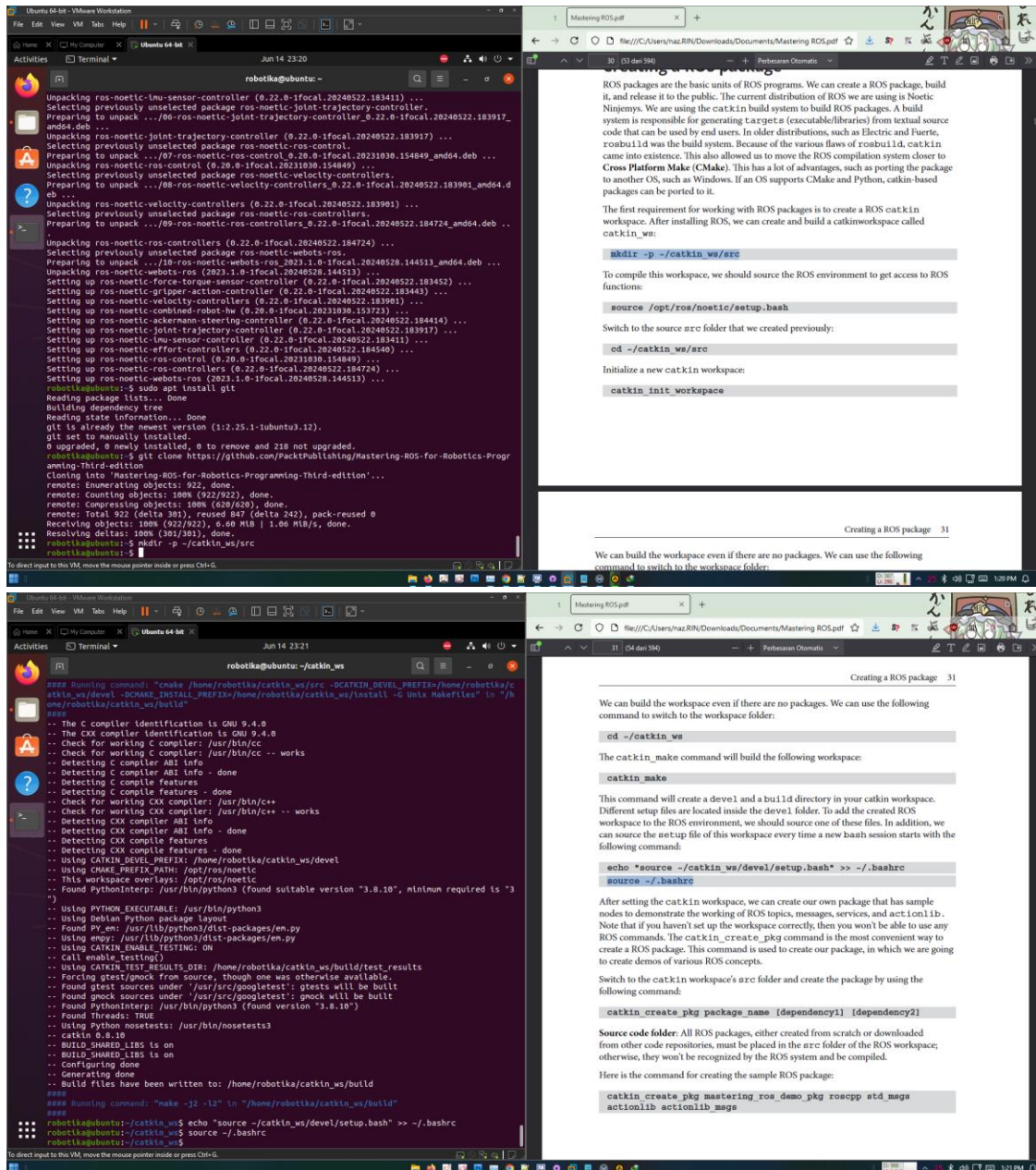


## Bab 1 & 2

Bab 1 dan 2 mempelajari tentang apa itu ROS serta mempelajari bagaimana cara instalasi, konfigurasi, memahami server dan client ROS, memahami publisher dan subscriber dari ROS



