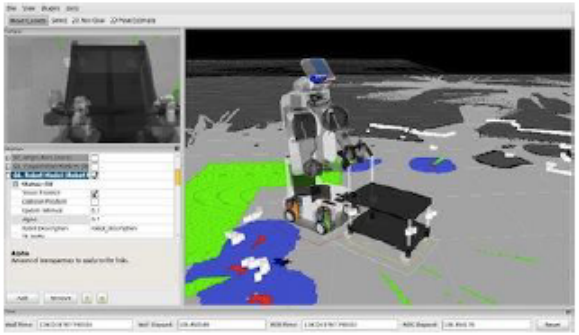
ECE 4703
Mobile Autonomous Robots

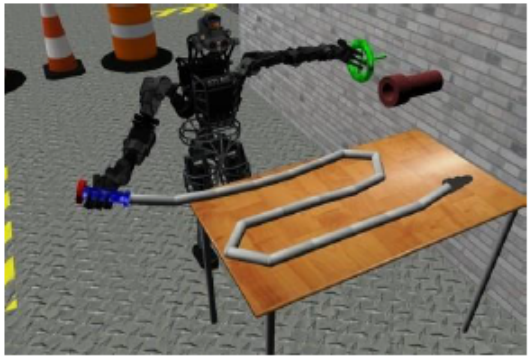
Jenny Zhen Yu
zhenyu@cpp.edu
Department of Electrical and Computer Engineering
California State Polytechnic University, Pomona

Various Development Tools for ROS

- Provides various development tools needed for robot development
- Improving the efficiency of robot development

- Command-Line Tools
  - Robot access only with commands provided by ROS without GUI & Use almost all ROS features
- RViz
  - Provides powerful 3D visualization tool
  - Visualizes sensor data such as laser, camera, etc.
  - Represents robot configuration and planned motion
- RQT
  - Provides Qt-based framework for developing graphic interface
  - Displays nodes and connection information between them (rqt_graph)
  - Floats encoder, voltage, or number that changes over time (rqt_plot)
  - Records data in message form and play back (rqt_bag)
- Gazebo
  - 3D simulator which supports physics engine, robot, sensor, environmental model, etc.
  - High compatibility with ROS





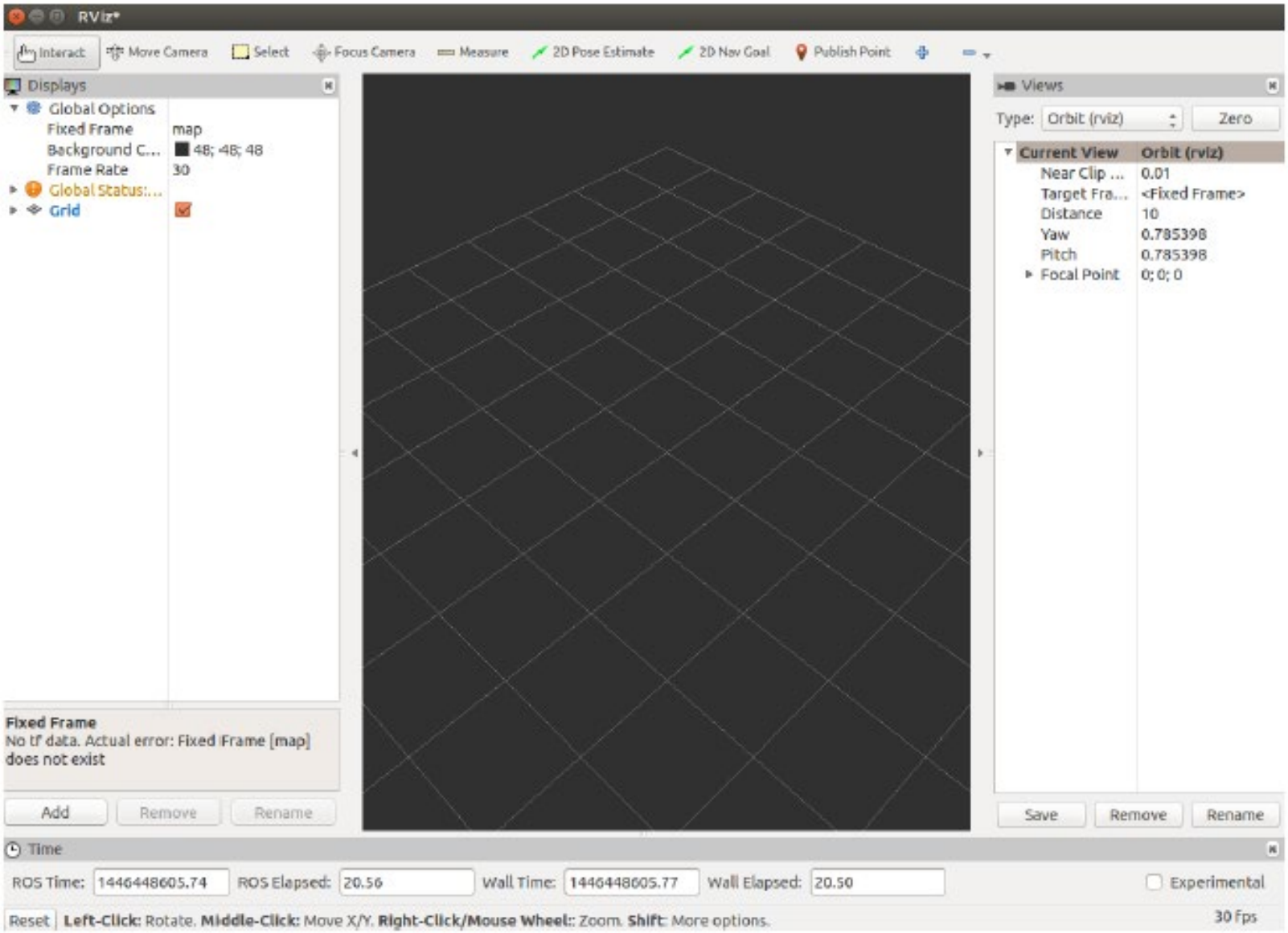
http://www.ros.org/core-components/

RViz

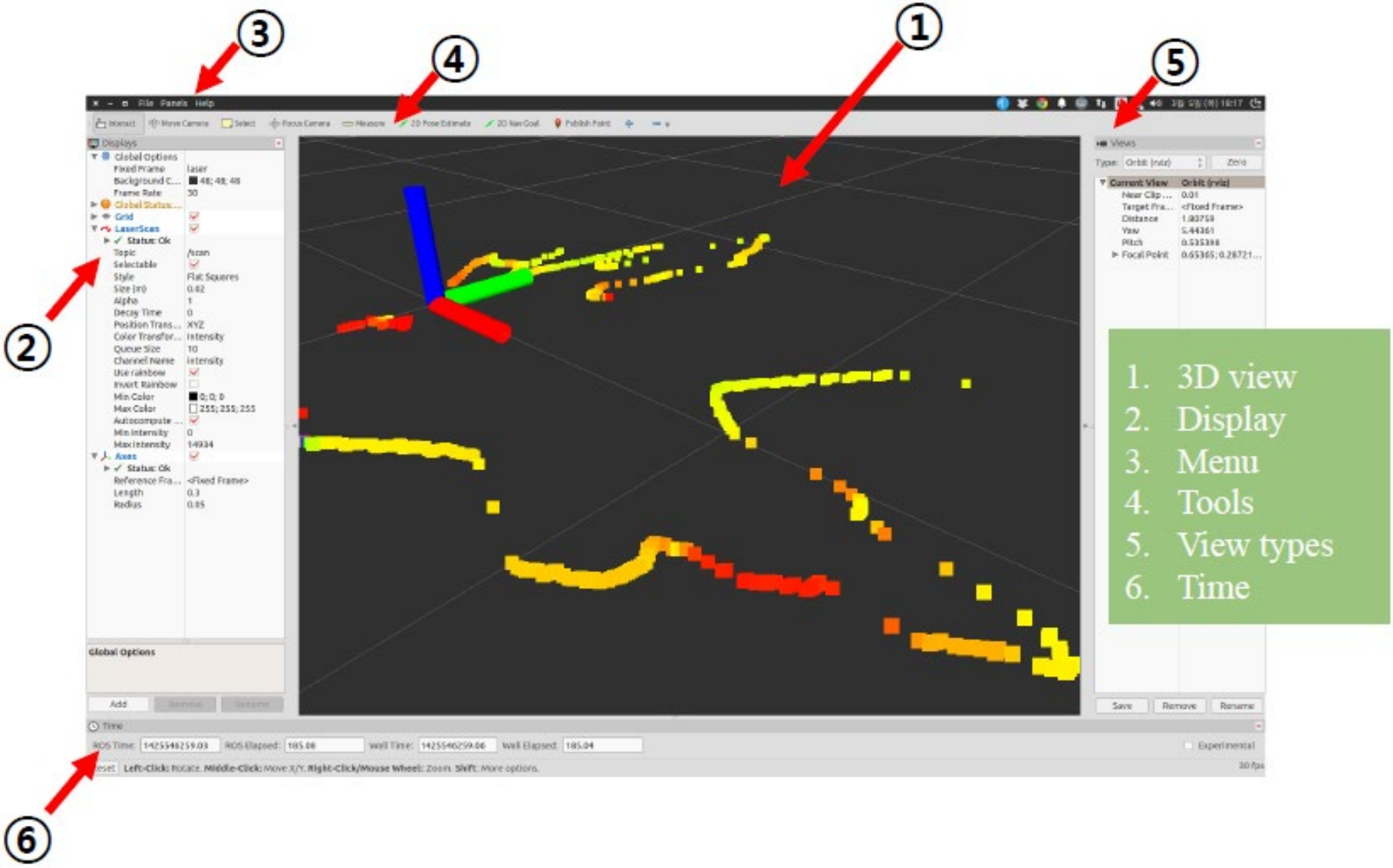
RViz(ROS Visualization Tool)

- 3D visualization tool for ROS
  - Visualization of sensor data
  - Distance data of LASER distance sensor(LDS)
  - Point cloud data from depth camera such as 'RealSense', 'Kinect', 'Xtion', etc.
  - Image data of camera
  - Inertia data of IMU sensor ...
- Represents robot configuration and planned motion
  - URDF (Unified Robot Description Format)
- Navigation
- Manipulation
- Tele-operation

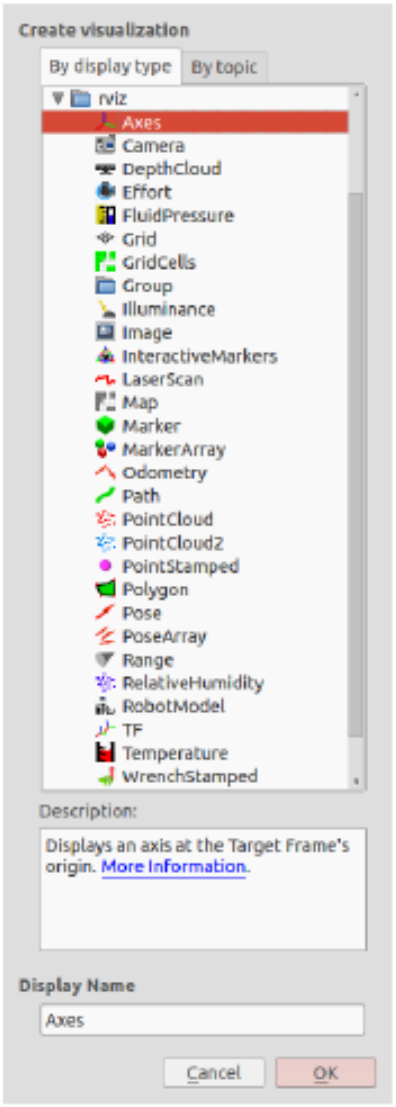
RViz initial screen (not configured yet)



Screen Configuration of RViz (for LDS)



Display type of RViz (Click ‘ADD’in Display Menu)

