

Nama : Muhammad Farrel Ahadi Tama
Kelas : TK-45-01
NIM : 1103210177

BAB 1

Introduction to ROS

ROS adalah kerangka kerja fleksibel untuk menulis perangkat lunak robotik, menyediakan alat dan pustaka untuk berbagai tugas seperti pengiriman pesan, komputasi terdistribusi, dan implementasi algoritma canggih. Proyek ini dimulai pada tahun 2007 oleh Morgan Quigley dan dikembangkan lebih lanjut di Willow Garage. ROS bertujuan untuk menstandarkan pemrograman robot dan menyediakan komponen perangkat lunak yang dapat digunakan kembali.

Fitur Utama ROS:

1. Kemampuan Tinggi:
 - Fungsionalitas siap pakai seperti SLAM dan AMCL untuk navigasi otonom dan MoveIt untuk perencanaan gerakan.
 - Paket-paket ini sangat dapat dikonfigurasi untuk memenuhi berbagai aplikasi robotik.
2. Banyak Alat:
 - Alat seperti rqt_gui, RViz, dan Gazebo untuk debugging, visualisasi, dan simulasi.
 - Alat-alat ini kuat dan sumber terbuka, jarang ditemukan di kerangka kerja perangkat lunak lainnya.

ROS lebih dari sekedar kerangka pengembangan; dapat disebut sebagai meta-OS karena menawarkan alat dan pustaka, serta fungsi seperti sistem operasi, seperti abstraksi perangkat keras, manajemen paket, dan alat pengembang. ROS adalah pilihan yang kuat untuk pengembangan perangkat lunak robotik karena kemampuannya yang tinggi, alat yang ekstensif, dukungan untuk sensor dan aktuator, modularitas, dan komunitas yang berkembang pesat.

BAB 2

Getting Started with ROS Programming

Paket ROS adalah unit dasar dari program ROS. Untuk membuat dan membangun paket ROS, kita menggunakan sistem build catkin. ROS distribusi yang saat ini digunakan adalah Noetic Ninjemys. Sistem build bertanggung jawab untuk menghasilkan target (eksekusi/pustaka) dari kode sumber yang dapat digunakan oleh pengguna akhir.

Langkah pertama adalah membuat workspace catkin:

1. Buat direktori workspace:

```
mkdir -p ~/catkin_ws/src
```

2. Source lingkungan ROS untuk mendapatkan akses ke fungsi ROS:

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

3. Pindah ke direktori src:

```
cd ~/catkin_ws/src
```

4. Inisialisasi workspace catkin baru:

```
catkin_init_workspace
```

Meskipun belum ada paket yang dibuat, workspace ini sudah bisa dibangun dengan:

1. Pindah ke direktori workspace:

```
cd ~/catkin_ws
```

2. Jalankan perintah untuk membangun workspace:

```
catkin_make
```

Perintah ini akan membuat direktori devel dan build di dalam workspace catkin. Untuk menambahkan workspace ROS yang dibuat ke lingkungan ROS, kita perlu source file setup:

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
```

```
source ~/.bashrc
```

Setelah workspace catkin disiapkan, kita dapat membuat paket ROS kita sendiri dengan perintah:

```
catkin_create_pkg package_name [dependency1] [dependency2]
```

Setelah paket dibuat, tambahkan dependencies tambahan dengan mengedit file CMakeLists.txt dan package.xml. Untuk membangun paket:

```
cd ~/catkin_ws && catkin_make
```

Kemudian kita bisa mulai menambahkan node ke dalam folder src dari paket ini. Topik ROS digunakan sebagai metode komunikasi antara node ROS, memungkinkan mereka untuk berbagi informasi secara terus-menerus. Contoh kode untuk node penerbit topik (demo_topic_publisher.cpp) yang menerbitkan nilai integer ke topik /numbers dan node penerima topik (demo_topic_subscriber.cpp) dibahas di bagian ini.

```
farrel@ubuntu:~$ rosrun mastering_ros_demo_pkg demo_topic_publisher
[rosrun] Couldn't find executable named demo_topic_publisher below /home/farrel/catkin_ws/src/mastering_ros_demo_pkg
farrel@ubuntu:~$
```