Ubuntu 64-bit - VMware Workstation

Activities Terminal Jun 14 23:43

Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage: 100%  
started roslaunch server http://ubuntu:11311/  
ros\_comm version 1.15.6

SUMMARY  
PARAMETERS  
\* /roslaunch: 1.15.6  
\* /rosversion: 1.16.0

NODES  
auto-starting new master  
process[master]: started with pid [36148]  
ROS\_MASTER\_URI=http://ubuntu:11311/  
setting /run\_id to http://ubuntu:11311/  
process[roscout-1]: started core serv

robotika@ubuntu: ~\$ catkin\_make

INFO [1718433795.624911478]: 473  
INFO [1718433795.724977026]: 474  
INFO [1718433795.824984128]: 475  
INFO [1718433795.925011866]: 476  
INFO [1718433796.024929366]: 477  
INFO [1718433796.124987226]: 478  
INFO [1718433796.224997699]: 479  
INFO [1718433796.325004996]: 480  
INFO [1718433796.424982496]: 481  
INFO [1718433796.524980322]: 482  
INFO [1718433796.625006513]: 483  
INFO [1718433796.725111745]: 484  
INFO [1718433796.824982212]: 485  
INFO [1718433796.925094081]: 486  
INFO [1718433797.024993503]: 487  
INFO [1718433797.125101919]: 488  
INFO [1718433797.224973203]: 489  
INFO [1718433797.325087967]: 490  
INFO [1718433797.424984444]: 491

robotika@ubuntu: ~\$

catkin\_ws generated

To direct input to this VM, move the mouse pointer inside or press Ctrl-G.

catkin\_ws

38 Getting Started with ROS Programming

The following diagram shows how the nodes communicate with each other. We can see that the `demo_topic_publisher` node publishes the `/demo_topic` and then subscribes to the `demo_topic_subscriber` node:

```
graph LR
    A[demo_topic_publisher] --> B[demo_topic_subscriber]
```

Figure 2.3 - Graph showing the communication between the publisher and subscriber nodes

We can use the `rostopic` and `rostopic` tools to debug and understand the working of the two nodes:

Ubuntu 64-bit - VMware Workstation

Activities Terminal Jun 14 23:57

Press Ctrl-C to interrupt  
Done checking log file disk usage. Usage: 100%  
started roslaunch server http://ubuntu:11311/  
ros\_comm version 1.15.6

SUMMARY  
PARAMETERS  
\* /roslaunch: 1.15.6  
\* /rosversion: 1.16.0

NODES  
auto-starting new master  
process[master]: started with pid [36148]  
ROS\_MASTER\_URI=http://ubuntu:11311/  
setting /run\_id to http://ubuntu:11311/  
process[roscout-1]: started core serv

robotika@ubuntu: ~\$ catkin\_make

INFO [1718434649.412434918]: anjay nabor  
INFO [1718434649.512325330]: 204  
INFO [1718434649.612417732]: anjay nabor  
INFO [1718434649.712334683]: 205  
INFO [1718434649.812409961]: anjay nabor  
INFO [1718434649.912338699]: 206  
INFO [1718434649.112401216]: anjay nabor  
INFO [1718434649.212413250]: 207  
INFO [1718434649.312448912]: anjay nabor  
INFO [1718434649.412462085]: anjay nabor  
INFO [1718434650.012355083]: 209  
INFO [1718434650.112454892]: anjay nabor  
INFO [1718434650.212357910]: 210  
INFO [1718434650.312428059]: anjay nabor  
INFO [1718434650.412376141]: 211  
INFO [1718434650.512448792]: anjay nabor  
INFO [1718434650.612363443]: 212  
INFO [1718434650.712443425]: anjay nabor  
INFO [1718434650.812361562]: 213  
INFO [1718434650.912435515]: anjay nabor  
INFO [1718434650.112358985]: 214  
INFO [1718434650.212432861]: anjay nabor

robotika@ubuntu: ~\$

catkin\_ws generated

To direct input to this VM, move the mouse pointer inside or press Ctrl-G.

