Nama : Satya Athaya Daniswara

NIM : 1103213152

CHAPTER 1

Setup your sources.list

Setup your computer to accept software from packages.ros.org.

- sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

Set up your keys

- sudo apt install curl # if you haven't already installed curl
- curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add -

Installation

First, make sure your Debian package index is up-to-date:

- sudo apt update

Now pick how much of ROS you would like to install.

- **Desktop-Full Install: (Recommended)** : Everything in **Desktop** plus 2D/3D simulators and 2D/3D perception packages
    - sudo apt install ros-noetic-desktop-full

or click here

- **Desktop Install:** Everything in **ROS-Base** plus tools like rqt and rviz
    - sudo apt install ros-noetic-desktop

or click here

- **ROS-Base: (Bare Bones)** ROS packaging, build, and communication libraries. No GUI tools.
    - sudo apt install ros-noetic-ros-base

or click here

There are even more packages available in ROS. You can always install a specific package directly.

- sudo apt install ros-noetic-PACKAGE

e.g.

sudo apt install ros-noetic-slam-gmapping

To find available packages, see ROS Index or use:

apt search ros-noetic

Environment setup

You must source this script in every **bash** terminal you use ROS in.

source /opt/ros/noetic/setup.bash

It can be convenient to automatically source this script every time a new shell is launched. These commands will do that for you.

Bash

If you have more than one ROS distribution installed, ~/.bashrc must only source the setup.bash for the version you are currently using.

echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc

source ~/.bashrc

zsh

echo "source /opt/ros/noetic/setup.zsh" >> ~/.zshrc

source ~/.zshrc

Dependencies for building packages

Up to now you have installed what you need to run the core ROS packages. To create and manage your own ROS workspaces, there are various tools and requirements that are distributed separately. For example, rosinstall is a frequently used command-line tool that enables you to easily download many source trees for ROS packages with one command.

To install this tool and other dependencies for building ROS packages, run:

sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential

Initialize rosdep

Before you can use many ROS tools, you will need to initialize rosdep. rosdep enables you to easily install system dependencies for source you want to compile and is required to run some core components in ROS. If you have not yet installed rosdep, do so as follows.

sudo apt install python3-rosdep

With the following, you can initialize rosdep.

sudo rosdep init

rosdep update

CHAPTER 2

1. Jalankan perintah “**roscore**” terlebih dahulu
2. Membuat folder catkin dengan beberapa perintah berikut :

   mkdir -p ~/catkin_ws/src

   cd ~/catkin_ws/src

   catkin_init_workspace

   cd ..

catkin_make

source devel/setup.bash

3. Membuat folder “my_firtst_package” dengan beberapa perintah berikut :

   cd ~/catkin_ws/src

   catkin_create_pkg my_first_package std_msgs rospy roscpp

   cd ~/catkin_ws

   catkin_make

   source devel/setup.bash

4. Buat folder “scripts” lalu buat file “talker.py” dan “listener.py” di dalamnya, berikut perintahnya :

   nano scripts/talker.py

   lalu isi filenya :

```
#!/usr/bin/env python3
import rospy
from std_msgs.msg import String

def talker():
    pub = rospy.Publisher('chatter', String, queue_size=10)
    rospy.init_node('talker', anonymous=True)
    rate = rospy.Rate(1)  # 1 Hz
    while not rospy.is_shutdown():
        hello_str = "Hello ROS %s" % rospy.get_time()
        rospy.loginfo(hello_str)
        pub.publish(hello_str)
        rate.sleep()

if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
```

```
        pass
```

Simpan “ctrl+o” lalu “enter” lalu exit “ctrl+x”

Selanjutnya buat file

nano scripts/listener.py

lalu isi filenya :

```
#!/usr/bin/env python3
import rospy
from std_msgs.msg import String

def callback(data):
    rospy.loginfo("I heard: %s", data.data)

def listener():
    rospy.init_node('listener', anonymous=True)
    rospy.Subscriber("chatter", String, callback)
    rospy.spin()

if __name__ == '__main__':
    listener()
```

5. Pada terminal jalankan perintah berikut untuk izin eksekusi :

   chmod +x scripts/talker.py scripts/listener.py

6. Kembali lagi ke catkin :

   cd ~/catkin_ws

   catkin_make

   source devel/setup.bash

7. Pada terminal baru jalankan :

   source /opt/ros/noetic/setup.bash

source ~/catkin_ws/devel/setup.bash

rosrun my_first_package talker.py

8. Pada terminal baru jalankan :

source /opt/ros/noetic/setup.bash

source ~/catkin_ws/devel/setup.bash

rosrun my_first_package listener.py

Akan memunculkan dua output yang saling berhubungan

CHAPTER 3

Tutorial Menampilkan Model 3D Robot di ROS

Prasyarat

```bash
sudo apt-get install ros-noetic-urdf-tutorial
sudo apt-get install ros-noetic-joint-state-publisher-gui
```