```
tf2_py /opt/ros/noetic/share/tf2_py
tf2_ros /opt/ros/noetic/share/tf2_ros
tf_conversions /opt/ros/noetic/share/tf_conversions
theora_image_transport /opt/ros/noetic/share/theora_image_transport
topic_tools /opt/ros/noetic/share/topic_tools
trajectory_msgs /opt/ros/noetic/share/trajectory_msgs
transmission_interface /opt/ros/noetic/share/transmission_interface
turtle_actionlib /opt/ros/noetic/share/turtle_actionlib
turtle_tf /opt/ros/noetic/share/turtle_tf
turtle_tf2 /opt/ros/noetic/share/turtle_tf2
turtlesim /opt/ros/noetic/share/turtlesim
urdf /opt/ros/noetic/share/urdf
urdf_parser_plugin /opt/ros/noetic/share/urdf_parser_plugin
urdf_sdl_tutorial /opt/ros/noetic/share/urdf_sdl_tutorial
urdf_tutorial /opt/ros/noetic/share/urdf_tutorial
urdfdom_py /opt/ros/noetic/share/urdfdom_py
visualization_marker_tutorials /opt/ros/noetic/share/visualization_marker_tutorials
visualization_msgs /opt/ros/noetic/share/visualization_msgs
warehouse_ros /opt/ros/noetic/share/warehouse_ros
webkit_dependency /opt/ros/noetic/share/webkit_dependency
xacro /opt/ros/noetic/share/xacro
xnirpcpp /opt/ros/noetic/share/xnirpcpp
farrel@ubuntu:~$ roscore
... logging to /home/farrel/.ros/log/43e05742-2b08-11ef-9bbb-95fb471d137d/roslaunch-ubuntu-4993.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is 1GB.

started roslaunch server http://ubuntu:45987/
ros_comm version 1.16.0

SUMMARY
=====
PARAMETERS
 * /roscpp: noetic
 * /rosversion: 1.16.0

NODES
auto-starting new master
process[master]: started with pid [5003]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to 43e05742-2b08-11ef-9bbb-95fb471d137d
process[roscout-1]: started with pid [5013]
started core service [/roscout]
$
```

```
farrel@ubuntu:~/catkin_ws$ cd
farrel@ubuntu:~$ git clone https://github.com/PackPublishing/mastering-ROS-for-Robotics-Programming-Third-edition
Cloning into 'mastering-ROS-for-Robotics-Programming-Third-edition' ...
remote: Enumerating objects: 922, done.
remote: Counting objects: 100% (922/922), done.
remote: Compressing objects: 100% (620/620), done.
remote: Total 922 (delta 301), reused 847 (delta 242), pack-reused 0
Receiving objects: 100% (922/922), 6.60 MiB | 4.90 MiB/s, done.
Resolving deltas: 100% (301/301), done.
farrel@ubuntu:~$ catkin_make
Base path: /home/farrel
$
```

Tahapan di atas menunjukkan beberapa contoh dalam membuat dan menjalankan program ROS. Pertama, kita membuat paket ROS sebagai unit dasar dari sebuah program ROS, yang akan digunakan oleh robot untuk menjalankan aksi yang telah ditentukan dalam paket tersebut. Selanjutnya, ada beberapa contoh untuk membuat topik sebagai alat komunikasi antar node ROS, membuat pesan kustom untuk komunikasi, layanan untuk robot tersebut, dan aksi untuk mendapatkan tujuan, umpan balik, serta hasil dari robot. Namun, `mastering_ros_demo_pkg` tidak dapat terbaca saat mencoba menjalankan `roslaunch mastering_ros_demo_pkg demo_menjalankan`, meskipun paket tersebut sudah ada di daftar `rospack list`. Selain itu, hal ini juga terjadi saat mencoba menjalankan `roslaunch` untuk paket lain.

BAB 3

Working with ROS for 3D Modelling

ROS menyediakan beberapa paket yang baik untuk membangun model robot 3D. Paket-paket utama yang digunakan untuk pemodelan robot adalah:

- **urdf**: Paket utama untuk memodelkan robot menggunakan format XML. Paket ini memiliki beberapa komponen seperti `urdf_parser_plugin`, `urdfdom_headers`, `collada_parser`, dan `urdfdom`.
- **joint_state_publisher**: Digunakan untuk membaca deskripsi model robot dan menerbitkan nilai joint.
- **joint_state_publisher_gui**: Mirip dengan `joint_state_publisher`, tetapi dengan antarmuka GUI untuk interaksi dengan joint robot.
- **kdl_parser**: Parser untuk membangun pohon KDL dari model URDF robot.
- **robot_state_publisher**: Membaca status joint robot saat ini dan menerbitkan pose 3D dari setiap link robot.
- **xacro**: Berfungsi untuk membuat file URDF lebih ringkas dan mudah dibaca.

Setelah memahami paket-paket yang digunakan dalam pemodelan 3D robot, kita dapat menganalisis model pertama menggunakan format file URDF. URDF digunakan untuk menggambarkan deskripsi kinematik dan dinamik robot, serta model visual dan tabrakan robot. Beberapa tag utama dalam URDF adalah:

- **link**: Mewakili satu link dari robot, termasuk properti dinamis, visual, dan tabrakan.
- **joint**: Mewakili joint robot, termasuk kinematika, dinamika, dan batas pergerakan joint.
- **robot**: Mengenkapsulasi seluruh model robot, termasuk link dan joint.
- **gazebo**: Digunakan untuk menyertakan parameter simulasi untuk simulator Gazebo.

Selanjutnya, untuk membuat deskripsi robot, kita membuat paket ROS dalam workspace catkin:

```
catkin_create_pkg mastering_ros_robot_description_pkg roscpp tf geometry_msgs urdf
rviz xacro
```

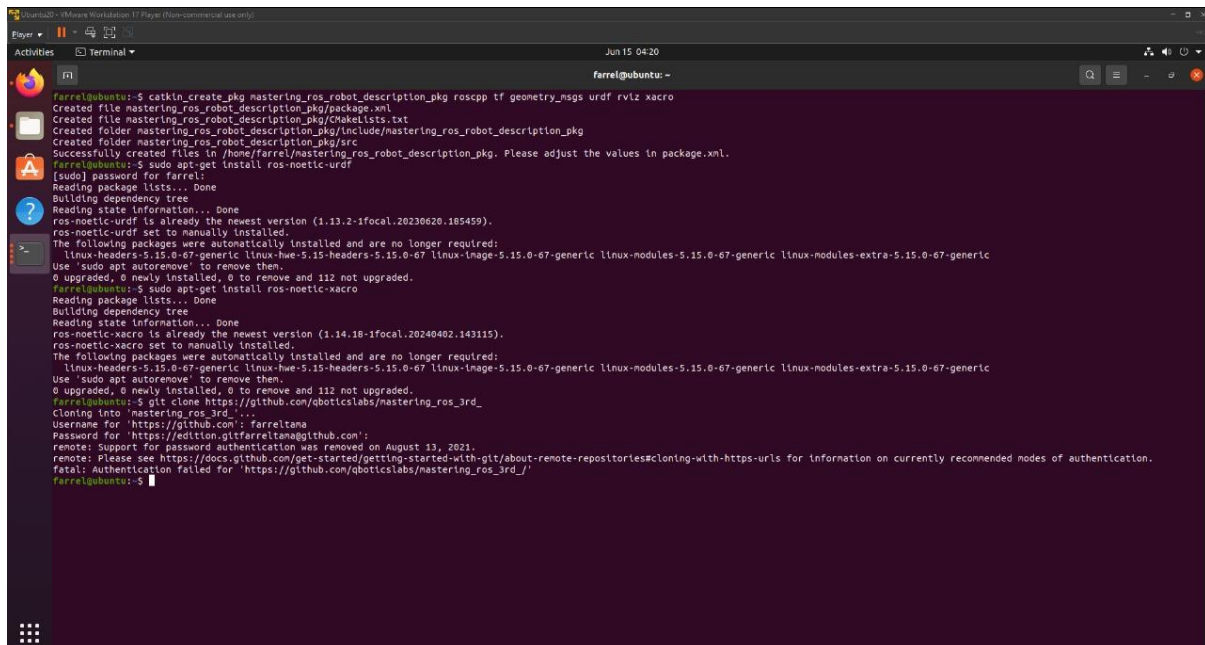
Jika paket urdf dan xacro belum terinstal, kita bisa menginstalnya menggunakan manajer paket:

```
sudo apt-get install ros-noetic-urdf
sudo apt-get install ros-noetic-xacro
```

Kita kemudian membuat file URDF robot dalam paket ini dan membuat file launch untuk menampilkan file URDF yang dibuat di RViz. Paket lengkap tersedia di repositori Git berikut:

```
git clone https://github.com/qboticslabs/mastering\_ros\_3rd\_edition.git
cd mastering_ros_robot_description_pkg/
```

Sebelum membuat file URDF, kita membuat tiga folder bernama urdf, meshes, dan launch di dalam folder paket. Folder urdf untuk menyimpan file URDF dan xacro, folder meshes untuk menyimpan mesh yang akan dimasukkan dalam file urdf, dan folder launch untuk menyimpan file launch ROS.