catkin\_ws

40 Getting Started with ROS Programming

We can test the message by adding the following lines of code to `demo_msgs.msg` file:

```
add_message_definition(demo_msgs.msg_publisher)
add_message_definition(demo_msgs.msg_subscriber)
```

add\_dependencies(demo\_msgs\_publisher mastering\_ros\_demo\_pkg\_generate\_messages\_cpp)

add\_dependencies(demo\_msgs\_subscriber mastering\_ros\_demo\_pkg\_generate\_messages\_cpp)

target\_link\_libraries(demo\_msgs\_publisher \${catkin\_LIBRARIES})

target\_link\_libraries(demo\_msgs\_subscriber \${catkin\_LIBRARIES})

An important difference between this edited `Makefile.txt` file and the older one is the dependency specification to the messages generated in `mastering_ros_demo_pkg`. This dependency is specified with the `add_dependencies` instruction. Note that if you forget to include this instruction, the ROS system will start compiling the C++ source code before the message is generated. In this way, a compilation error will be generated since the header files of the custom messages could not be found. Now, we are ready to build the package.

Let's build the package using `catkin_make` and test the node by following these steps:

1. Run roscore:
2. Start the custom message publisher node:
3. Start the custom message subscriber node:

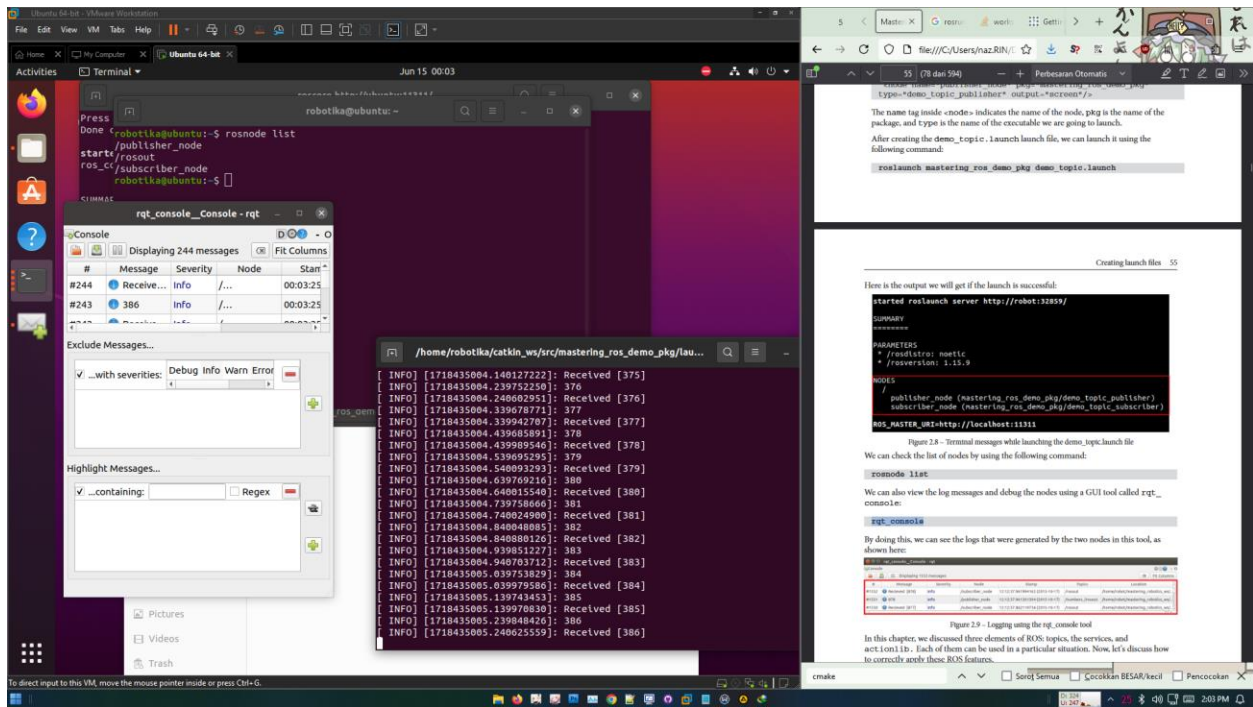
The publisher node publishes a string along with an integer, while the subscriber node subscribes to the topic and prints its values. The output and graph are as follows:

```
graph LR
    A[demo_msgs_publisher] --> B[demo_msgs_subscriber]
```