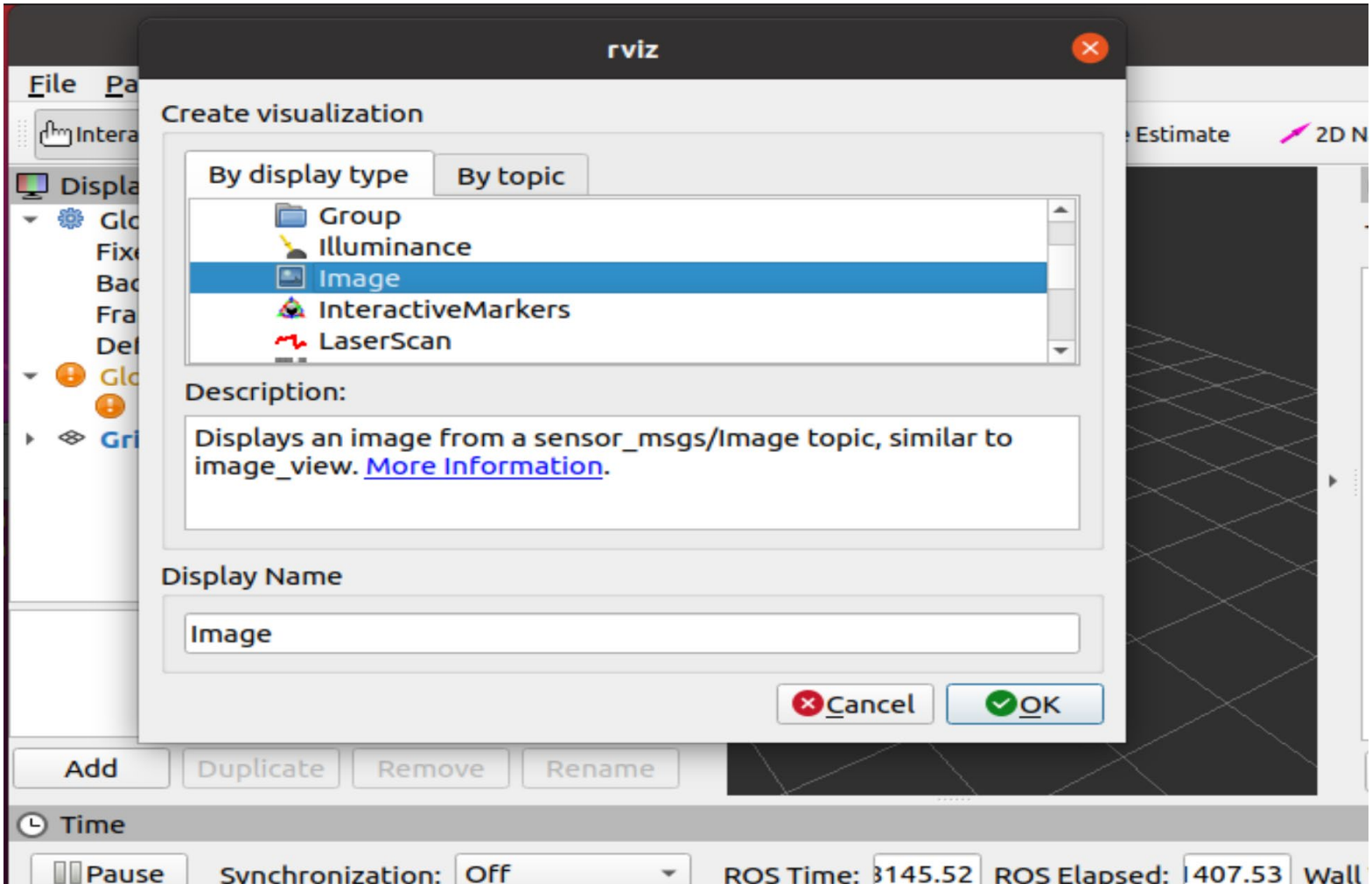


- Axes
- Camera
- Depth cloud
- Effort
- Fluid pressure
- Grid
- Grid cells (used for map)
- Group
- Illuminance
- Video
- Interactive marker
- Laser scan
- Map
- Marker
- Marker array
- Odometry
- Path
- Point cloud
- Point cloud2
- Point stamped
- Polygon
- Pose
- Pose array
- Range
- Temperature
- Robot model
- TF
- Relative Humidity
- WrenchStamped