```
farrel@ubuntu: ~$ catkin_create_pkg mastering_ros_robot_description_pkg roscpp tf geometry_msgs urdf rviz xacro
Created file mastering_ros_robot_description_pkg/package.xml
Created file mastering_ros_robot_description_pkg/CMakeLists.txt
Created folder mastering_ros_robot_description_pkg/include/mastering_ros_robot_description_pkg
Created folder mastering_ros_robot_description_pkg/src
Successfully created files in /home/farrel/mastering_ros_robot_description_pkg. Please adjust the values in package.xml.
farrel@ubuntu: ~$ sudo apt-get install ros-noetic-urdf
[sudo] password for farrel:
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-urdf is already the newest version (1.13.2-1focal.20230620.185459).
ros-noetic-urdf set to manually installed.
The following packages were automatically installed and are no longer required:
  linux-headers-5.15.0-67-generic linux-hwe-5.15-headers-5.15.0-67 linux-image-5.15.0-67-generic linux-modules-5.15.0-67-generic linux-modules-extra-5.15.0-67-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 112 not upgraded.
farrel@ubuntu: ~$ sudo apt-get install ros-noetic-xacro
Reading package lists... Done
Building dependency tree
Reading state information... Done
ros-noetic-xacro is already the newest version (1.14.18-1focal.20240402.143115).
ros-noetic-xacro set to manually installed.
The following packages were automatically installed and are no longer required:
  linux-headers-5.15.0-67-generic linux-hwe-5.15-headers-5.15.0-67 linux-image-5.15.0-67-generic linux-modules-5.15.0-67-generic linux-modules-extra-5.15.0-67-generic
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 112 not upgraded.
farrel@ubuntu: ~$ git clone https://github.com/qboticslabs/mastering_ros_3rd_
Cloning into 'mastering_ros_3rd_':
Username for 'https://github.com': farreltama
Password for 'https://edition.git:farreltama@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/qboticslabs/mastering_ros_3rd_/'
farrel@ubuntu: ~$
```

BAB 4

Simulating Robots Using ROS and Gazebo

Di bagian ini, kita akan mensimulasikan robot tersebut di Gazebo menggunakan ROS. Sebelum memulai, instal paket berikut untuk bekerja dengan Gazebo dan ROS:

```
sudo apt-get install ros-noetic-gazebo-ros-pkgs ros-noetic-gazebo-msgs ros-noetic-gazebo-plugins ros-noetic-gazebo-ros-control
```

Versi default yang terinstal dari paket Noetic ROS adalah Gazebo 11.x. Berikut fungsi dari masing-masing paket:

- **gazebo_ros_pkgs:** Berisi pembungkus dan alat untuk menghubungkan ROS dengan Gazebo.

- **gazebo-msgs:** Berisi pesan dan struktur data layanan untuk interfacing dengan Gazebo dari ROS.
- **gazebo-plugins:** Berisi plugin Gazebo untuk sensor, aktuator, dan sebagainya.
- **gazebo-ros-control:** Berisi pengendali standar untuk komunikasi antara ROS dan Gazebo.

Setelah instalasi, cek apakah Gazebo terinstal dengan benar menggunakan perintah berikut:

```
roscore & roslaunch gazebo_ros gazebo
```

Perintah ini akan membuka GUI Gazebo. Jika simulator Gazebo sudah tersedia, kita dapat mulai mengembangkan model simulasi lengan dengan tujuh DOF untuk Gazebo.

Untuk membuat model simulasi lengan robotik, kita dapat memperbarui deskripsi robot yang ada dengan menambahkan parameter simulasi. Buat paket yang diperlukan untuk mensimulasikan lengan robotik dengan perintah berikut:

```
catkin_create_pkg seven_dof_arm_gazebo gazebo_msgs gazebo_plugins gazebo_ros
gazebo_ros_control mastering_ros_robot_description_pkg
```

Atau, paket lengkap tersedia di repositori Git berikut. Anda bisa mengkloning repositori tersebut sebagai referensi untuk mengimplementasikan paket ini, atau mendapatkan paket dari kode sumber buku:

```
git clone https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git
```

```
cd Chapter4/seven_dof_arm_gazebo
```

Anda bisa melihat model simulasi lengkap robot di file `seven_dof_arm.xacro` yang ditempatkan di folder `mastering_ros_robot_description_pkg/urdf/`. File ini berisi tag URDF yang diperlukan untuk simulasi, termasuk definisi bagian tabrakan, inersia, transmisi, joint, link, dan parameter Gazebo.

```

farrel@ubuntu:~$ roscore & roslaunch gazebo_ros gazebo
[1] 35729
... logging to /home/farrel/.ros/log/6cd2dccc-2b0a-11ef-9bbb-95fb471d137d/roslaunch-ubuntu-35729.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is 410B.
started roslaunch server http://ubuntu:35409/
ros_comm version 1.16.0