Figure 2.4 - Running the publisher and subscriber using custom message definition







## Bab 3

Bab 3 mempelajari cara modifikasi file URDF dan juga mencoba Rviz untuk menampilkan demo robot

The image is a composite of four screenshots illustrating the process of visualizing a 3D robot model in RViz.

**Top-Left Screenshot:** Shows the RViz interface. The main window displays a 3D model of a robot arm with a blue base link, a red tilt link, and a green pan link. The left sidebar shows the 'Displays' panel with 'RobotModel' selected. The 'Global Options' panel shows settings for the fixed frame, background color, and status. The bottom status bar shows ROS Time, ROS Elapsed, and Wall Time.

**Top-Right Screenshot:** Shows a PDF document titled 'Mastering ROS.pdf'. The text discusses visualizing the 3D robot model in RViz. It mentions that after designing the URDF, the user can view it in RViz by creating a view demo launch launch file and putting the following code into the launch file. The code snippet shows the launch file structure for visualizing the robot model.

**Bottom-Left Screenshot:** Shows a terminal window and a file explorer. The terminal window displays the output of the 'roscd' command, showing the current directory is the 'mastering\_ros\_robot' package. The file explorer shows the 'urdf' directory containing files like 'base\_link.urdf', 'pan\_tilt.urdf', and 'tilt.urdf'. A small window shows the 'pan\_tilt.urdf' file, which defines the 'pan' joint and 'tilt' joint.

**Bottom-Right Screenshot:** Shows a PDF document titled 'Mastering ROS.pdf'. The text discusses working with ROS for 3D modeling. It mentions that in the previous version of ROS, the GUI of joint\_state\_publisher was enabled thanks to a ROS parameter called 'gui'. To start the GUI in the launch file, this parameter had to be set to true before starting the joint\_state\_publisher node. The text also mentions that in the current version of ROS, launch files should be updated to launch joint\_state\_publisher\_gui instead of joint\_state\_publisher.

Ubuntu 64-bit - VMware Workstation

File Edit View VM Help

Jun 15 00:40

urdfviz - RViz

File Panels Help

Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point

Displays

Global Options

Fixed Frame base\_link

Background Color 122; 122; 122

Frame Rate 30

Default Light

Global Status: OK

Fixed Frame OK

Grid

Status: OK

Reference Frame <Fixed Frame>

Plane Cell Count 10

Normal Cell Count 0

Cell Size 1

Line Style Lines

Color 0; 0; 0

Alpha 0.5

Plane XY

Offset 0; 0; 0

Add Duplicate Remove Rename

Time

Pause Synchronization: Off ROS Time: 1718437210.72 ROS Elapsed: 33.42 Wall Time: 1718437210.76 Wall Elapsed: 33.39

Reset 31 fps

```
[WARN] [1718437176.551215193]: link 'front_left_wheel' material 'DarkGray' undefined.
[WARN] [1718437176.551363393]: The root link base_footprint has an inertia specified in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an extra dummy link to your URDF.
[WARN] [1718437177.110222924]: link 'front_right_wheel' material 'DarkGray' undefined.
[WARN] [1718437177.110253417]: link 'front_right_wheel' material 'DarkGray' undefined.
[WARN] [1718437177.110281964]: link 'front_left_wheel' material 'DarkGray' undefined.
[WARN] [1718437177.110289142]: link 'front_left_wheel' material 'DarkGray' undefined.
```

To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

Mastering ROS.pdf

1 (11 dei 394)

Perfexaram Otomatik

mobile robot with differential wheeled mechanisms.

