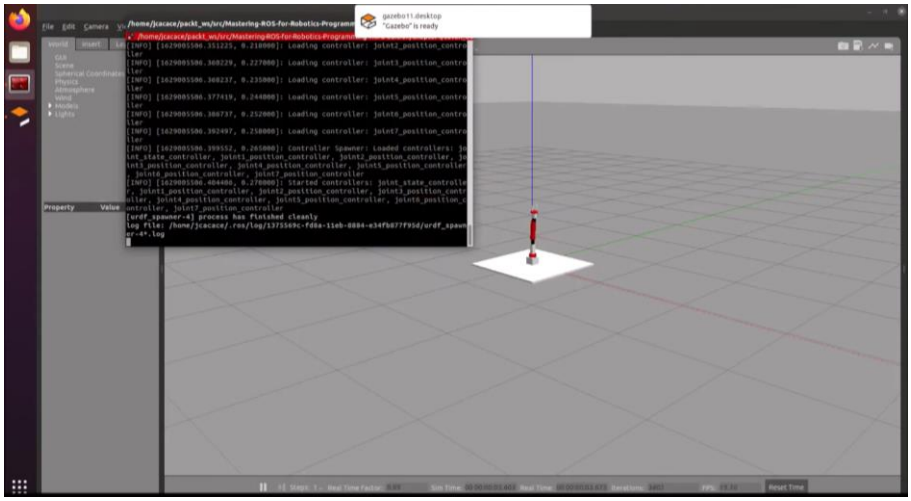
Dalam bab ini, kita juga akan melihat cara membuat dan menggunakan plugin Gazebo serta nodelets dalam ROS:

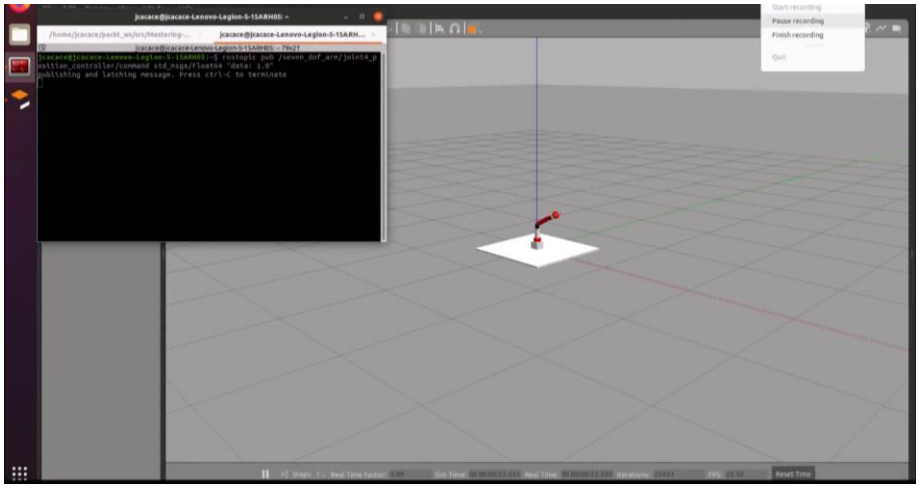
- **Membuat Plugin Gazebo**: Langkah-langkah untuk membuat plugin yang dapat memperluas kemampuan simulasi di Gazebo.
- **Menggunakan Nodelets**: Contoh implementasi nodelets dan cara mengkonfigurasinya untuk meningkatkan performa aplikasi ROS.
- **Contoh Pluginlib**: Studi kasus penggunaan pluginlib untuk membuat sistem yang modular dan fleksibel.

Untuk hasil contoh dari bab ini bisa dilihat berikut ini:

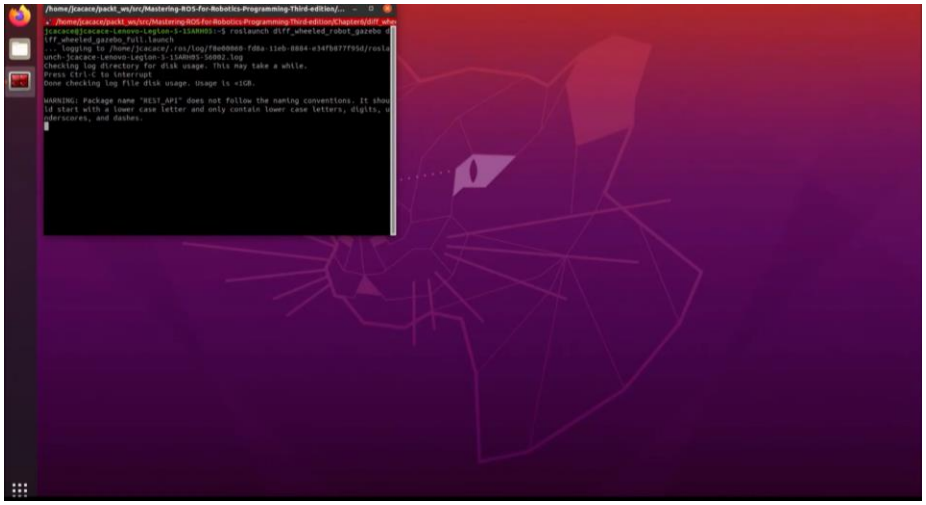
(1)*Moving the robot joints using ROS controllers in Gazebo*