SUMMARY
=====
PARAMETERS
 * /roscore: roscore
 * /rosversion: 1.16.0

NODES
auto-starting new master
process[master]: started with pid [35755]
ROS_MASTER_URI=http://ubuntu:11311/

setting /run_id to 6cd2dccc-2b0a-11ef-9bbb-95fb471d137d
process[roscout-1]: started with pid [35784]
started core service [/roscout]
[ INFO ] [1718458925.53751330]: Finished loading Gazebo ROS API Plugin.
[ INFO ] [1718458925.538962670]: WaitForService: service [/gazebo/set_physics_properties] has not been advertised, waiting...
[ INFO ] [1718458925.537514720]: WaitForService: service [/gazebo/set_physics_properties] is now available.
[ INFO ] [1718458925.420146600]: Physics dynamic reconfigure ready.
context mismatch in svga_surface_destroy
context mismatch in svga_surface_destroy
context mismatch in svga_surface_destroy
context mismatch in svga_surface_destroy
farrel@ubuntu:~$ catkin_create_pkg seven_dof_arm_gazebo gazebo_msgs gazebo_plugins gazebo_ros gazebo_ros_control mastering_ros_robot_description_pkg
Created file seven_dof_arm_gazebo/package.xml
Created file seven_dof_arm_gazebo/CMakeLists.txt
Successfully created files in /home/farrel/seven_dof_arm_gazebo. Please adjust the values in package.xml.