### Creating a robot model for the differential drive mobile robot

A differential wheeled robot will have two wheels connected to opposite sides of the robot chassis, which is supported by one or two caster wheels. The wheels will control the speed of the robot by regulating the velocity of the single wheels. If the two motors are running at the same speed, the wheels will move forward or backward. If one wheel is running slower than the other, the robot will turn to the side of the lower speed. If we want to turn the robot to the left side, we reduce the velocity of the left wheel and vice versa.

There are two supporting wheels, called **caster wheels**, that will support the robot and rotate freely based on the movement of the main wheels.

90 Working with ROS for 3D Modeling

The URDF model of this robot is present in the cloned ROS package. The final robot model is shown here:

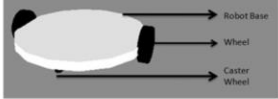



Figure 3.12: The differential drive mobile robot

The preceding robot has five joints and links. The two main joints connect the wheels to the robot, while the others are fixed joints connecting the caster wheels and the base footprint to the body of the robot.

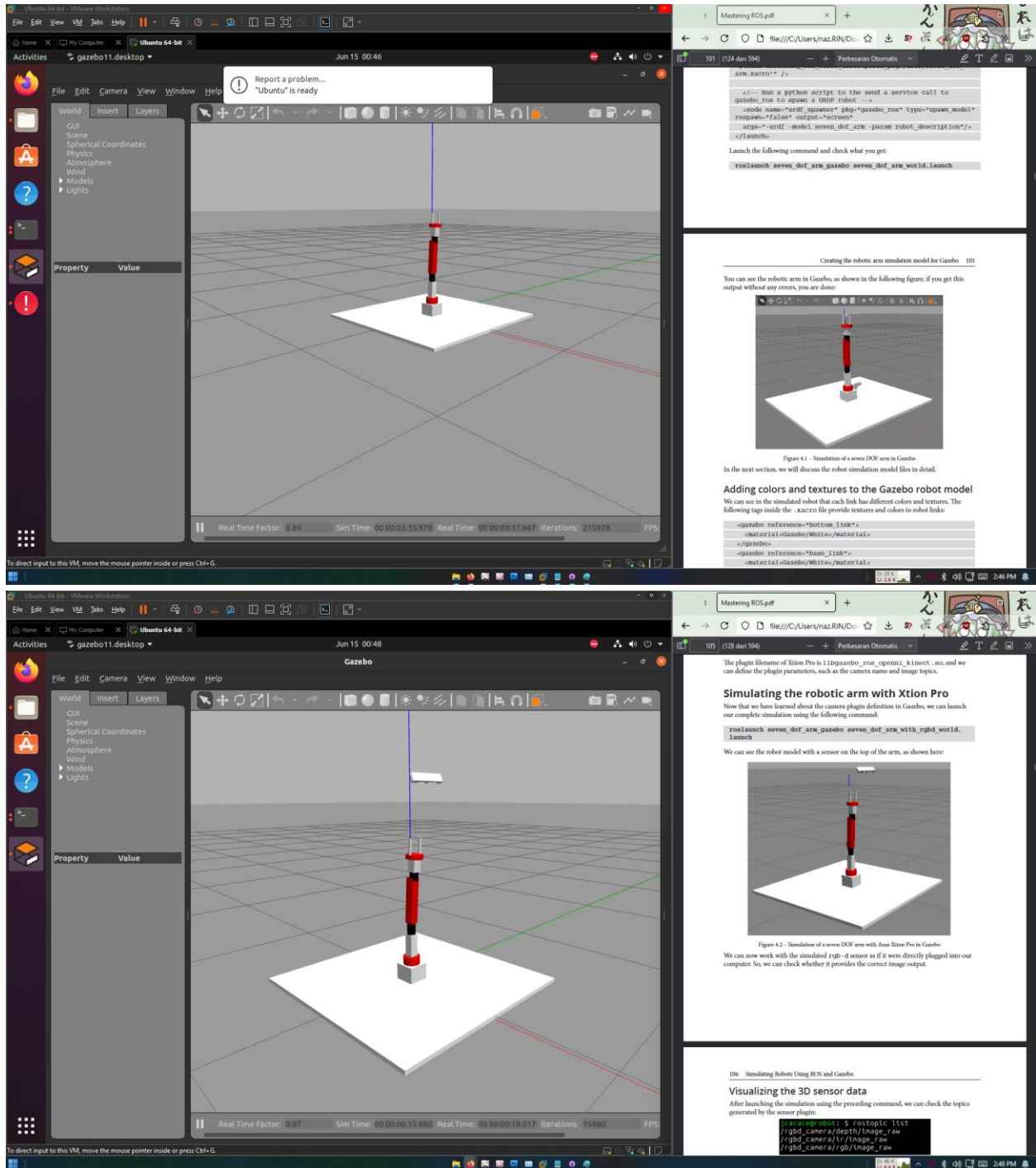
The preceding robot has five joints and five links. The two main joints connect the wheels with the base of the robot. The other joints are fixed and are used to link the caster wheels and the base footprint of the robot with its base link, respectively. Here is the connection graph of this robot.