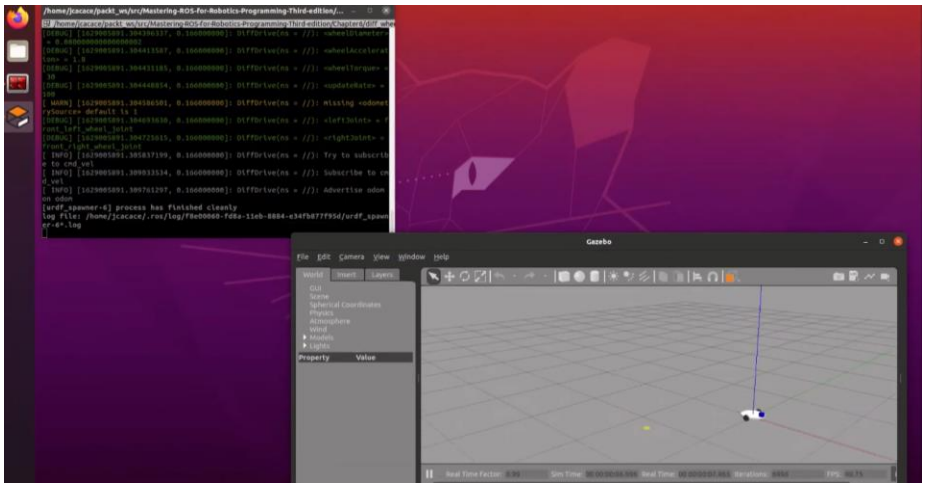


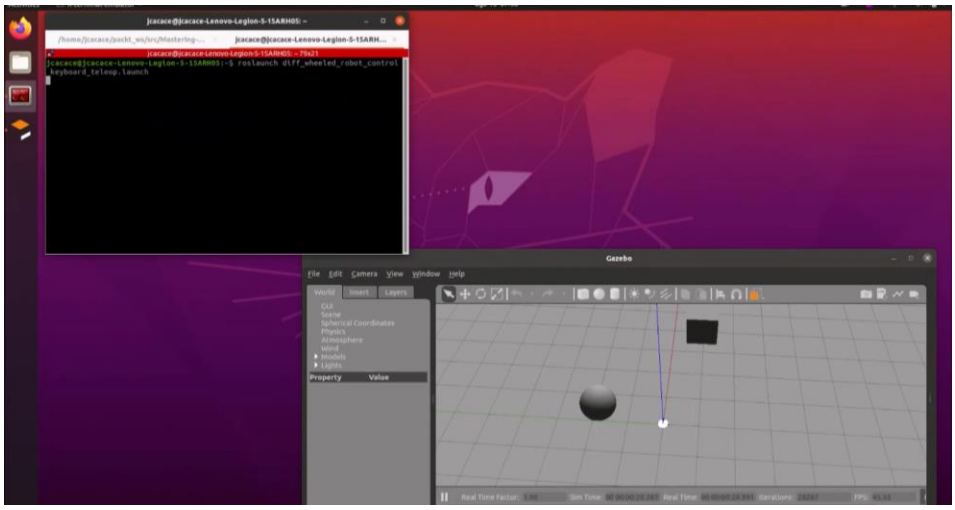


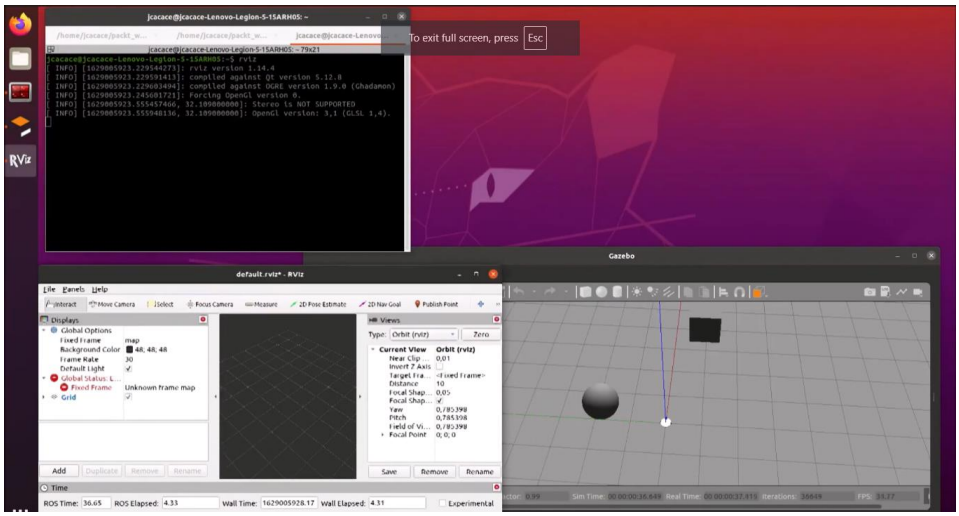


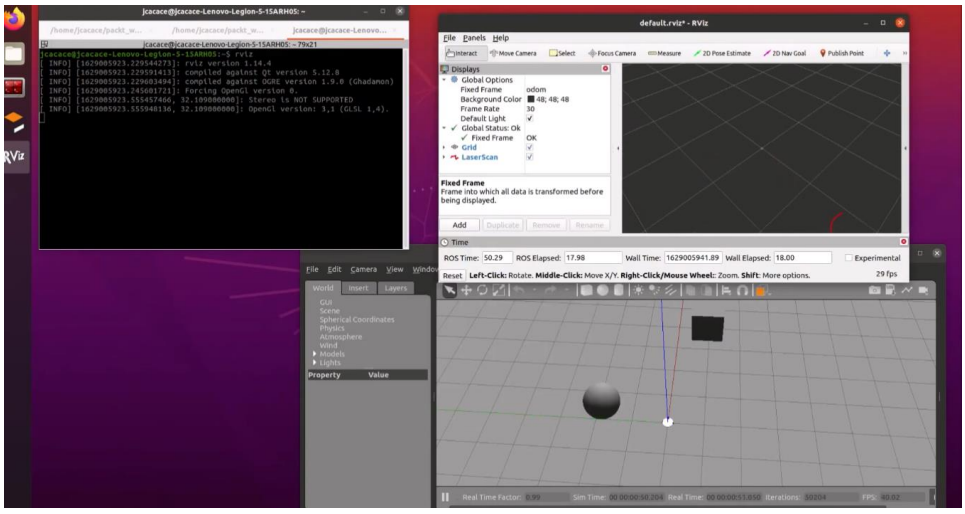
(2) *Adding the ROS teleop node*

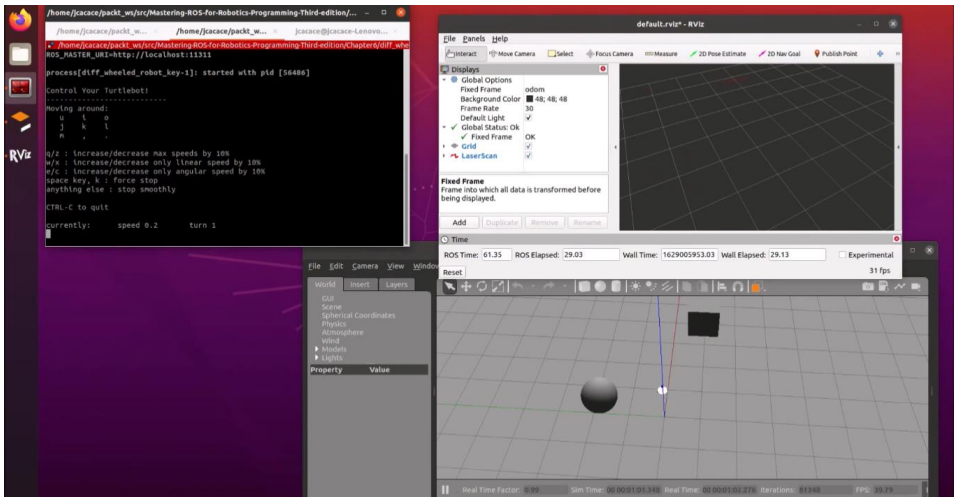





Gazebo


To exit full screen, press Esc
Gazebo
default.rviz* - RViz


default.rviz* - RViz



BAB V - *Simulating Robots Using ROS, CoppeliaSim & Webots*

Simulasi robot adalah langkah penting dalam pengembangan robotika, memungkinkan pengujian dan validasi algoritma kontrol serta perilaku robot dalam lingkungan virtual sebelum diterapkan pada robot fisik. Berikut adalah panduan untuk menggunakan ROS bersama dengan dua simulator populer: CoppeliaSim dan Webots.

1. ROS (Robot Operating System)

ROS adalah kerangka kerja open-source yang menyediakan alat dan pustaka untuk membangun sistem robotika yang kompleks. ROS mendukung integrasi dengan berbagai simulator, termasuk CoppeliaSim dan Webots, memungkinkan simulasi yang realistis dan interaktif.

2. CoppeliaSim

CoppeliaSim (sebelumnya dikenal sebagai V-REP) adalah simulator robotika yang serbaguna dan kuat. CoppeliaSim menyediakan lingkungan yang realistis untuk simulasi robot dengan fitur-fitur berikut:

- **Pemodelan Fisik yang Realistis**: Menggunakan mesin fisika untuk mensimulasikan dinamika robot dan interaksinya dengan lingkungan.
- **Scripting dan API**: Mendukung berbagai bahasa pemrograman dan API untuk mengontrol robot dan lingkungan.