farrel@ubuntu:~$ git clone https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition.git
fatal: destination path 'Mastering-ROS-for-Robotics-Programming-Third-edition' already exists and is not an empty directory.
farrel@ubuntu:~$ cd Chapter4/seven_dof_arm_gazebo
bash: cd: Chapter4/seven_dof_arm_gazebo: No such file or directory
farrel@ubuntu:~$ cd Chapter4/seven_dof_arm_gazebo
bash: cd: Chapter4/seven_dof_arm_gazebo: No such file or directory
farrel@ubuntu:~$ roslaunch seven_dof_arm_gazebo seven_dof_arm_world.launch
This launch file is deprecated. Please use the package (seven_dof_arm_gazebo) or is (seven_dof_arm_gazebo) a launch file name
The traceback for the exception was written to the log file
farrel@ubuntu:~$

```

BAB 5

Simulating Using ROS, Coppeliasim, and Webots

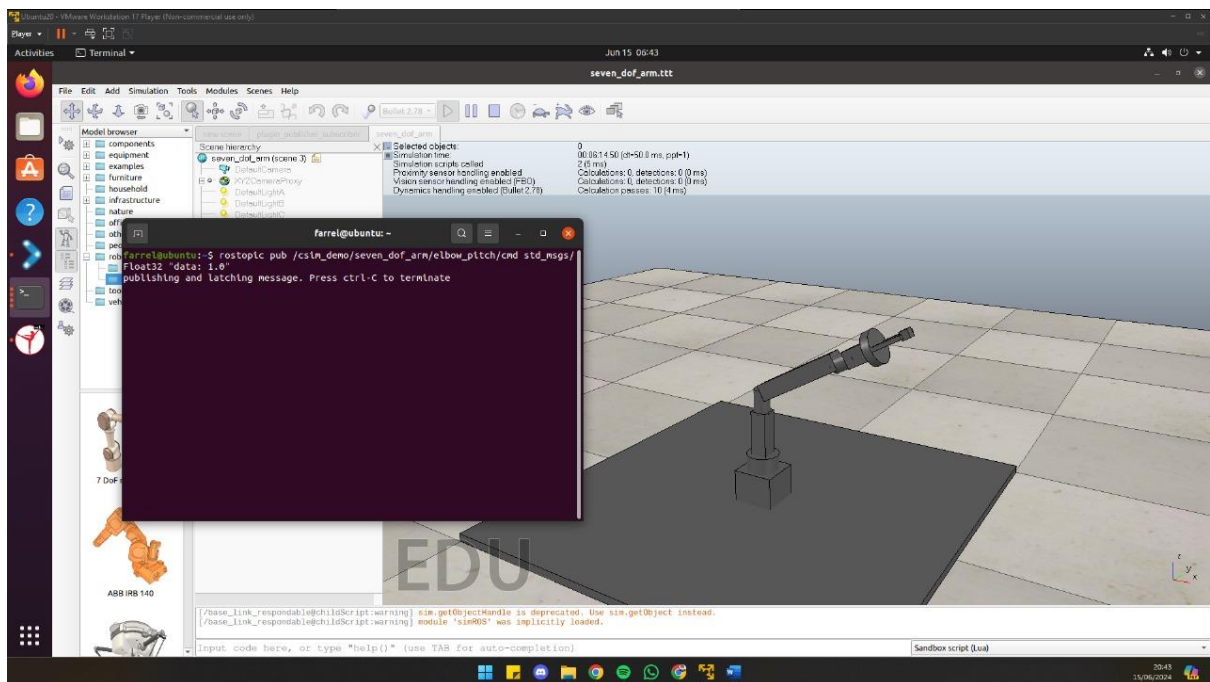
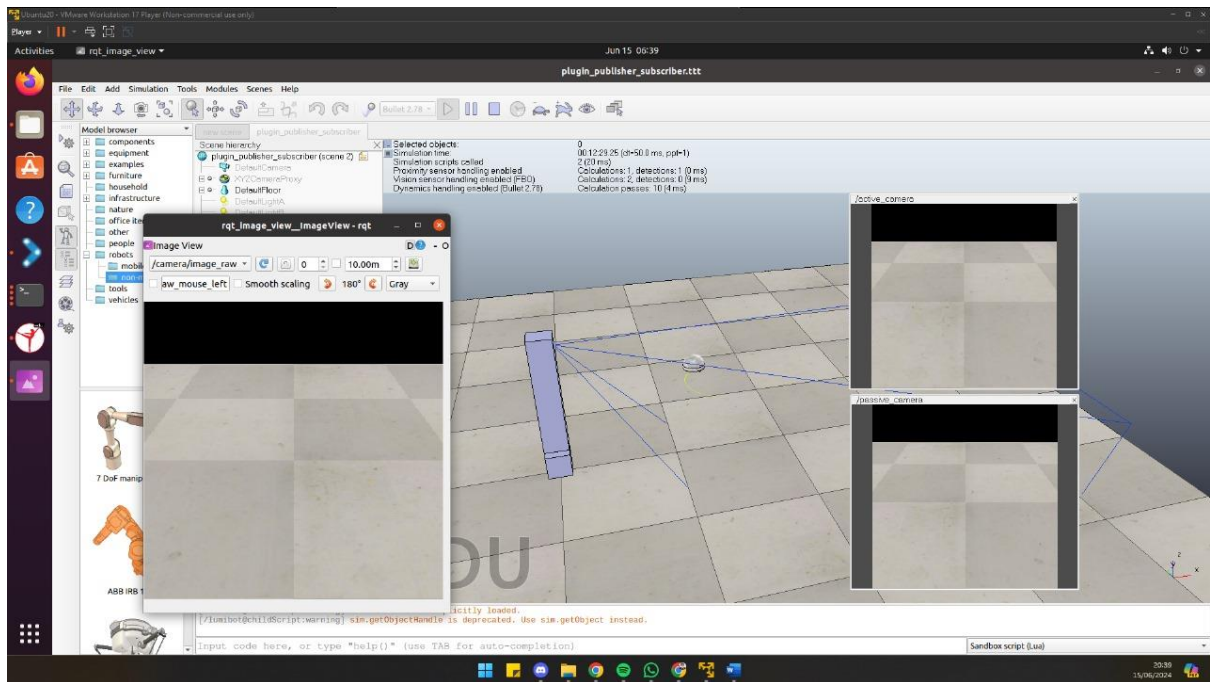
Bab ini menjelaskan cara mensimulasikan robot dengan ROS, Coppeliasim, dan Webots. Topik yang dibahas meliputi:

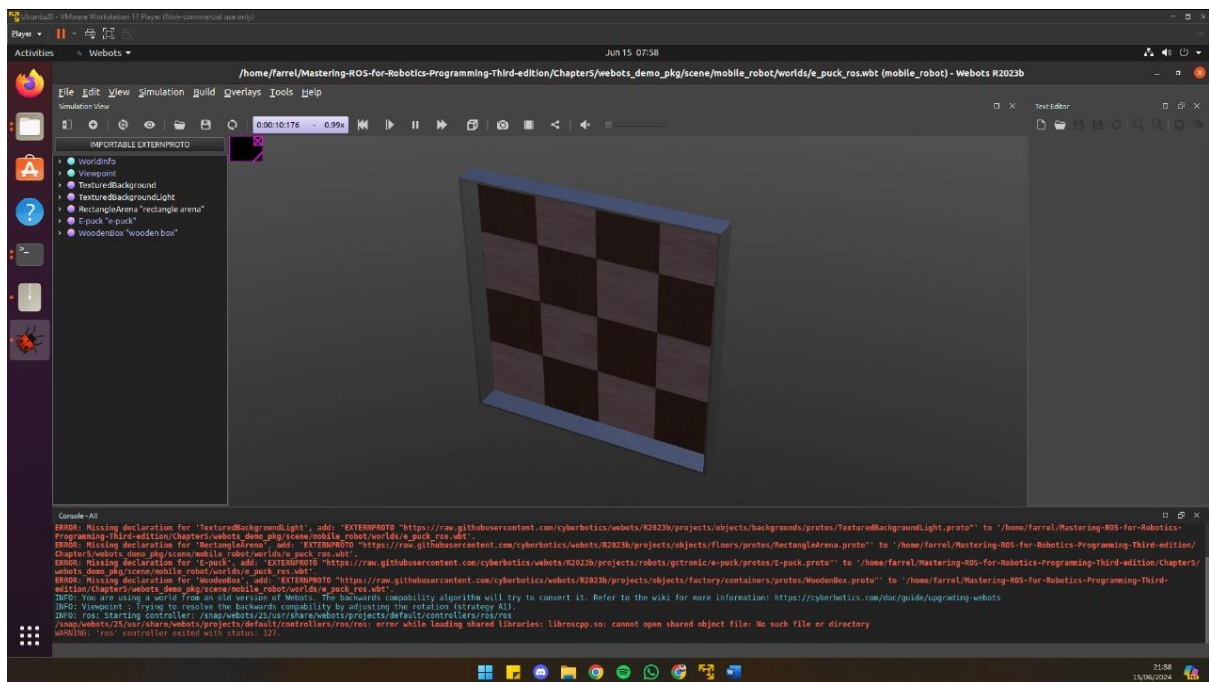
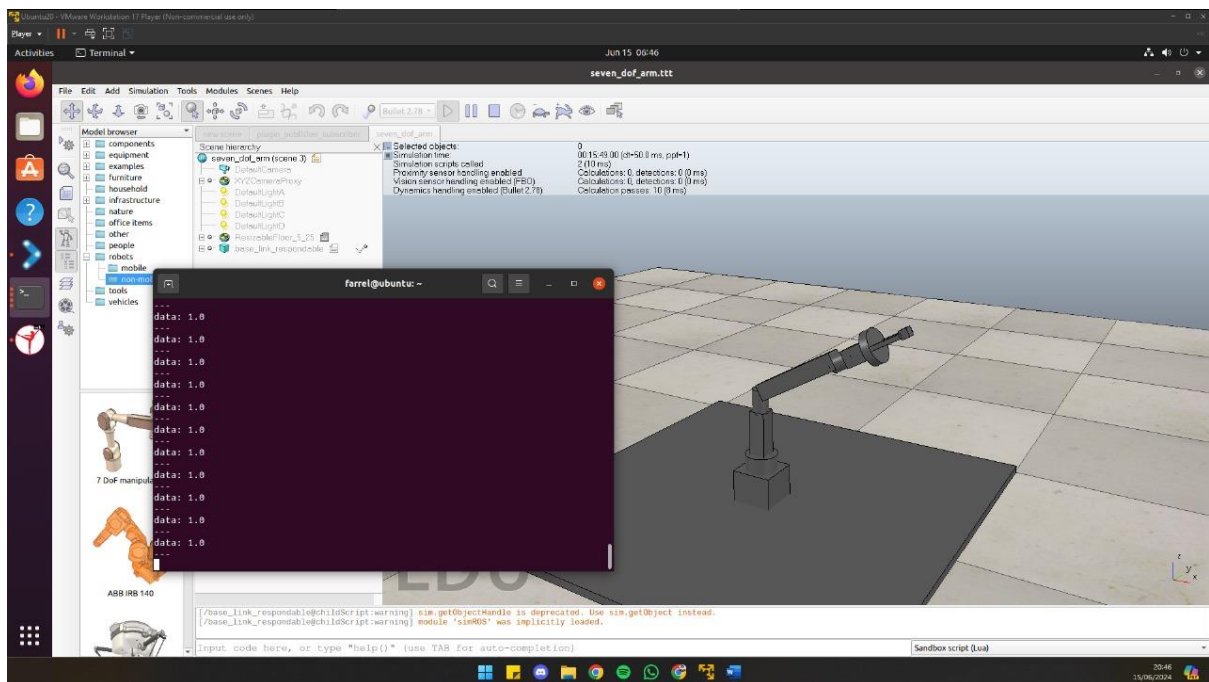
- Mengatur Coppeliasim dengan ROS, memahami plugin RosInterface, dan bekerja dengan pesan ROS dalam Coppeliasim.
- Mensimulasikan lengan robotik dan menambahkan antarmuka ROS ke pengontrol sendi Coppeliasim.