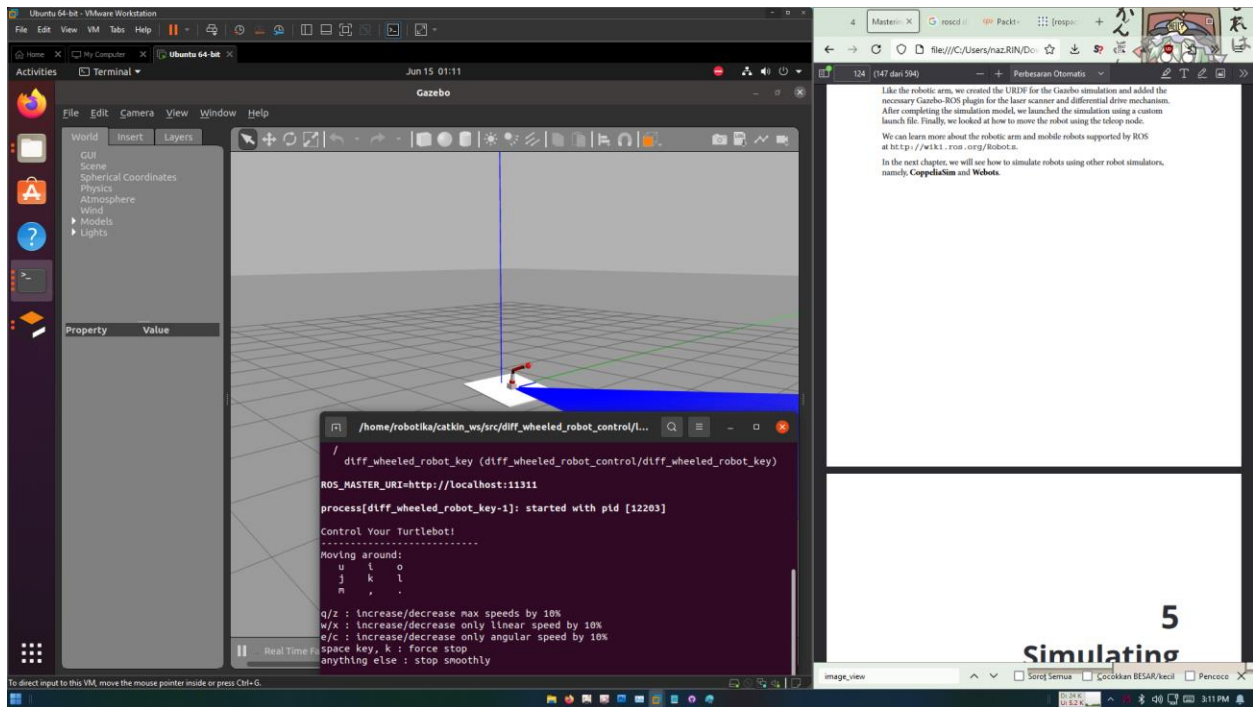
## Bab 4

Bab 4 mempelajari tentang menjalankan demo robot menggunakan Gazebo dan juga mengontrolnya dengan menggunakan teleop dan menampilkan gambar sensor menggunakan inmaghe\_view