Langkah 1: Membuat Package

```bash
mkdir -p ~/catkin_ws/src
cd ~/catkin_ws/src
catkin_create_pkg my_robot_description urdf rviz
cd my_robot_description
mkdir urdf
```

Langkah 2: Membuat File URDF

Buat file `robot.urdf` dalam folder `urdf`:

```
<?xml version="1.0"?>
<robot name="pan_tilt">
```

```
<link name="base_link">

  <visual>
    <geometry>
         <cylinder length="0.01" radius="0.2"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <material name="yellow">
      <color rgba="1 1 0 1"/>
    </material>
  </visual>

  <collision>
    <geometry>
         <cylinder length="0.03" radius="0.2"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0"/>
  </collision>
  <inertial>
         <mass value="1"/>
         <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
  </inertial>
</link>

<joint name="pan_joint" type="revolute">
  <parent link="base_link"/>
  <child link="pan_link"/>
  <origin xyz="0 0 0.1"/>
  <axis xyz="0 0 1" />
  <limit effort="300" velocity="0.1" lower="-3.14" upper="3.14"/>
  <dynamics damping="50" friction="1"/>
</joint>
```

```
<link name="pan_link">
  <visual>
    <geometry>
        <cylinder length="0.4" radius="0.04"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0.09"/>
    <material name="red">
      <color rgba="0 0 1 1"/>
    </material>
  </visual>
  <collision>
    <geometry>
        <cylinder length="0.4" radius="0.06"/>
    </geometry>
    <origin rpy="0 0 0" xyz="0 0 0.09"/>
  </collision>
  <inertial>
        <mass value="1"/>
        <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
  </inertial>
</link>

<joint name="tilt_joint" type="revolute">
  <parent link="pan_link"/>
  <child link="tilt_link"/>
  <origin xyz="0 0 0.2"/>
  <axis xyz="0 1 0" />
  <limit effort="300" velocity="0.1" lower="-4.64" upper="-1.5"/>
  <dynamics damping="50" friction="1"/>
</joint>
```

```
 <link name="tilt_link">
   <visual>
     <geometry>
          <cylinder length="0.4" radius="0.04"/>
     </geometry>
     <origin rpy="0 1.5 0" xyz="0 0 0"/>
     <material name="green">
       <color rgba="1 0 0 1"/>
     </material>
   </visual>
   <collision>
     <geometry>
          <cylinder length="0.4" radius="0.06"/>
     </geometry>
     <origin rpy="0 1.5 0" xyz="0 0 0"/>
   </collision>
   <inertial>
          <mass value="1"/>
          <inertia ixx="1.0" ixy="0.0" ixz="0.0" iyy="1.0" iyz="0.0" izz="1.0"/>
   </inertial>

 </link>



</robot>
```

Langkah 3: Membuat Launch File

Buat folder `launch` dan file `display.launch`:

```bash
mkdir launch
```

Isi `display.launch`:

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

<launch>

  <arg name="model" />

  <param name="robot_description" textfile="$(find
mastering_ros_robot_description_pkg)/urdf/pan_tilt.urdf" />

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

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

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

</launch>
```

Langkah 4: Build Workspace

```bash
cd ~/catkin_ws
catkin_make
source devel/setup.bash
```

Langkah 5: Menjalankan Visualisasi

```bash
roslaunch my_robot_description display.launch
```

Langkah 6: Konfigurasi RViz

1. Set Fixed Frame ke "base_link"
2. Add -> RobotModel
3. Global Status seharusnya berubah menjadi OK
4. Robot seharusnya terlihat di viewport

Troubleshooting

- Jika "Global Status: Error", periksa:
  - Fixed Frame sudah diset ke "base_link"
  - Package sudah di-build dengan `catkin_make`
  - Source workspace dengan `source devel/setup.bash`
  - Semua dependensi terinstall

CHAPTER 4

Simulating robot with ros and gazebo

Download link github :

https://github.com/PacktPublishing/Mastering-ROS-for-Robotics-Programming-Third-edition/tree/main/Chapter4/seven_dof_arm_gazebo

1. Moving the robots joint using ros controller in gazebo

roslaunch seven_dof_arm_gazebo seven_dof_arm_gazebo_control.launch

rostopic pub /seven_dof_arm/joint4_position_controller/command std_msgs/Float64 “data: 1.0”

2.adding the ros teleop mode

roslaunch diff_wheeled_robot_gazebo diff_wheeled_robot_gazebo_full.launch

roslaunch diff_wheeled_robot_control keyboard_teleop.launch

rviz

fixed frame -> odom ->create visualization by topic ->laserscan