- Mengatur Webots dengan ROS dan mensimulasikan robot bergerak dengan Webots.
- Menulis pengontrol di Webots, mensimulasikan lengan robotik dengan Webots dan ROS, serta menulis node teleop menggunakan webots_ros.
- Mengintegrasikan ROS dengan simulasi Webots.

Perintah untuk mengatur Coppeliasim:

- **tar vxf:** Mengekstrak file arsip.
- **mv:** Mengubah nama atau memindahkan direktori.
- **echo "export COPPELIASIM_ROOT=..." >> ~/.bashrc:** Menambahkan variabel lingkungan ke file ~/.bashrc.
- **cd \$COPPELIASIM_ROOT:** Mengubah direktori kerja saat ini.
- **./coppeliasim.sh:** Menjalankan script Coppeliasim.

Bab ini memberikan panduan lengkap untuk mensimulasikan dan mengontrol robot menggunakan berbagai alat dan teknik dalam lingkungan simulasi yang berbeda, serta menekankan pentingnya memahami plugin dan antarmuka untuk menghubungkan simulasi dengan ROS.





Terjadi kendala saat simulasi webots, terjadi error pada aplikasi webots yang digunakan

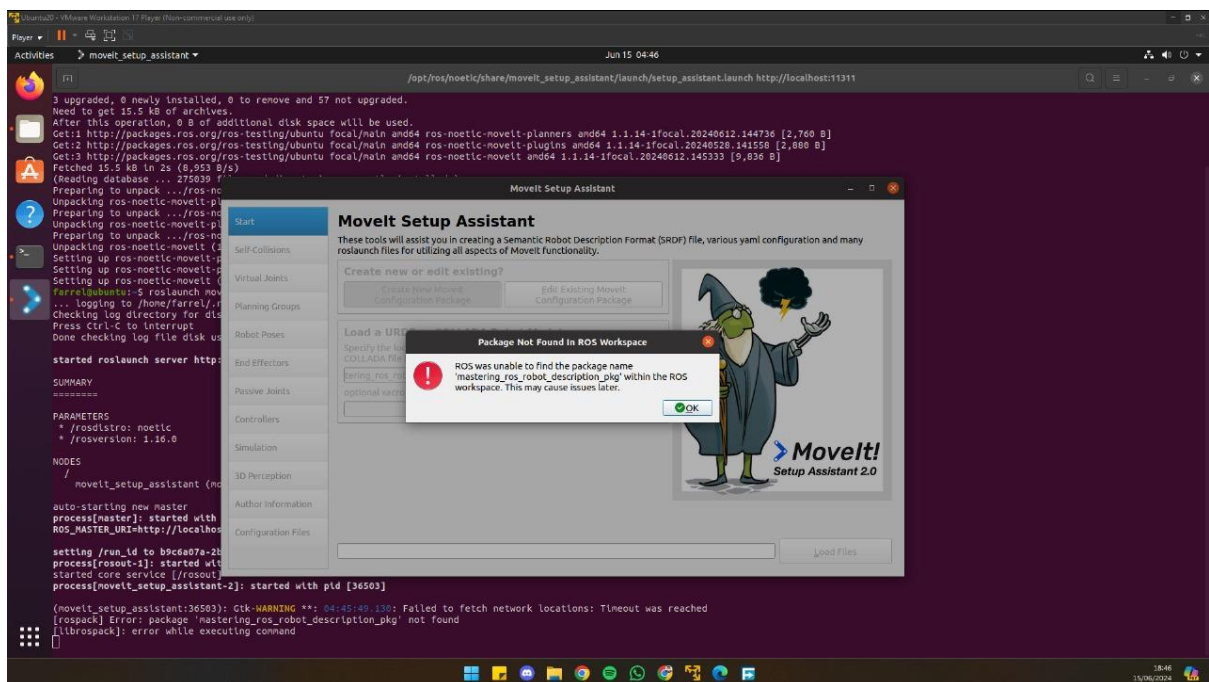
BAB 6

Pembahasan pada Bab ini: Penggunaan ROS MoveIt! dan Navigation Stack untuk Robot Manipulasi dan Navigasi Otonom