- **Integrasi dengan ROS**: Memungkinkan komunikasi dua arah antara ROS dan CoppeliaSim, sehingga node ROS dapat mengendalikan robot di CoppeliaSim dan sebaliknya.

3. Webots

Webots adalah simulator robotika open-source yang digunakan untuk pendidikan dan penelitian. Webots memiliki beberapa keunggulan, termasuk:

- **Antarmuka Pengguna yang Intuitif**: Memudahkan pemodelan dan simulasi robot.
- **Beragam Model Robot**: Menyediakan banyak model robot yang siap digunakan.
- **Integrasi dengan ROS**: Seperti CoppeliaSim, Webots juga mendukung integrasi dengan ROS, memungkinkan penggunaan node ROS untuk mengontrol robot di dalam simulator.

4. Langkah-langkah Simulasi dengan ROS, CoppeliaSim, dan Webots

4.1 Persiapan

- **Instalasi ROS**: Pastikan ROS terinstal di sistem Anda.
- **Instalasi CoppeliaSim/Webots**: Unduh dan instal simulator yang diinginkan dari situs resminya.

4.2 Konfigurasi

- **Plugin ROS**: Instal plugin yang diperlukan untuk menghubungkan ROS dengan CoppeliaSim atau Webots.
- **Konfigurasi Jaringan**: Pastikan ROS dan simulator dapat berkomunikasi melalui jaringan (misalnya, menggunakan ROS master URI yang benar).

4.3 Pemodelan dan Simulasi

- **Membuat Model Robot**: Buat atau impor model robot ke dalam CoppeliaSim atau Webots.
- **Menghubungkan dengan ROS**: Gunakan plugin ROS untuk menghubungkan simulator dengan ROS.
- **Menjalankan Simulasi**: Mulai simulasi di CoppeliaSim atau Webots dan kontrol robot menggunakan node ROS.