RViz

```
ab@ab-c:~$ roscore
```

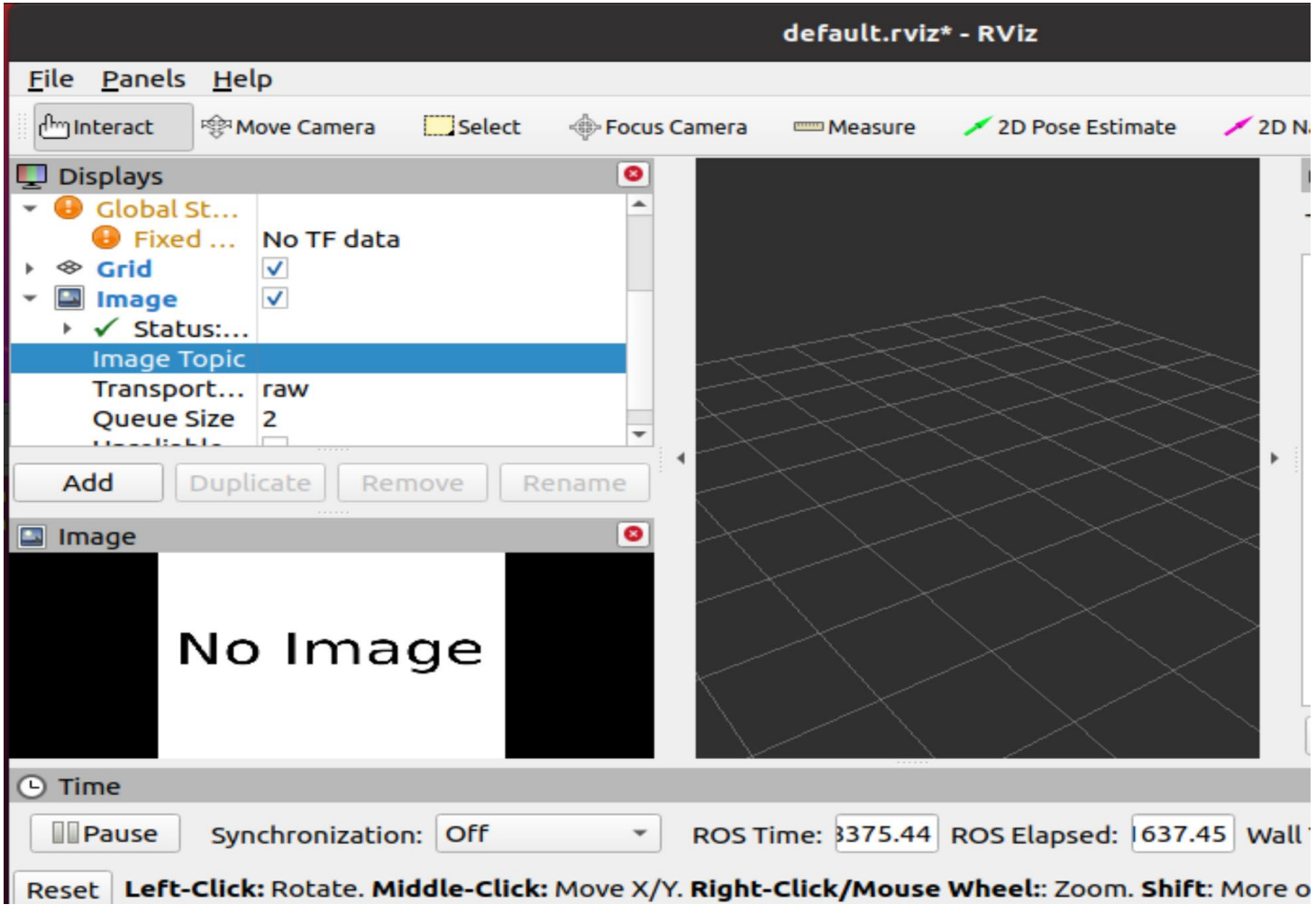
```
ab@ab-c:~$ rosrun rviz rviz
```



RViz

```
ab@ab-c:~$ roscore
```

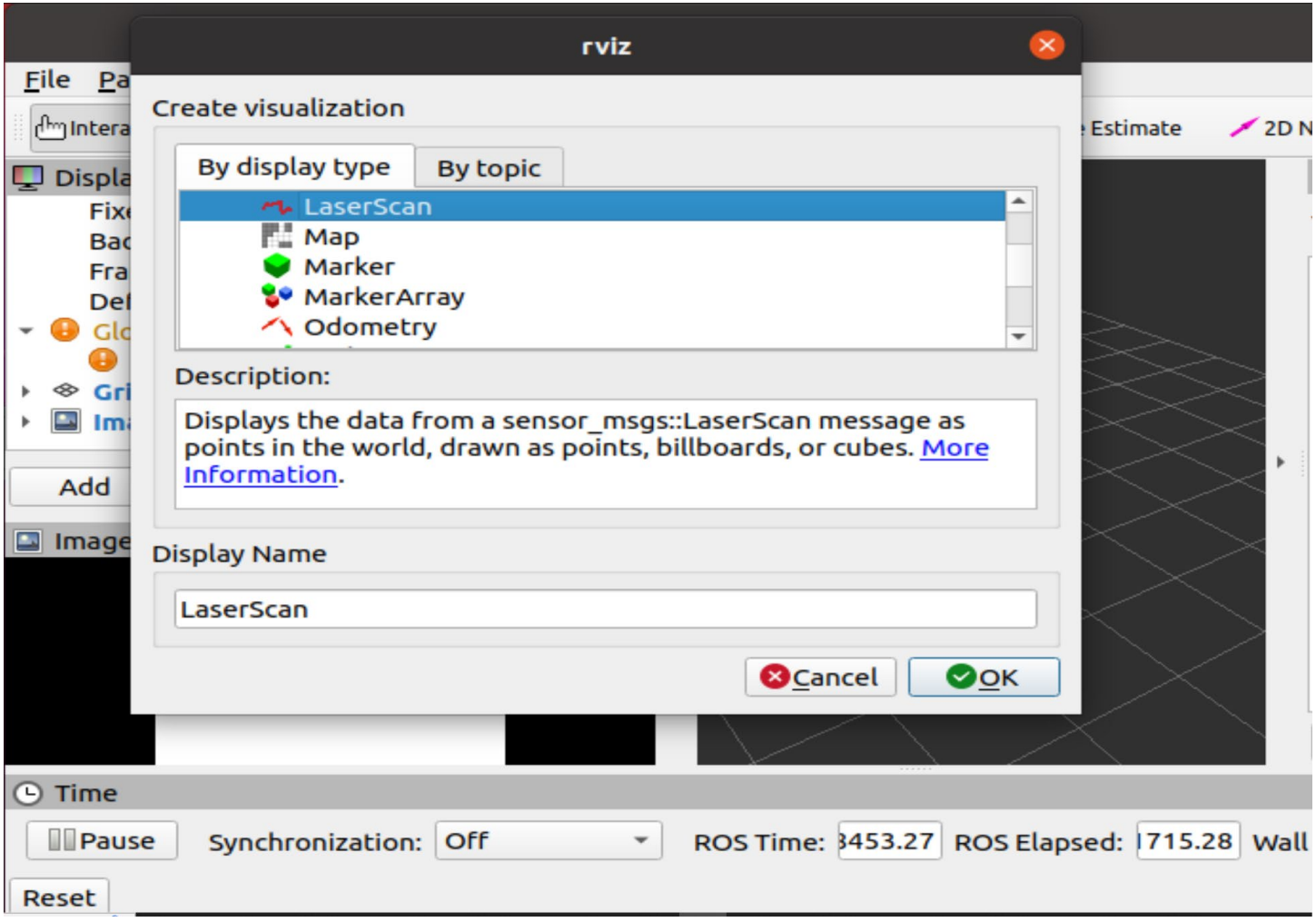
```
ab@ab-c:~$ rosrun rviz rviz
```



RViz

```
ab@ab-c:~$ roscore
```

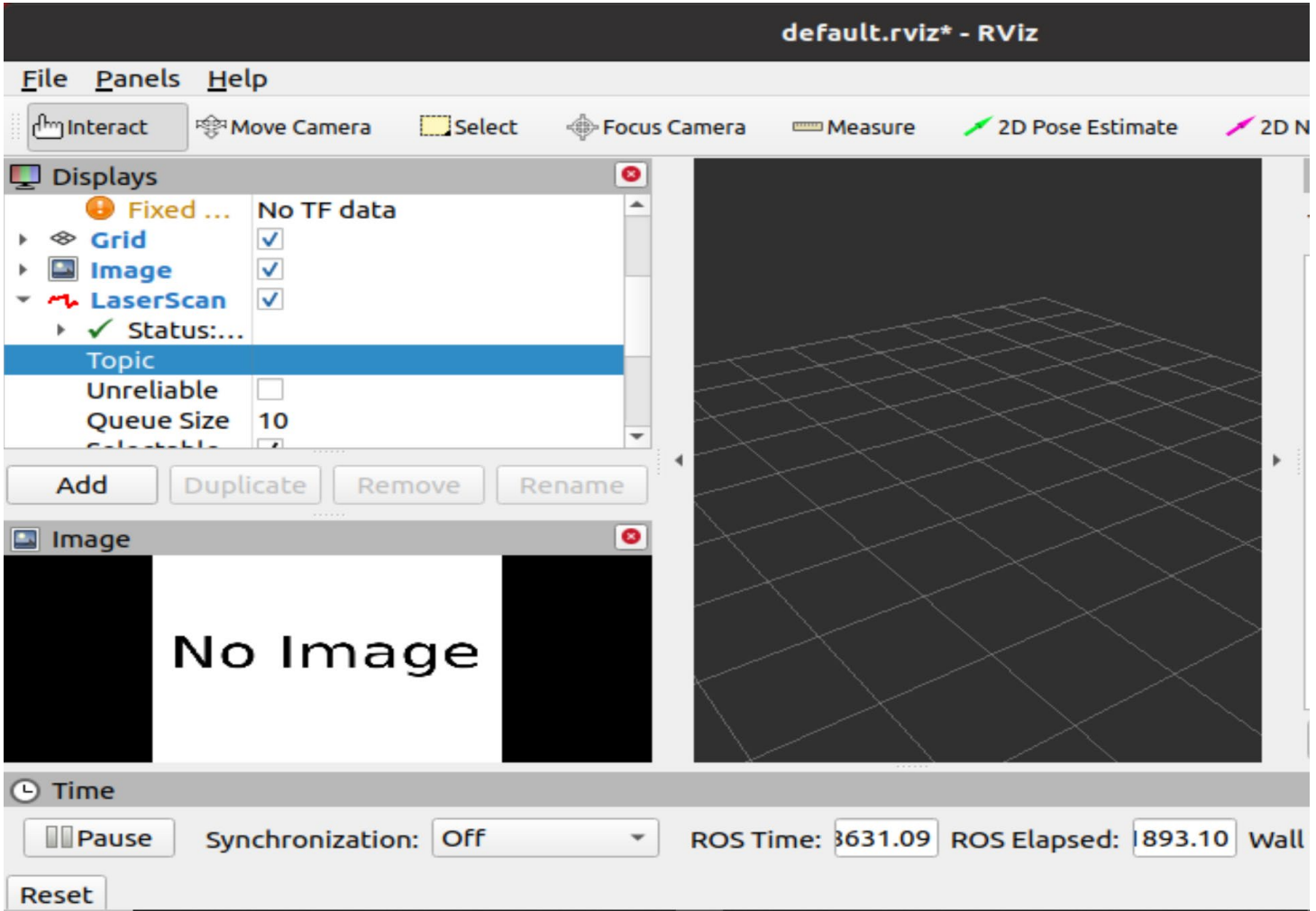
```
ab@ab-c:~$ rosrun rviz rviz
```



RViz

```
ab@ab-c:~$ roscore
```

```
ab@ab-c:~$ rosrun rviz rviz
```



RQT

RQT: Plug-in Type Comprehensive GUI Tool for ROS

- Starting with the ROS Fuerte version, the existing 'rxbag', 'rxplot', 'rxgraph', etc. have been merged with 'rqt'. It is now available as **comprehensive GUI tool** for ROS with plug-ins such as 'rqt_bag', 'rqt_plot', 'rqt_graph', etc.
- Since 'rqt' is developed with 'Qt', users can freely add and develop **plugins**
- Let's take a look at **'rqt_image_view'**, **'rqt_graph'**, **'rqt_plot'**, **'rqt_bag'** which are representative plugins of 'rqt'

- In addition, there are plugins such as
- rqt_action, rqt_gui, rqt_plot, rqt_runtime_monitorrqt_bag, rqt_gui_cpp, rqt_pose_view, rqt_rvizrqt_bag_plugins, rqt_gui_py, rqt_publisher, rqt_service_callerrqt_capabilities, rqt_image_view, rqt_py_common, rqt_shellrqt_console, rqt_launch, rqt_py_console, rqt_srvrqt_controller_manager, rqt_logger_level, rqt_reconfigure, rqt_tf_treerqt_dep, rqt_moveit, rqt_robot_dashboard, rqt_toprqt_ez_publisher, rqt_msg, rqt_robot_monitor, rqt_topicrqt_graph, rqt_nav_view, rqt_robot_steering, rqt_web, etc. (wow.. ——;;)

RQT Plug-in #1

1. Action

- Action Type Browser | Check the data structure of action type

2. Configuration

- Dynamic Reconfigure | Change the GUI setting value to change the setting value provide by the nodes
- Launch | GUI version of 'roslaunch'

3. Introspection

- Node Graph | Graph view showing relationship diagrams and message flow of running nodes
- Package Graph | Graph view showing node dependencies
- Process Monitor | Check CPU utilization, memory usage, and number of threads of running nodes

4. Logging

- Bag | ROS data logging
- Console | Check for messages such as warning, error that occur on the nodes
- Logger Level | Select and display logger information such as Debug, Info, Warn, Error, Fatal

RQT Plug-in #2

5. Miscellaneous Tools

- Python Console | Python console screen
- Shell | Activate shell
- Web | Activate web browser

6. Robot

- Depending on the robot, add a plug-in such as a dashboard

7. Robot Tools

- Controller Manager | Plug-in required to control the controller
- Diagnostic Viewer | Check robot device and error
- Moveit! Monitor | Check 'Moveit!' data used in robot arm planning
- Robot Steering | Robot adjustment GUI tool, used in remote control to steer the robot
- Runtime Monitor | Check for errors and warning on nodes in real time

RQT Plug-in #3

8. Services

- Service Caller | Connect to the running service server and request service
- Service Type Browser | Check the data structure of the service type

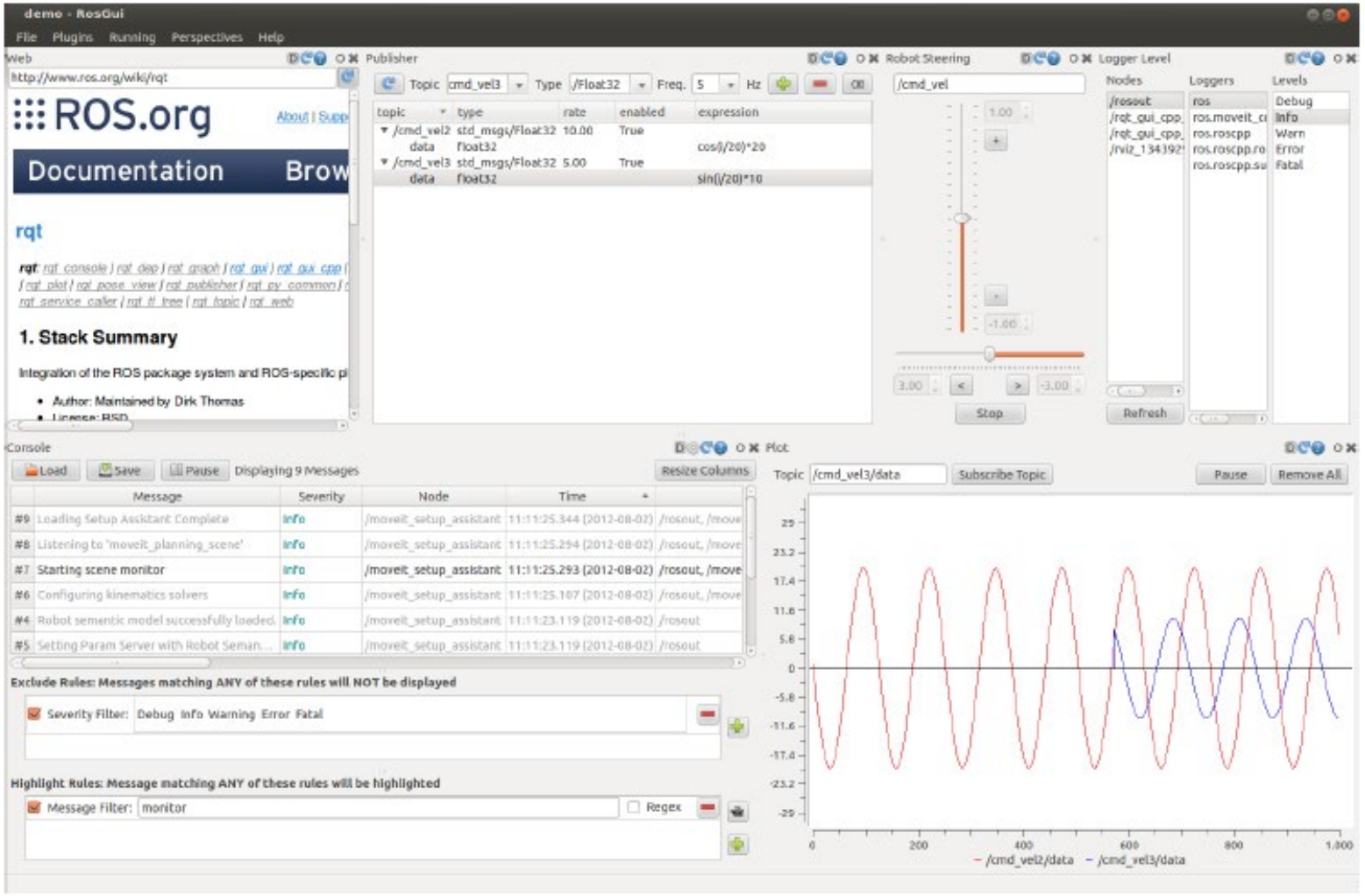
9. Topics

- Easy Message Publisher | Publish topic in GUI environment
- Topic Publisher | Create and publish topic
- Topic Type Browser | Check the data structure of the topic type
- Topic Monitor | Check the information of selected topic

10. Visualization

- Image View | Check image data of camera
- Navigation Viewer | Check location and target point of robot navigation
- Plot | 2D data plot GUI plug-in, 2D data plotting
- Pose View | Show current TF location and model location
- RViz | Rviz plug-in which is 3D visualization tool
- TF Tree | Graph view showing tf relation as a tree structure