Bab ini membahas framework MoveIt! untuk perencanaan gerak robot, termasuk alat-alat seperti perencanaan lintasan dan penghindaran tabrakan. Arsitektur MoveIt!, khususnya node `move_group`, dijelaskan secara rinci. Bab ini juga menunjukkan cara menghasilkan paket konfigurasi MoveIt! menggunakan Setup Assistant tool untuk menambahkan sendi virtual, grup perencanaan, dan end effector robot.

Selain itu, bab ini membahas stack navigasi ROS yang memungkinkan robot bergerak secara otonom di lingkungan yang telah dipetakan. Ini mencakup pemetaan simultan dan lokalisasi (SLAM), serta perencanaan lintasan menggunakan paket seperti `amcl` dan peta statis. Bab ini menjelaskan persyaratan perangkat keras dan cara mengkonfigurasi paket navigasi seperti `gmapping` dan `move_base`.

Contoh praktis diberikan untuk menggunakan RViz dalam perencanaan gerak dan visualisasi, serta menghubungkan paket konfigurasi MoveIt! dengan Gazebo untuk simulasi. Pembaca akan belajar bagaimana menggabungkan MoveIt! dan stack navigasi ROS untuk memastikan robot dapat merencanakan dan mengeksekusi lintasan yang kompleks serta menavigasi secara otonom di lingkungan dinamis.



hambatan saat simulasi : File xacro tidak dapat dijalankan meskipun telah menggunakan file lain dan mengulanginya dari awal.