5. Keuntungan Menggunakan Simulasi

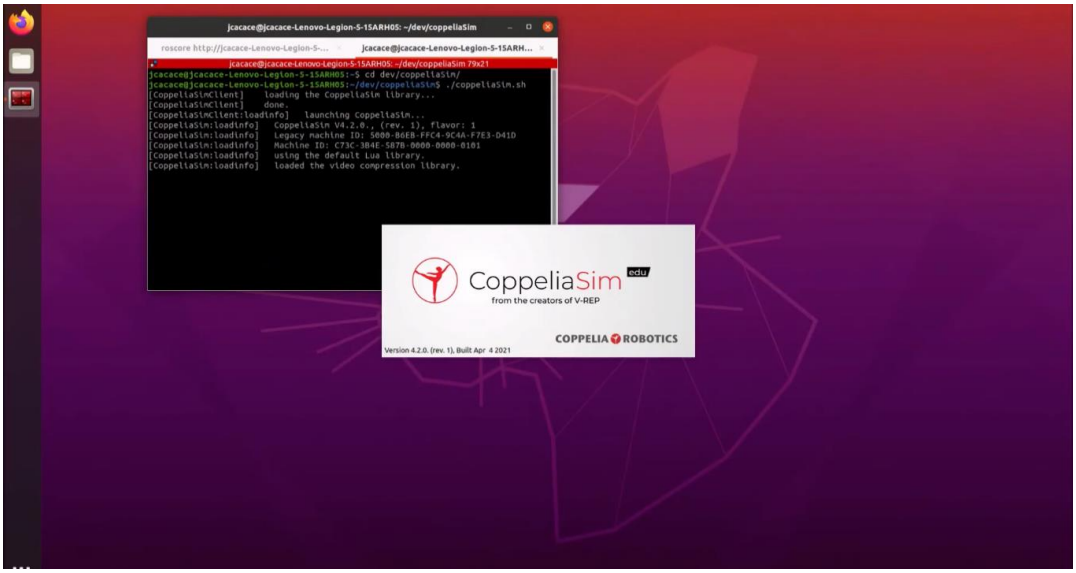
- **Pengujian Awal**: Memungkinkan pengujian awal algoritma dan kontrol tanpa risiko merusak robot fisik.
- **Hemat Biaya dan Waktu**: Mengurangi kebutuhan untuk perangkat keras robot yang mahal dan memungkinkan iterasi yang cepat.
- **Lingkungan yang Terkontrol**: Memungkinkan pengujian dalam berbagai kondisi yang mungkin sulit atau mahal untuk direplikasi di dunia nyata.

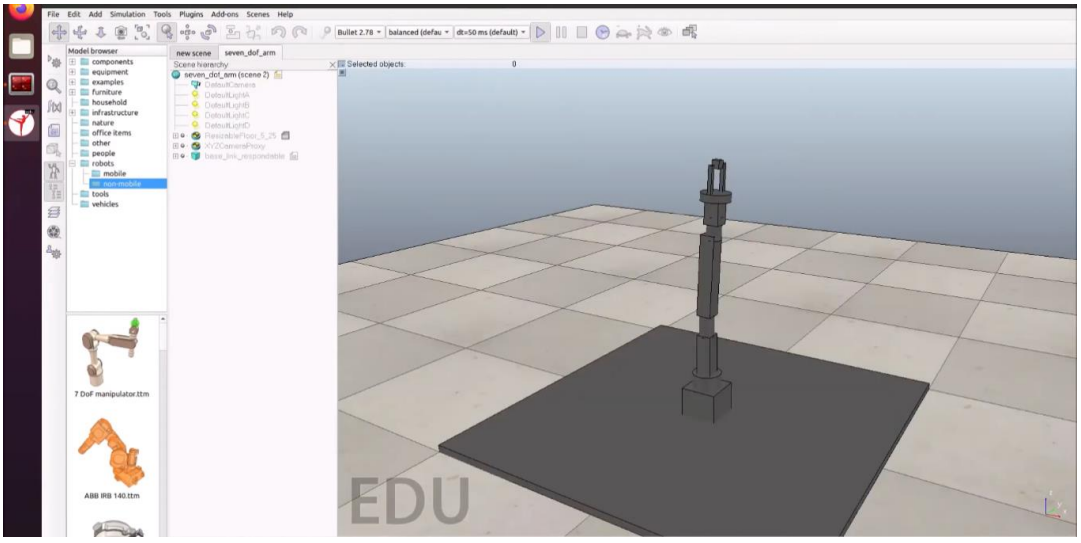
6. Contoh Aplikasi

- **Pemrograman Robot Manipulator**: Menggunakan ROS untuk mengendalikan lengan robot di CoppeliaSim atau Webots.
- **Simulasi Kendaraan Otonom**: Menggunakan ROS untuk mengembangkan dan menguji algoritma navigasi untuk kendaraan otonom dalam simulator.
- **Eksperimen Robotika Pendidikan**: Menggunakan simulator untuk mengajarkan konsep-konsep dasar robotika kepada siswa tanpa memerlukan perangkat keras fisik.