RQT Example



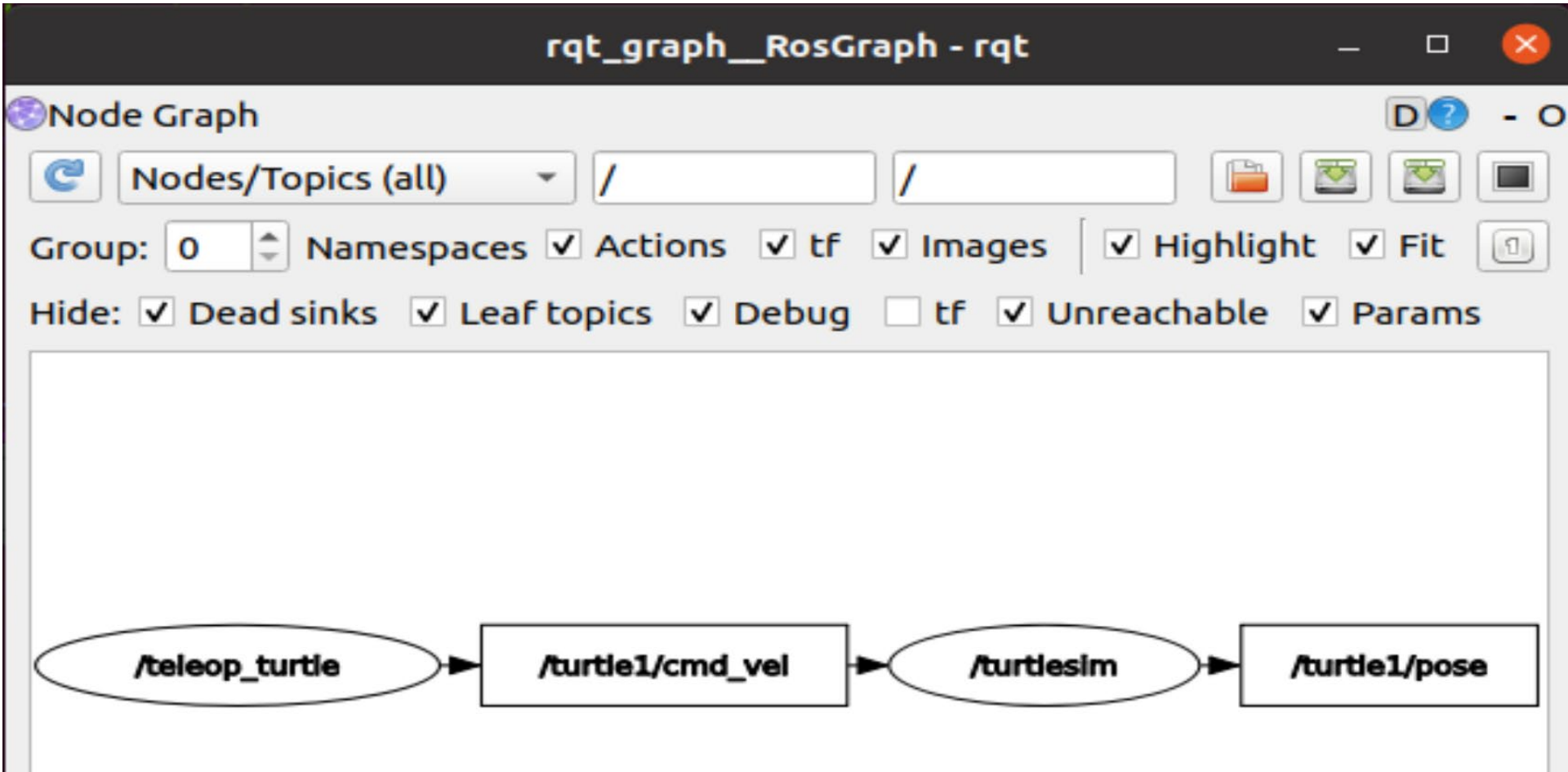
http://www.ros.org/core-components/

rqt_graph

```
ab@ab-c:~$ roscore
```

```
ab@ab-c:~$ rosrun turtlesim turtlesim_node
```

```
ab@ab-c:~$ rqt_graph
```

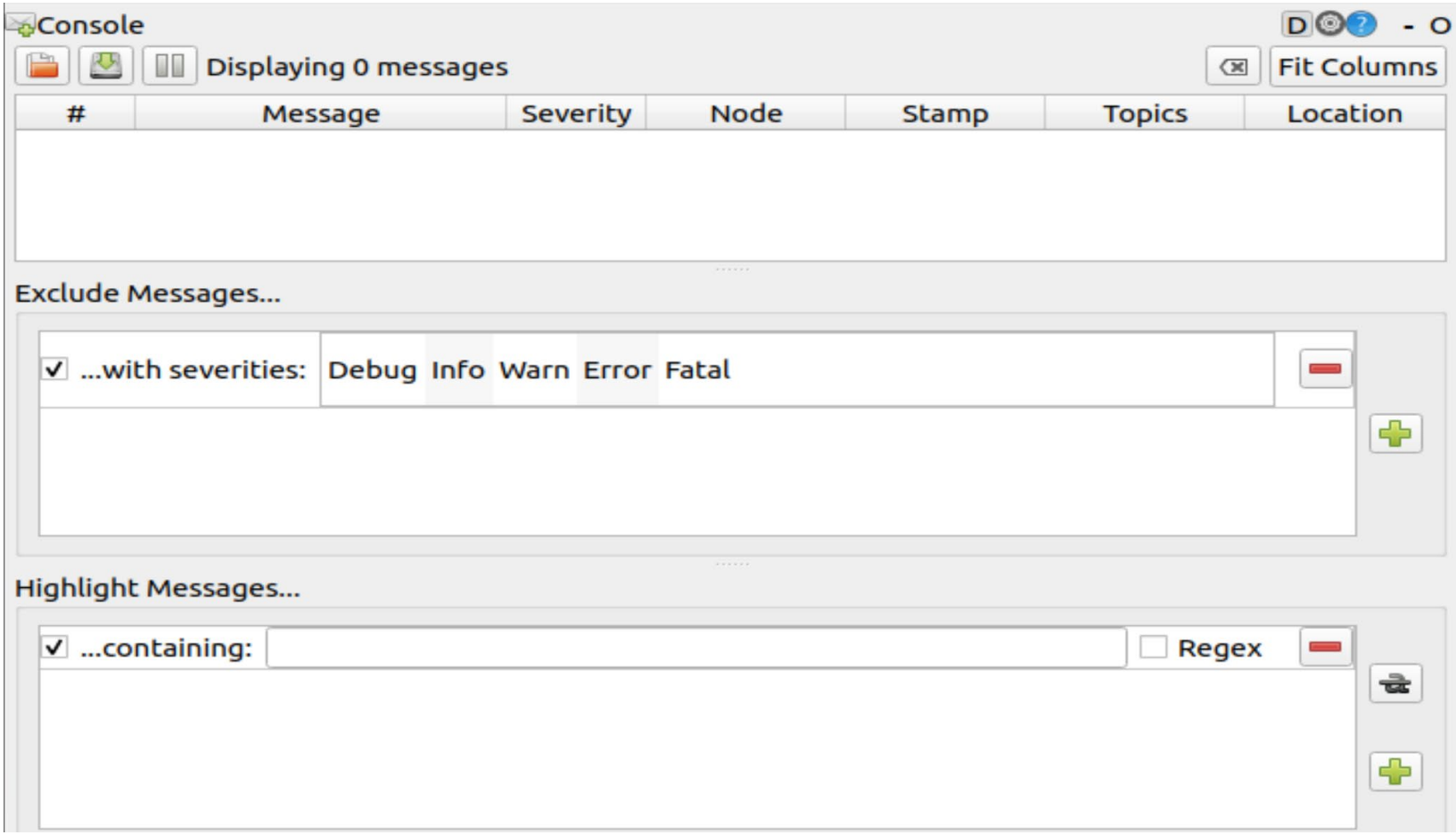


rqt_console

```
ab@ab-c:~$ roscore
```

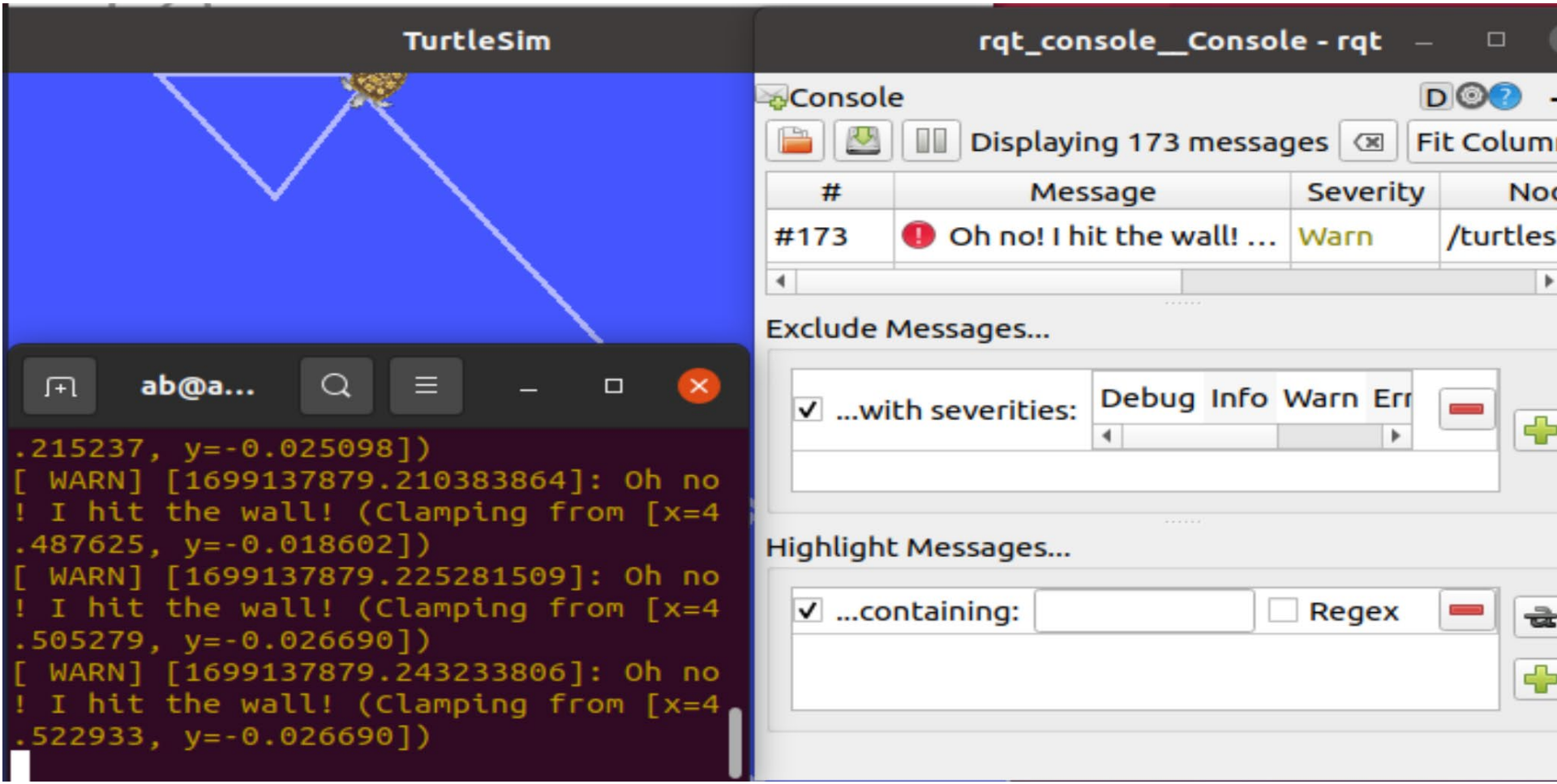
```
ab@ab-c:~$ rosrun turtlesim turtlesim_node
```

```
ab@ab-c:~$ rqt_console
```