## Bab 5

### Bab 5 Mencoba menjalankan demo robot menggunakan CoppeliaSim dan juga mengontrolnya menggunakan ROS

The collage consists of four screenshots illustrating the integration of ROS with CoppeliaSim for robot simulation and control.

**Top-Left Screenshot:** Shows the CoppeliaSim interface with a scene named 'plugin\_publisher\_subscriber'. The 'Model browser' on the left lists various components like 'components', 'equipment', 'examples', 'Furniture', 'Household', 'Infrastructure', 'nature', 'office items', 'other', 'people', 'robots', 'mobile', 'tools', and 'vehicles'. The 'Scene hierarchy' on the right shows the loaded scene. The 'Image View' window displays a camera feed from the active camera. The console output shows the execution of the 'plugin\_publisher\_subscriber' script, including warnings about deprecated methods and the successful execution of the 'sim.call\_type' function.

**Top-Right Screenshot:** Shows a ROS node interface for image streaming. The 'Image View' window displays a camera feed from the active camera. The console output shows the execution of the 'plugin\_publisher\_subscriber' script, including warnings about deprecated methods and the successful execution of the 'sim.call\_type' function.

**Bottom-Left Screenshot:** Shows a ROS terminal window with error messages and a Lua script for controlling a 7-DOF arm. The terminal output shows the execution of the 'seven\_dof\_arm.lua' script, including warnings about deprecated methods and the successful execution of the 'sim.call\_type' function. The script is designed to control the arm's position and orientation.

**Bottom-Right Screenshot:** Shows a ROS node interface for controlling a 7-DOF arm. The 'Image View' window displays a camera feed from the active camera. The console output shows the execution of the 'seven\_dof\_arm.lua' script, including warnings about deprecated methods and the successful execution of the 'sim.call\_type' function.

The collage consists of four screenshots from a ROS2 tutorial:

- Top-left:** A screenshot of the **MoveIt Setup Assistant** window. The 'Define Planning Groups' tab is active, showing a list of joints for a 7-DOF arm. The 'Create New Planning Group' dialog is open, with 'Kinematics' set to 'None' and 'Group Default Planner' set to 'None'. The 'Add Components to Group' section shows 'Recommended' components: 'Add Kin. Chain', 'Add Joints', 'Add Subgroups', and 'Add Links'. The 'Advanced Options' section is also visible.
- Top-right:** A screenshot of the 'Create New Planning Group' dialog box. It shows the 'Kinematics' dropdown set to 'None' and the 'Group Default Planner' dropdown set to 'None'. The 'Add Components to Group' section is expanded, showing 'Recommended' components: 'Add Kin. Chain', 'Add Joints', 'Add Subgroups', and 'Add Links'. The 'Advanced Options' section is also visible.
- Bottom-left:** A screenshot of a terminal window. It shows the execution of the command `roslaunch seven_dof_arm_gazebo dof_arm_bringup_moveit.l` and the subsequent navigation to the `seven_dof_arm_world` directory. The terminal output shows the launch process and the location of the `seven_dof_arm_world` directory.
- Bottom-right:** A screenshot of a presentation slide titled 'Motion planning of a robot in RViz using the MoveIt! configuration package'. The slide explains that the MoveIt! plugin provides an RViz interface for motion planning. It also shows a screenshot of the RViz interface with the MoveIt! plugin loaded, displaying a 3D model of the robot arm in a simulated environment.