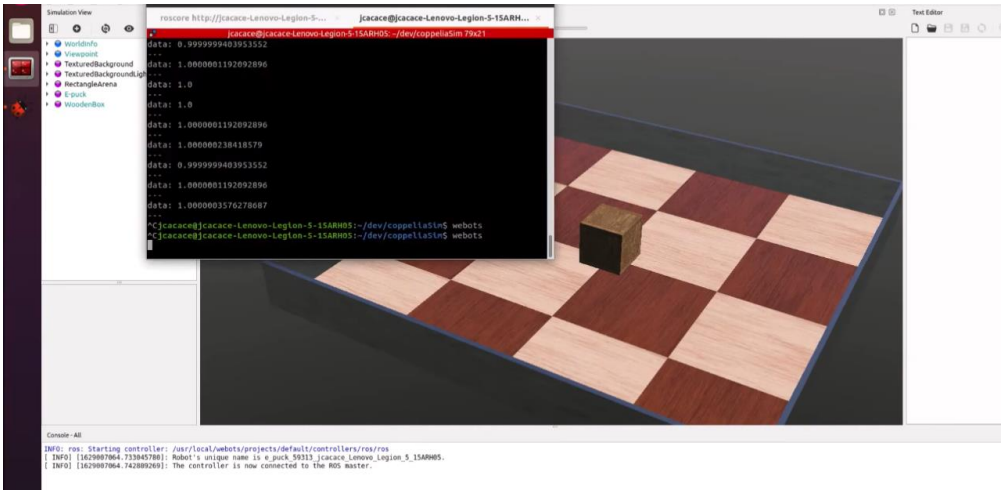
Untuk hasil contoh dari bab ini bisa dilihat berikut ini:

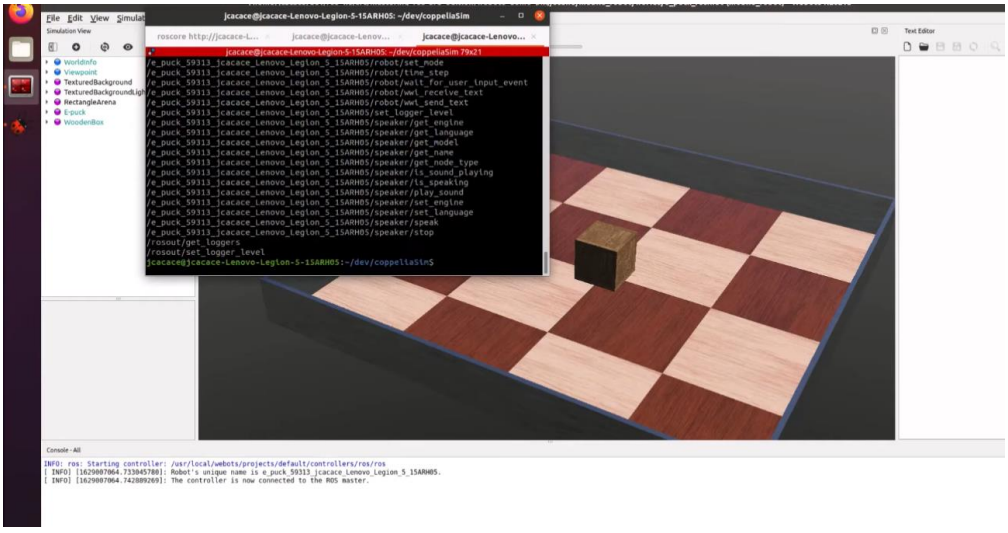
(1)Simulating a robotic arm using CoppeliaSim & ROS