rqt_console

```
ab@ab-c:~$ rosrun turtlesim turtle_teleop_key
```

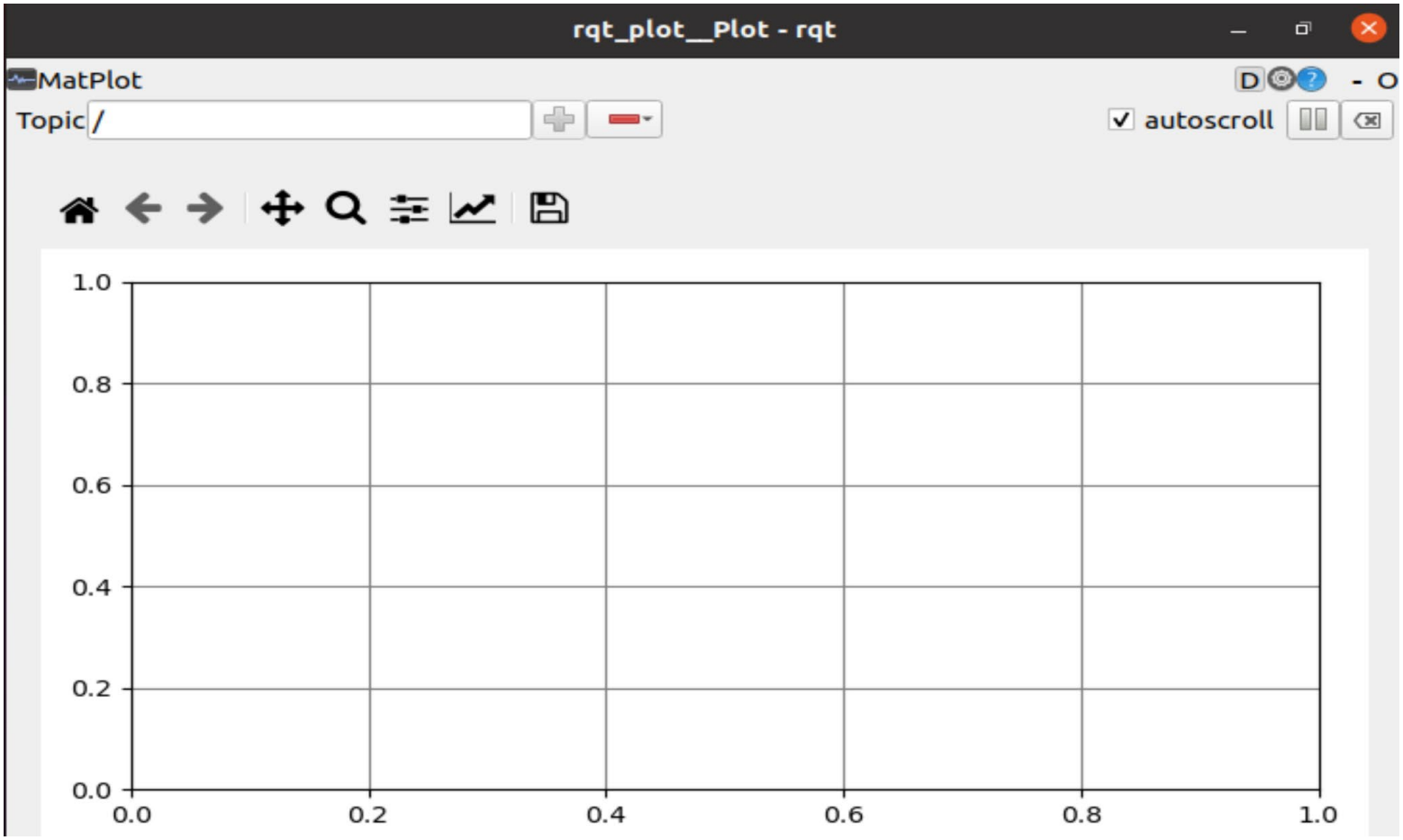


rqt_plot

```
ab@ab-c:~$ roscore
```

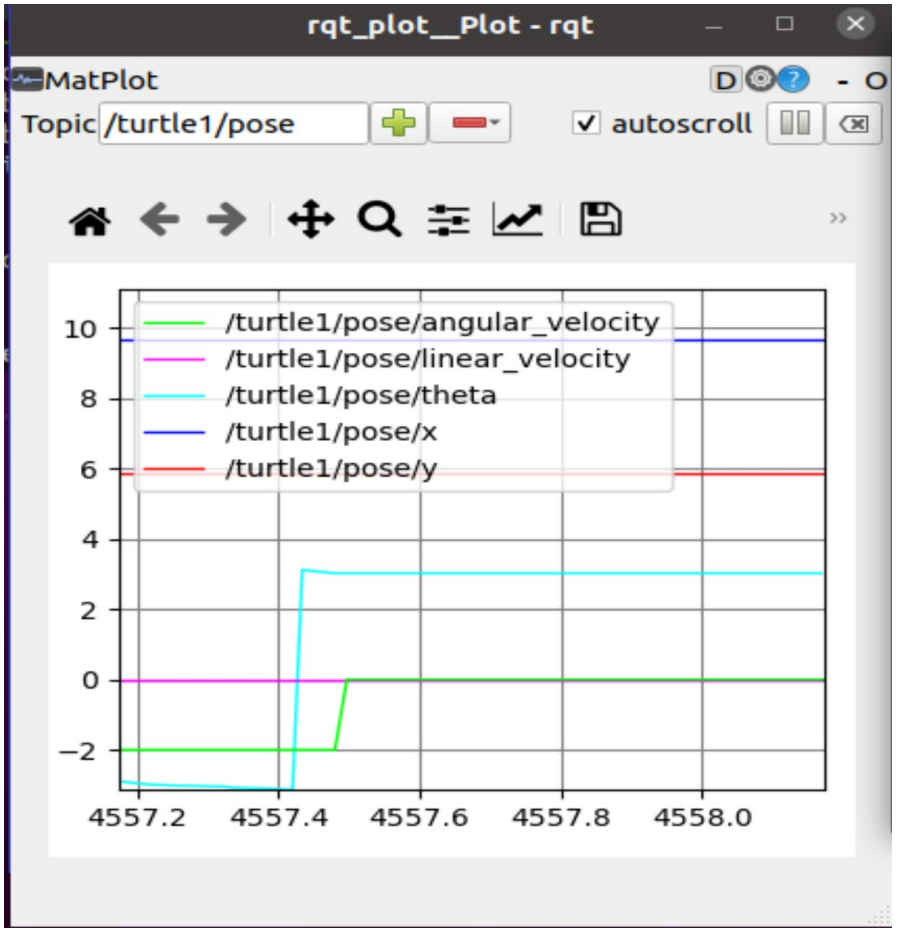
```
ab@ab-c:~$ rosrun turtlesim turtlesim_node
```

```
ab@ab-c:~$ rqt_plot
```



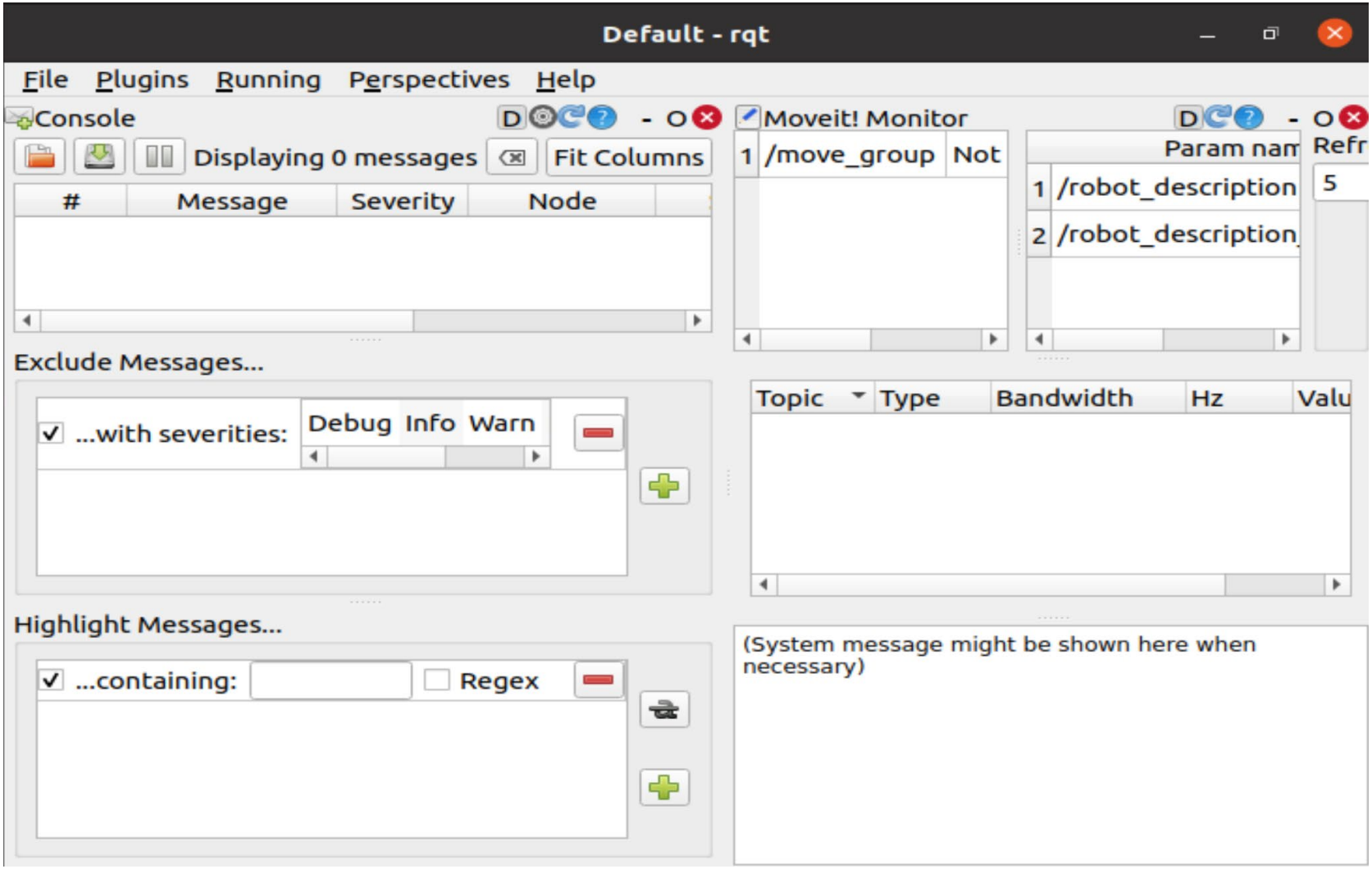
rqt_plot

```
ab@ab-c:~$ rosrun turtlesim turtle_teleop_key
```



rqt

```
ab@ab-c:~$ rqt
```

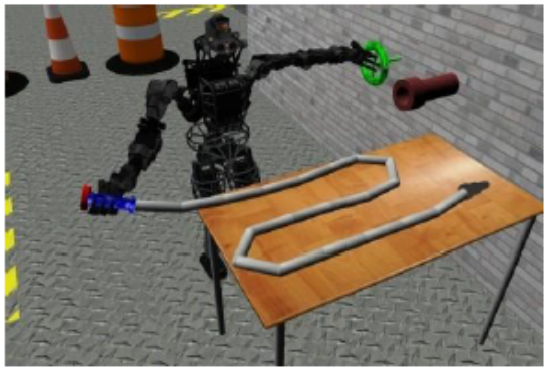


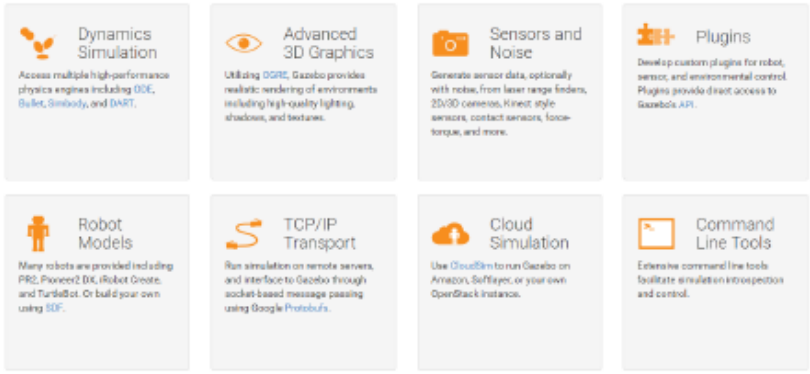
Gazebo

Gazebo

- Gazebo is 3D Simulator with Physics Engine, Robot model, Sensor, Environment model, and so on. It helps you to get data similar to the one in real environment.
- Gazebo is regarded as the best simulator among recently-introduced Open Simulator. In addition, it is selected as an **official simulator for DARPA Robotics Challenge**
- It is highly compatible with ROS.

