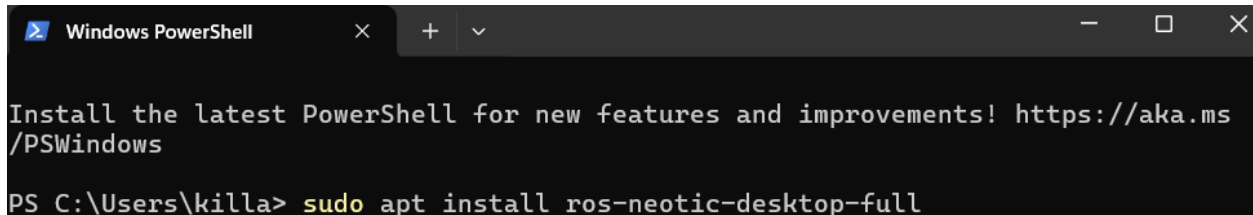


NAMA : FAKHRI ARASYID

NIM :1103190057

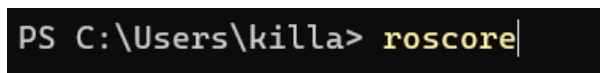
UAS

Chapter 1

A screenshot of a Windows PowerShell terminal window. The title bar shows 'Windows PowerShell' with standard window controls. The terminal content includes a message: 'Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows' and a command prompt: 'PS C:\Users\killa> sudo apt install ros-neotic-desktop-full'.

```
Windows PowerShell
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\killa> sudo apt install ros-neotic-desktop-full
```

Jadi, ros-neotic-desktop-full merujuk pada paket instalasi ROS yang mencakup semua pustaka, alat, dan paket-paket yang umumnya diperlukan untuk pengembangan aplikasi robotik secara komprehensif. Dengan menggunakan perintah `sudo apt install`, sistem akan mengunduh dan menginstal semua paket yang diperlukan untuk versi ROS Neotic desktop-full secara otomatis.

A screenshot of a Windows PowerShell terminal window showing the command to start the ROS core. The prompt is 'PS C:\Users\killa> roscore' followed by a cursor.

```
PS C:\Users\killa> roscore|
```

Dengan menjalankan roscore, Anda menginisialisasi infrastruktur yang diperlukan untuk membangun dan menjalankan aplikasi robotik menggunakan ROS. Ini adalah langkah pertama yang biasanya dilakukan sebelum memulai node-node lain seperti pengendali robot, sensor, atau elemen perangkat lunak lainnya yang terintegrasi dalam sistem robotik yang menggunakan ROS.

```
PS C:\Users\killa> source /opt/ros/noetic/setup.bash
```

/opt/ros/noetic/setup.bash: Ini adalah lokasi di mana skrip setup.bash untuk instalasi ROS Noetic berada. Skrip ini berisi serangkaian perintah untuk mengatur berbagai variabel lingkungan (seperti PATH, PYTHONPATH, ROS_PACKAGE_PATH, dan lain-lain) yang diperlukan agar Anda dapat menggunakan perintah dan alat yang terkait dengan ROS Noetic.

Chapter 2

```
PS C:\Users\killa> ~$ catkin_make
```

Perintah ini akan memulai proses kompilasi untuk semua paket ROS yang ada di dalam catkin_ws. Setelah selesai, Anda dapat menggunakan node-node dan library yang telah Anda bangun untuk menjalankan aplikasi robotik atau percobaan lainnya di lingkungan ROS.

Jadi, catkin_make adalah perintah penting dalam pengembangan aplikasi dan pembangunan paket-paket di dalam lingkungan ROS.

```
PS C:\Users\killa> ~$ rosrun mastering_ros_demo_pkg demo_service_client
```

Perintah ini akan menjalankan node demo_service_client yang ada dalam paket mastering_ros_demo_pkg. Node ini mungkin akan melakukan koneksi ke layanan ROS lainnya (seperti server layanan) untuk melakukan operasi tertentu, misalnya meminta data atau mengontrol perangkat keras.