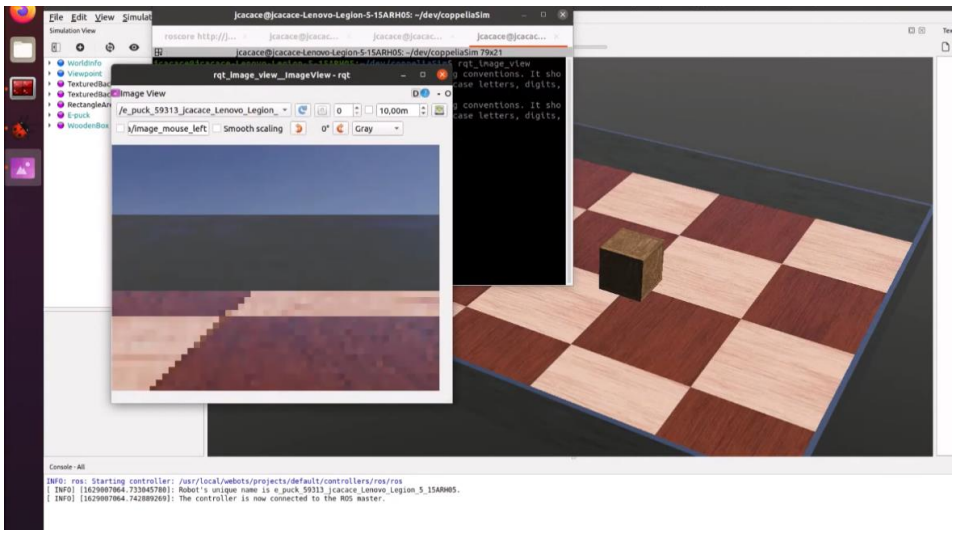


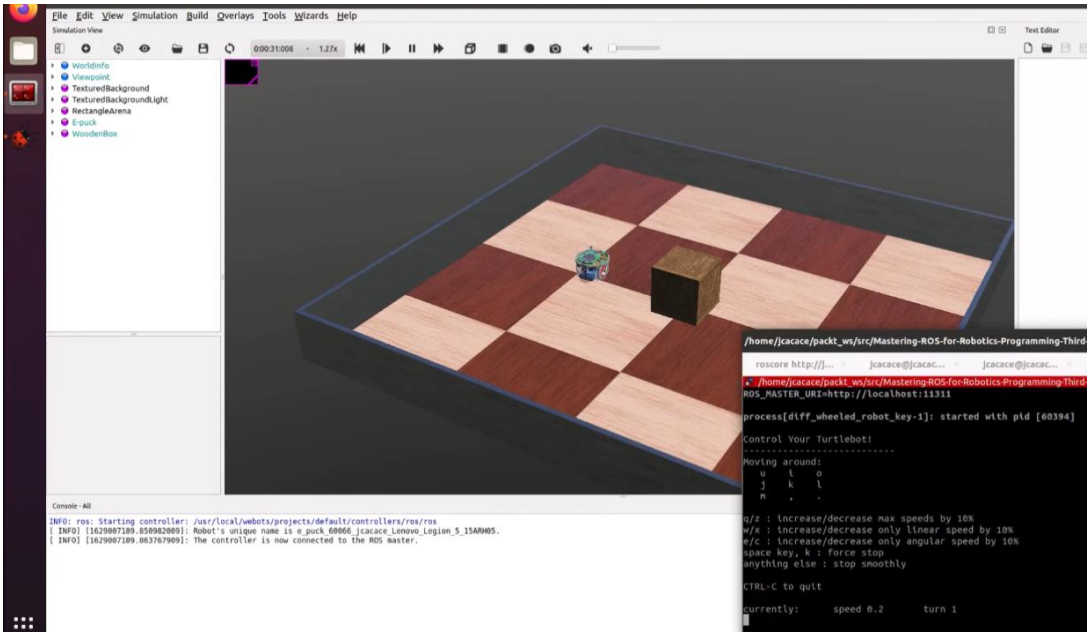


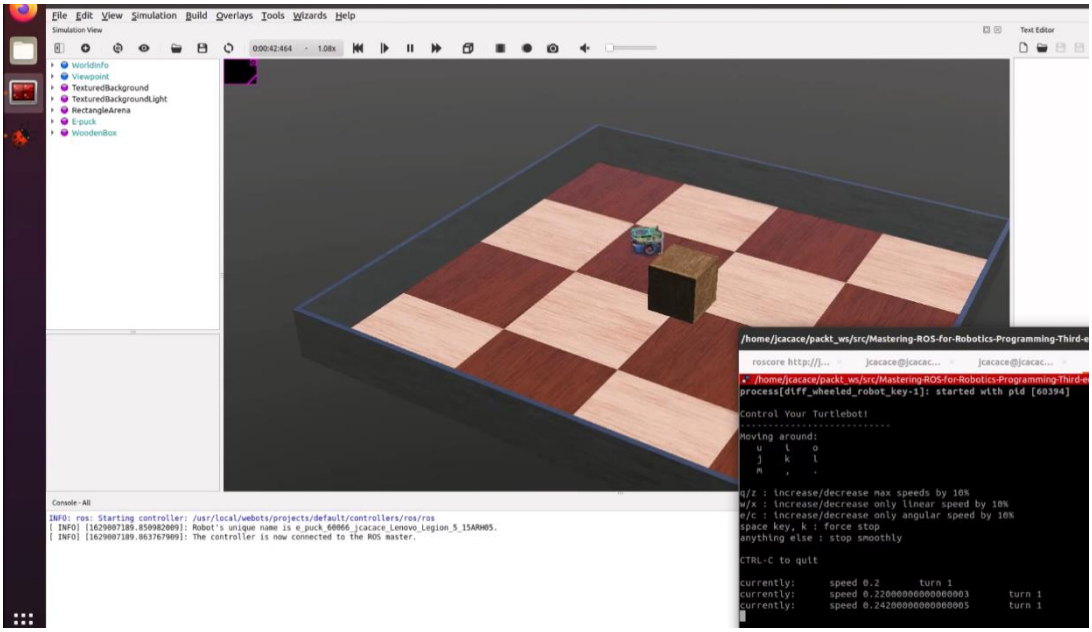
(2)Setting up Webots with ROS











BAB VI - *Using the ROS MoveIt! & Navigation Stack*

Node move_group dalam MoveIt!

Node **move_group** berfungsi sebagai integrator utama dalam sistem MoveIt!, menghubungkan berbagai fungsionalitas melalui plugin untuk pemecahan kinematika dan perencanaan gerak. Berikut adalah rincian mengenai cara kerja dan penggunaan node ini:

1.1 Fungsi Utama move_group

- **Integrasi Fungsionalitas**: Menghubungkan plugin untuk pemecahan kinematika (Inverse Kinematics/IK) dan perencanaan gerak.
- **Perencanaan Gerak**: Setelah lintasan gerak direncanakan, lintasan tersebut dikirim ke pengontrol robot melalui antarmuka **FollowJointTrajectoryAction** untuk dieksekusi baik pada robot fisik maupun dalam simulator seperti Gazebo.

1.2 Integrasi dengan Alat Lain

- **RViz**: MoveIt! dapat diintegrasikan dengan antarmuka grafis RViz untuk visualisasi dan kontrol interaktif.
- **Gazebo**: MoveIt! juga dapat dihubungkan dengan simulator Gazebo untuk simulasi fisik yang lebih realistis.

1.3 Perencanaan Gerak

Perencanaan gerak adalah teknik untuk menemukan jalur optimal yang memungkinkan robot bergerak dari posisi awal ke posisi tujuan tanpa menabrak rintangan. Proses ini membutuhkan:

- **Deskripsi Robot**: Menggunakan file URDF (Unified Robot Description Format) untuk mendeskripsikan robot.
- **Geometri Lingkungan**: Representasi lingkungan di mana robot akan bergerak.

1.4 Penambahan Kendala

Kendala dapat ditambahkan ke perencana gerak untuk mengontrol berbagai aspek pergerakan robot, seperti:

- **Posisi dan Orientasi**: Kendala pada posisi dan orientasi akhir robot.
- **Visibilitas**: Kendala yang memastikan robot tetap terlihat oleh sensor.
- **Batas Sendi**: Batasan pada rentang gerakan sendi robot.
- **Kendala Khusus**: Kendala khusus yang ditentukan oleh pengguna.

1.5 Adaptor Permintaan Perencanaan Gerak

Adaptor ini melakukan praproses dan pascaproses pada permintaan dan respons perencanaan gerak, memastikan bahwa jalur yang dihasilkan sesuai dengan kendala dan persyaratan sistem.

1.6 Fleksibilitas MoveIt!

MoveIt! menawarkan fleksibilitas tinggi dengan memungkinkan pengguna mengganti algoritma IK menggunakan plugin. Selain itu, MoveIt! telah mengintegrasikan pemecahan kinematika maju (Forward Kinematics/FK) dan perhitungan Jacobian.

1.7 Pemeriksaan Tabrakan

- **CollisionWorld**: Objek ini digunakan untuk melakukan pemeriksaan tabrakan, mendukung berbagai jenis objek dan lingkungan.
- **Matriks Tabrakan yang Diizinkan (ACM)**: Matriks ini digunakan untuk mengoptimalkan komputasi pemeriksaan tabrakan, membantu sistem menentukan tabrakan yang perlu diperiksa.

1.8 Menghubungkan Lengan Robot dengan MoveIt!

Untuk menghubungkan lengan robot dengan MoveIt!, node **move_group** memerlukan:

- **URDF dan SRDF**: Deskripsi robot dan semantik.
- **File Konfigurasi**: File yang mengatur parameter dan kendala.
- **Topik Status Sendi dan Informasi TF**: Untuk mengetahui status sendi dan transformasi frame.

1.9 Setup Assistant

MoveIt! Setup Assistant adalah alat yang membantu menghasilkan semua elemen yang diperlukan untuk paket konfigurasi MoveIt!, termasuk file URDF, SRDF, konfigurasi, dan topik yang diperlukan.