GAZEBO





http://gazebosim.org/

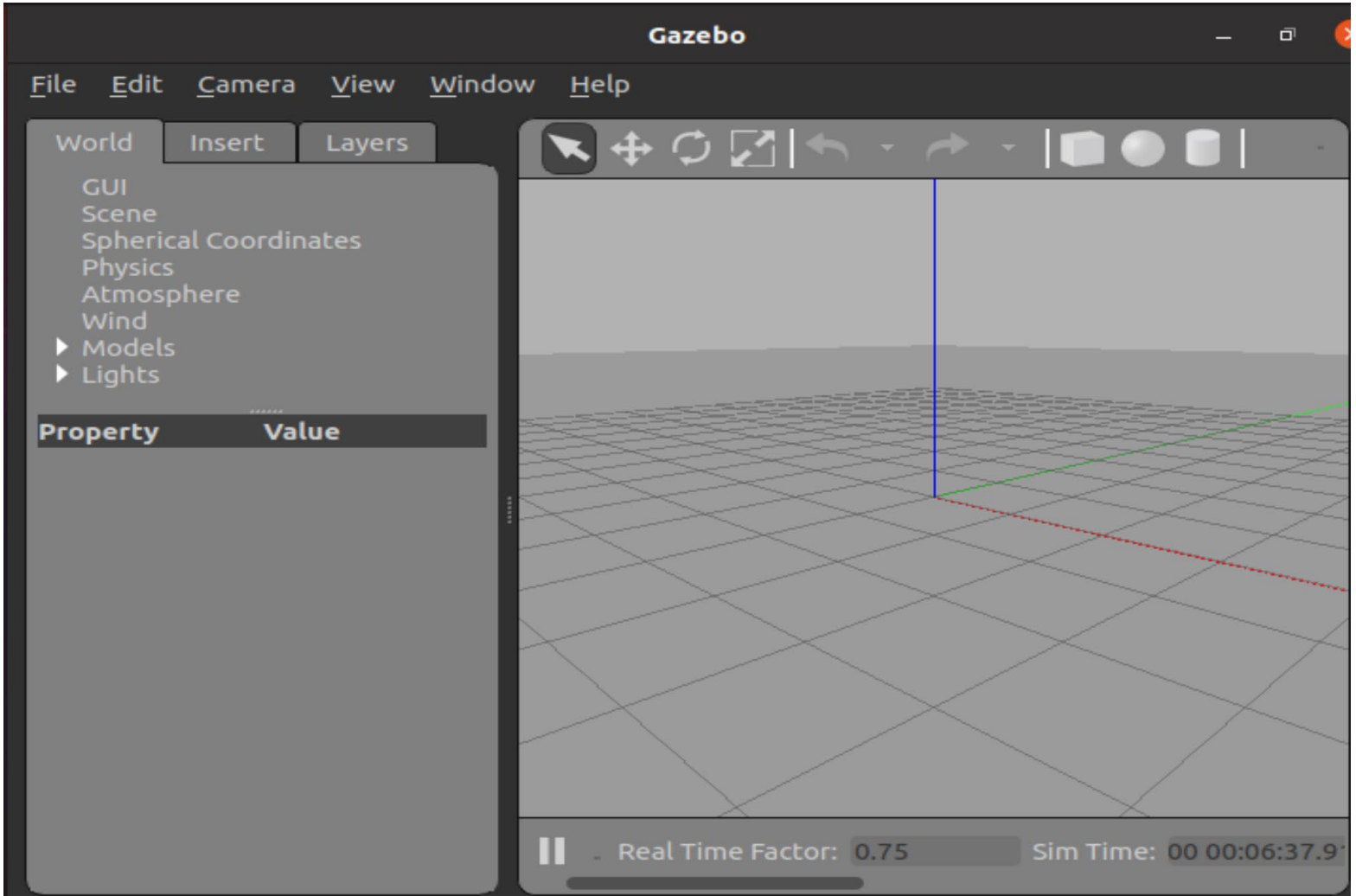
Gazebo

https://www.youtube.com/watch?v=R3xUKYcG_bc

Gazebo

```
ab@ab-c:~$ sudo apt install ros-noetic-gazebo-*
```

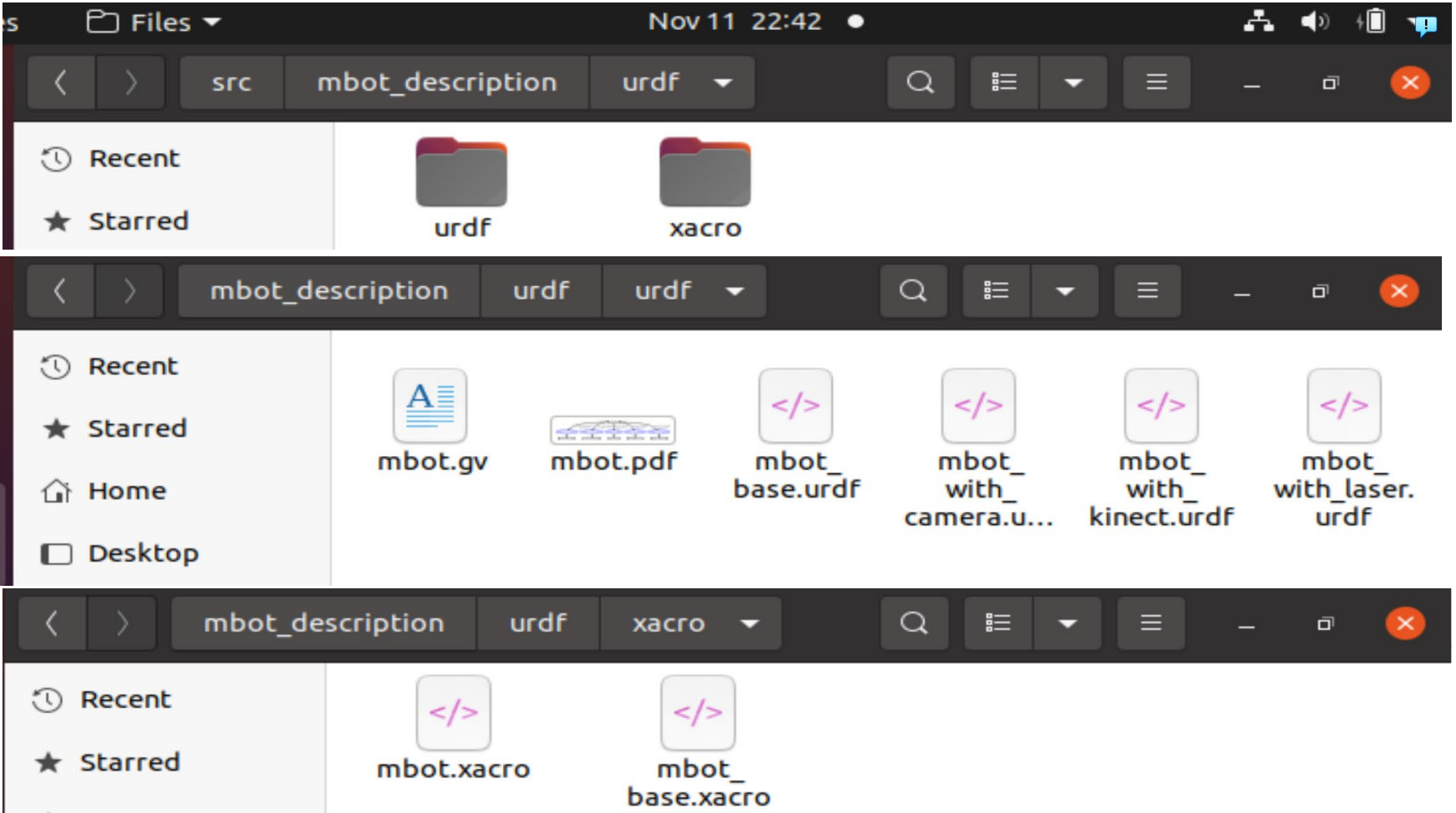
```
ab@ab-c:~$ gazebo gazebo
```



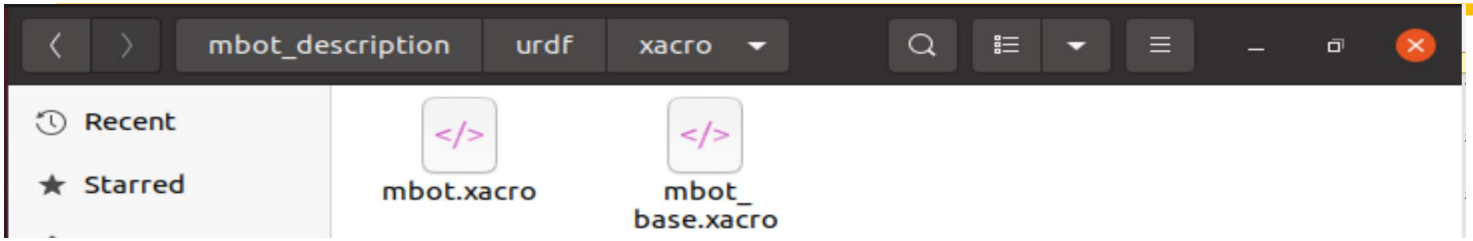
Lecture 16: URDF Model Optimization

Create New sub-Folders in urdf

Put files built in last class into sub-folder “urdf”, copy paste mbot.xacro, mbot_base.xacro into sub-folder “xacro”



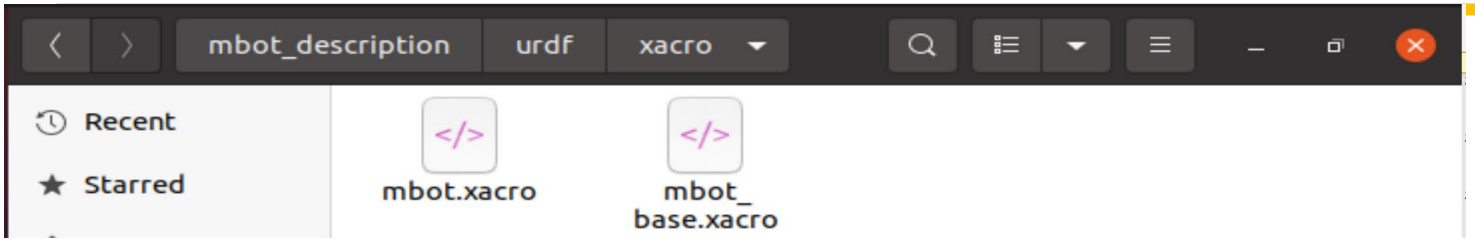
Create mbot_base.xacro



Open | mbot_base.xacro | ~/catkin_ws/src/mbot_description/urdf/xacro | Save

```
1  <?xml version="1.0"?>
2  <robot name="mbot" xmlns:xacro="http://www.ros.org/wiki/xacro">
3
4      <!-- PROPERTY LIST -->
5      <xacro:property name="M_PI" value="3.1415926"/>
6      <xacro:property name="base_radius" value="0.20"/>
7      <xacro:property name="base_length" value="0.16"/>
8
9      <xacro:property name="wheel_radius" value="0.06"/>
10     <xacro:property name="wheel_length" value="0.025"/>
11     <xacro:property name="wheel_joint_y" value="0.19"/>
12     <xacro:property name="wheel_joint_z" value="0.05"/>
13
14     <xacro:property name="caster_radius"  value="0.015"/> <!--
   wheel_radius - ( base_length/2 - wheel_joint_z) -->
15     <xacro:property name="caster_joint_x" value="0.18"/>
16
17     <!-- Defining the colors used in this robot -->
18     <material name="yellow">
19         <color rgba="1 0.4 0 1"/>
20     </material>
21     <material name="black">
22         <color rgba="0 0 0 0.95"/>
23     </material>
24     <material name="gray">
25         <color rgba="0.75 0.75 0.75 1"/>
26     </material>
27
```