Dengan menggunakan rosrun, Anda dapat dengan mudah menjalankan node-node ROS tanpa harus menavigasi langsung ke direktori tempat executable tersebut berada, karena rosrun akan mencari dan menjalankan node berdasarkan nama paket dan nama executable yang Anda berikan.

```
PS C:\Users\killa> ~$ rosrun mastering_ros_demo_pkg demo_topic_publishers
```

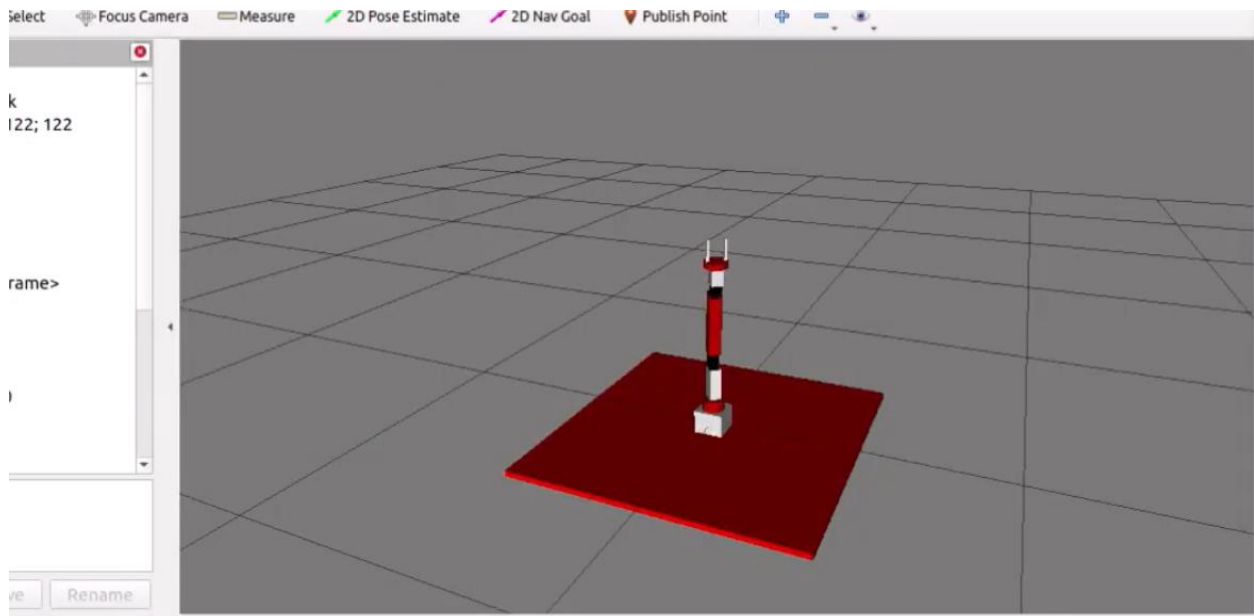
rosrun: Seperti yang telah dijelaskan sebelumnya, rosrun adalah perintah dalam ROS yang digunakan untuk menjalankan sebuah node atau executable ROS tanpa harus mengetahui lokasi file executable secara langsung.

mastering_ros_demo_pkg: Ini adalah nama paket ROS tempat node demo_topic_publishers berada. Paket ROS adalah unit dasar di dalam ROS yang mengorganisir kode, konfigurasi, dan sumber daya lainnya.

demo_topic_publishers: Ini adalah nama node atau executable yang akan dijalankan oleh rosrun. Dalam konteks ini, node tersebut mungkin bertanggung jawab untuk mempublikasikan (publish) data ke topik-topik (topics) yang ada di dalam sistem ROS.