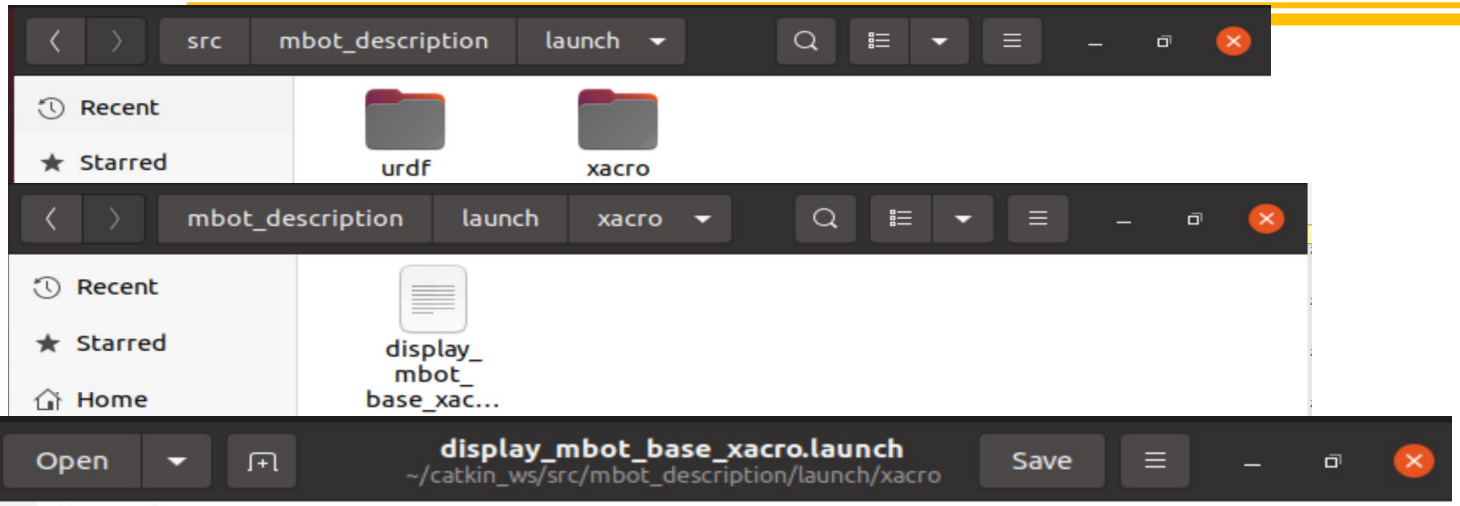
Create mbot.xacro



Open
mbot.xacro
~/catkin_ws/src/mbot_description/urdf/xacro
Save

```
1 <?xml version="1.0"?>
2 <robot name="mbot" xmlns:xacro="http://www.ros.org/wiki/xacro">
3
4     <xacro:include filename="$(find mbot_description)/urdf/xacro/-
  mbot_base.xacro" />
5
6     <xacro:mbot_base/>
7
8 </robot>
```

Create display_mbot_base_xacro.launch



Open | display_mbot_base_xacro.launch | ~/catkin_ws/src/mbot_description/launch/xacro | Save

```
1  <launch>
2          <arg name="model" default="$(find xacro)/xacro --inorder '$(find
   mbot_description)/urdf/xacro/mbot.xacro'" />
3          <arg name="gui" default="true" />
4
5          <param name="robot_description" command="$(arg model)" />
6
7
8          <param name="use_gui" value="$(arg gui)"/>
9
10
11         <node name="joint_state_publisher" pkg="joint_state_publisher"
   type="joint_state_publisher" />
12
13
14         <node name="robot_state_publisher" pkg="robot_state_publisher"
   type="robot_state_publisher" />
15
16
17         <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
   mbot_description)/config/mbot.rviz" required="true" />
18
19 </launch>
20
```

Create mbot.rviz

Open | mbot.rviz ~/catkin_ws/src/mbot_description/config | Save

```
1  Panels:
2    - Class: rviz/Displays
3      Help Height: 78
4      Name: Displays
5      Property Tree Widget:
6        Expanded:
7          - /Global Options1
8          - /Status1
9          - /TF1
10       Splitter Ratio: 0.5
11     Tree Height: 473
12   - Class: rviz/Selection
13     Name: Selection
14   - Class: rviz/Tool Properties
15     Expanded:
16       - /2D Pose Estimate1
17       - /2D Nav Goal1
18       - /Publish Point1
19     Name: Tool Properties
20     Splitter Ratio: 0.588679016
21   - Class: rviz/Views
22     Expanded:
23       - /Current View1
24     Name: Views
25     Splitter Ratio: 0.5
26   - Class: rviz/Time
27     Experimental: false
```

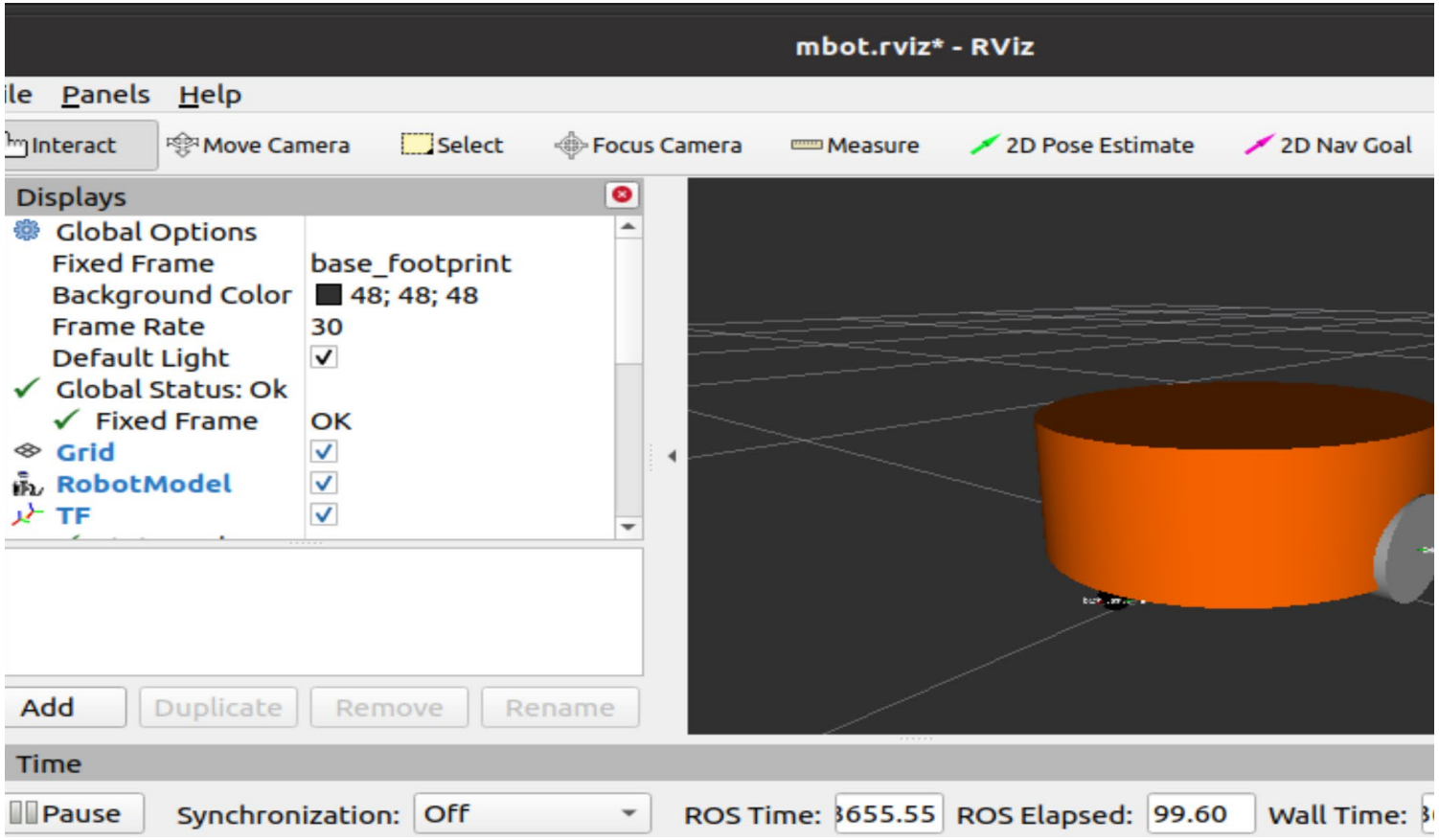
Display Model

Method 1:

```
ab@ab-c:~/catkin_ws/src/mbot_description/urdf/xacro$ rosrun xacro xacro mbot.xa
cro>mbot.urdf
```

Method 2:

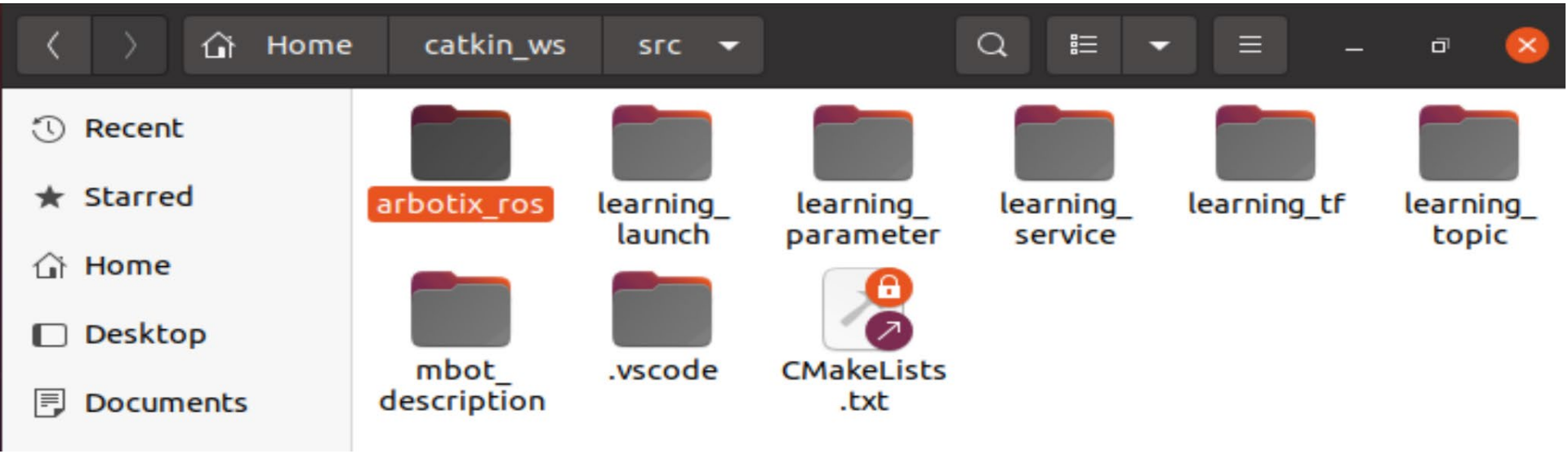
```
ab@ab-c:~$ roslaunch mbot_description display_mbot_base_xacro.launch
```



Lecture 17: ArbotiX & RViz

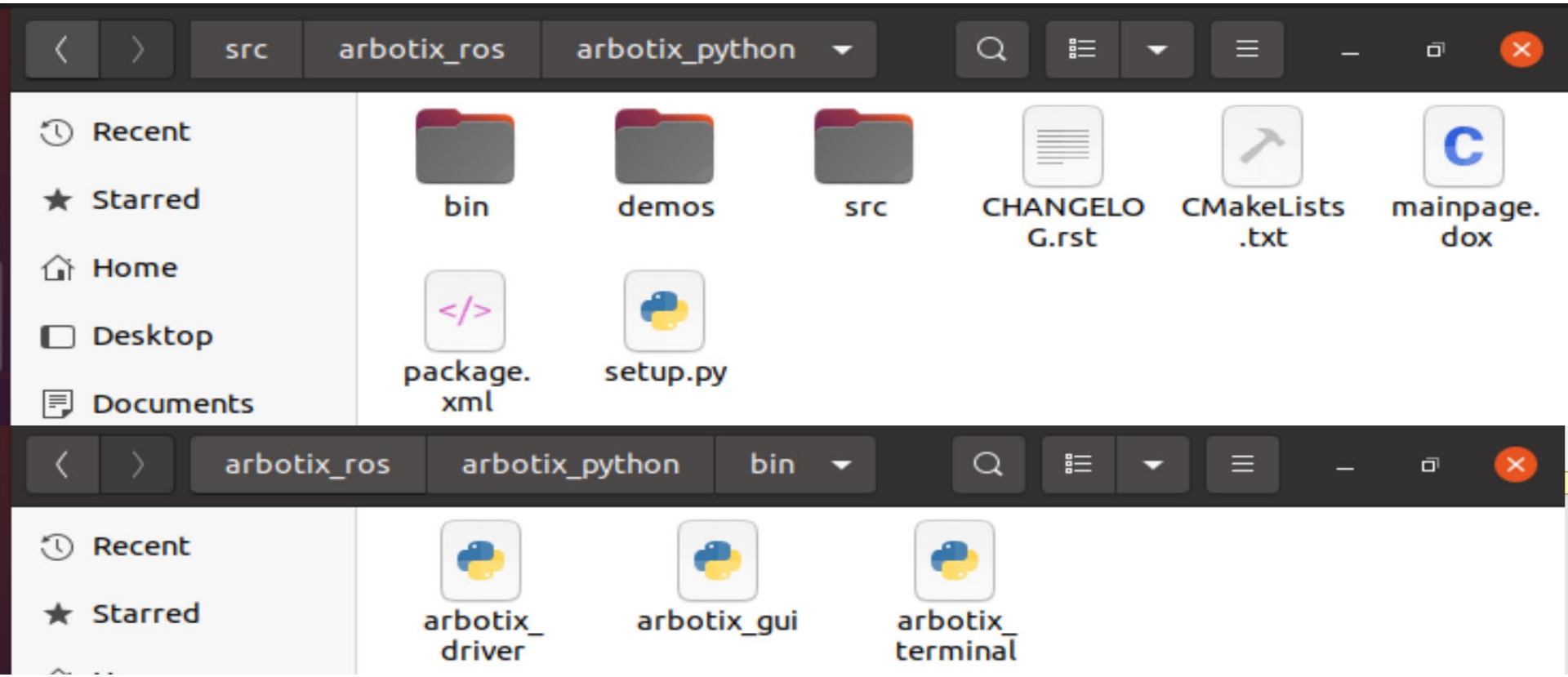
Install ArbotiX

```
ab@ab-c:~/catkin_ws/src$ sudo apt install git
ab@ab-c:~/catkin_ws/src$ git clone https://github.com/vanadiumlabs/arbotix_ros.git
ab@ab-c:~/catkin_ws$ catkin_make
ab@ab-c:~/catkin_ws$ source devel/setup.bash
```