Chapter 3

ARM



<?xml version="1.0" ?>

```
<launch>

  <arg name="model" />

  <!-- Parsing xacro and setting robot_description parameter -->

  <param name="robot_description" command="$(find xacro)/xacro $(find
mastering_ros_robot_description_pkg)/urdf/seven_dof_arm.xacro" />

  <node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher" />

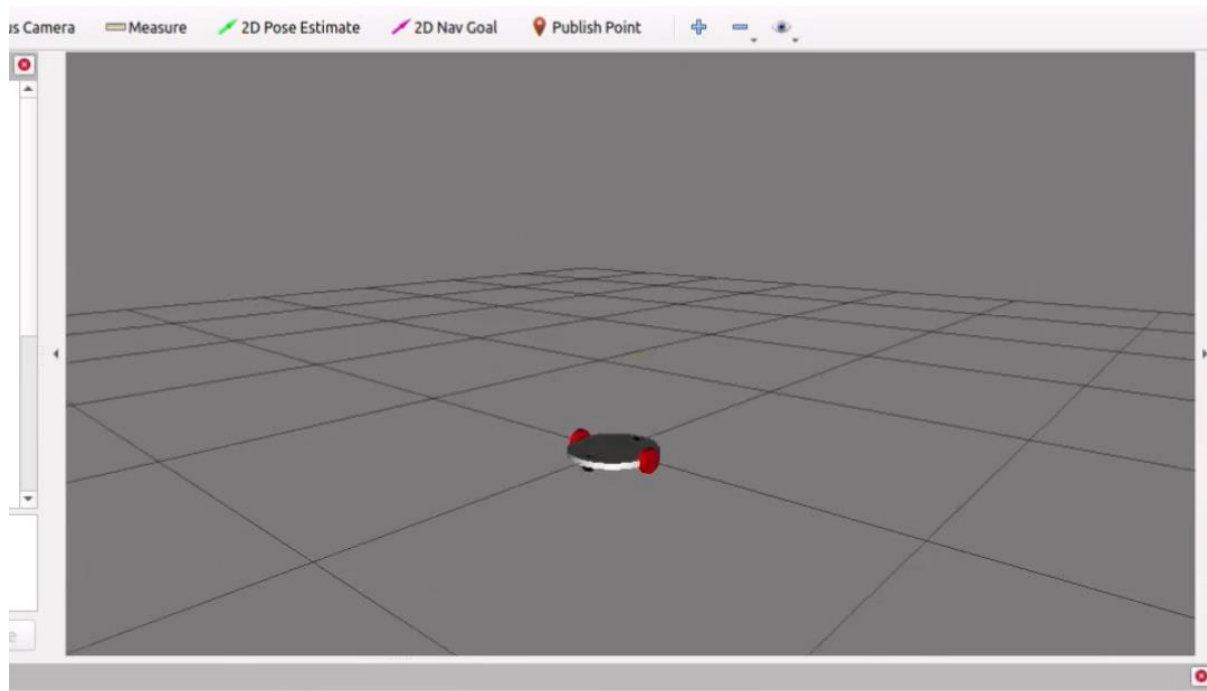
  <node name="joint_state_publisher_gui" pkg="joint_state_publisher_gui"
type="joint_state_publisher_gui" />

  <!-- Launch visualization in rviz -->

  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
mastering_ros_robot_description_pkg)/urdf.rviz" required="true" />

</launch>
```

ROBOT



```
<?xml version="1.0" ?>
```

```
<launch>
```

```
  <arg name="model" />
```

```
  <!-- Parsing xacro and setting robot_description parameter -->
```

```
  <param name="robot_description" command="$(find xacro)/xacro $(find  
mastering_ros_robot_description_pkg)/urdf/diff_wheeled_robot.xacro" />
```

```
  <!-- Setting gui parameter to true for display joint slider -->
```

```
  <!-- Starting Joint state publisher node which will publish the joint values -->
```

```
  <node name="joint_state_publisher_gui" pkg="joint_state_publisher_gui"  
type="joint_state_publisher_gui" />
```

```
  <!-- Starting robot state publish which will publish tf -->
```

```
<node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher" />
```

```
<!-- Launch visualization in rviz -->
```

```
<node name="rviz" pkg="rviz" type="rviz" args="-d $(find
mastering_ros_robot_description_pkg)/urdf.rviz" required="true" />
```

```
</launch>
```