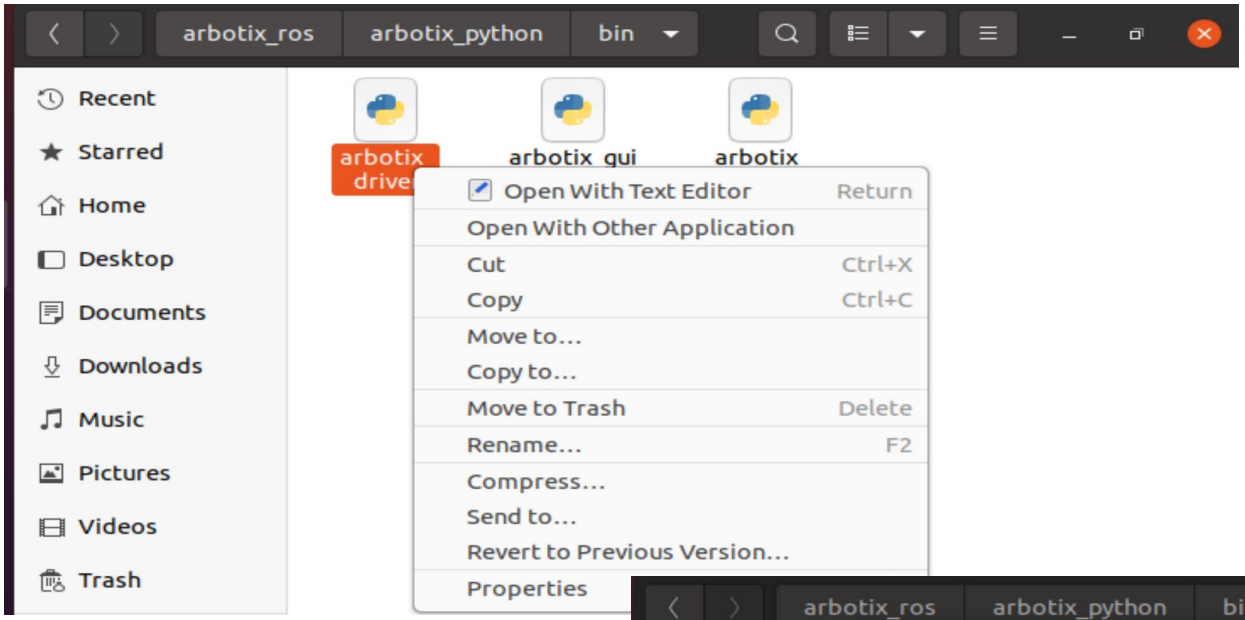


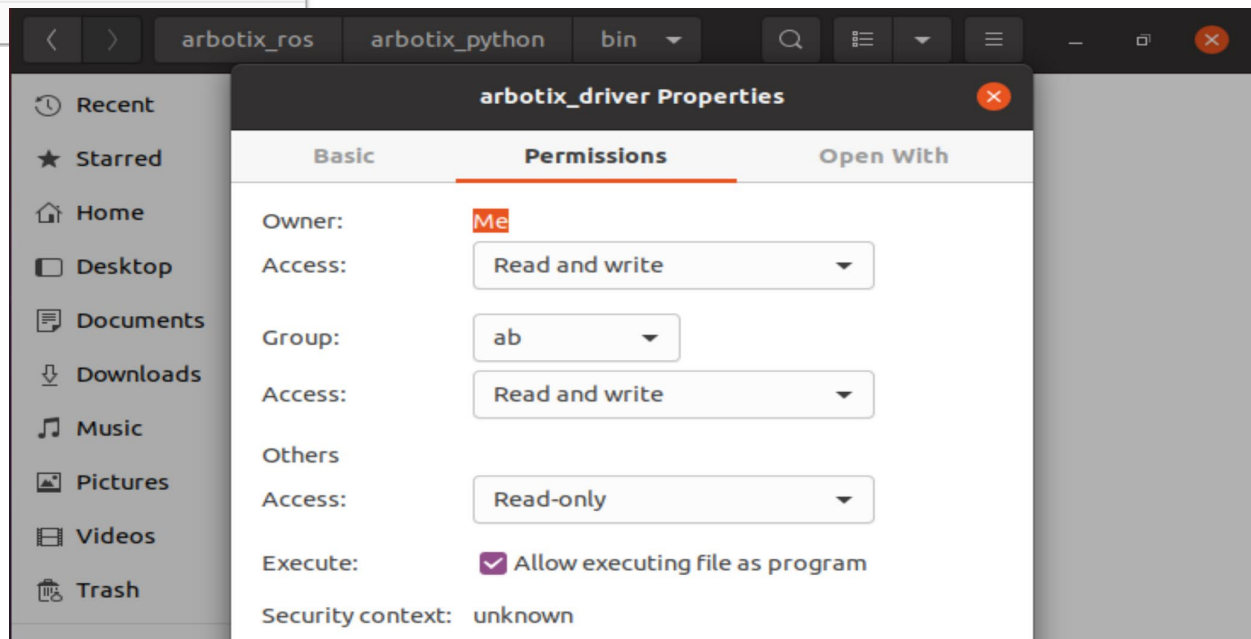
arbotix_python

Allow executing file as program

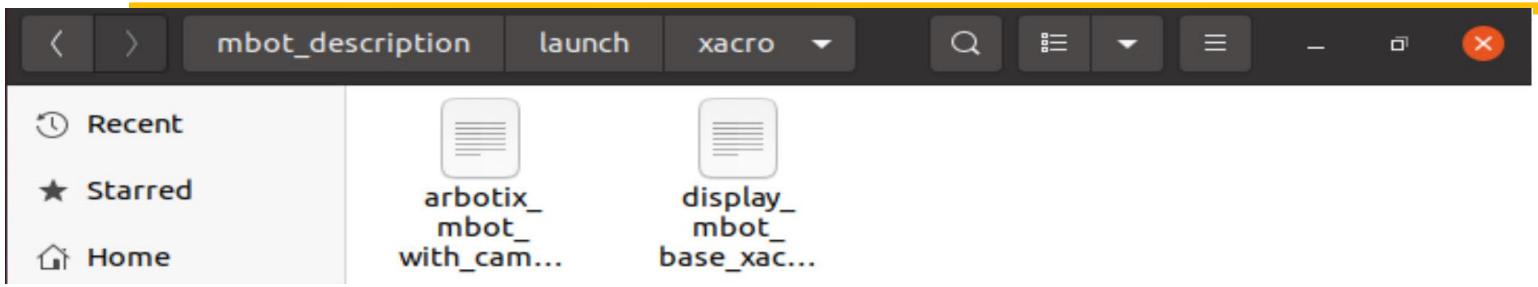


arbotix_python





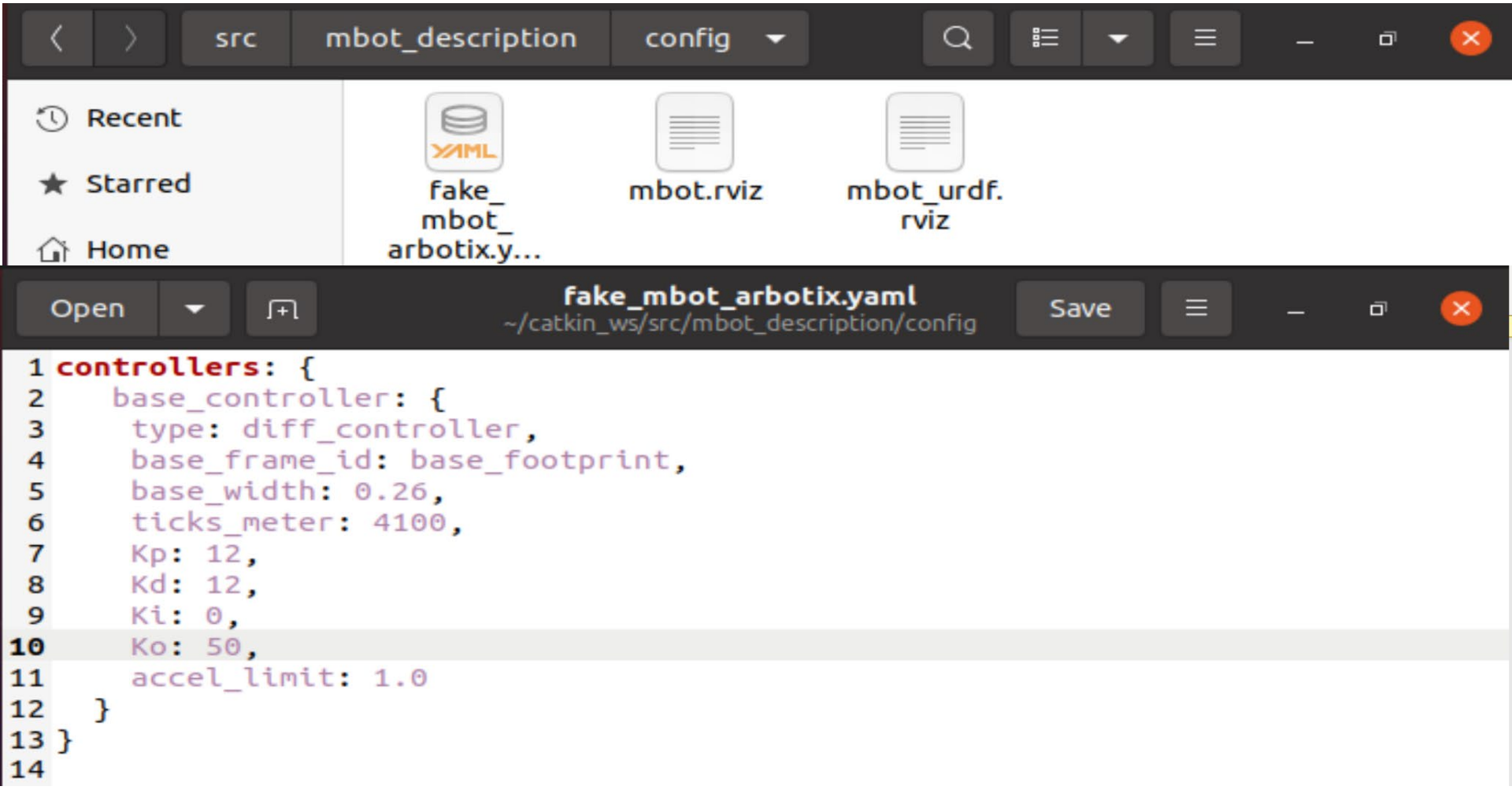
Create arbotix_mbot_with_camera_xacro.launch



Open | arbotix_mbot_with_camera_xacro.launch ~/catkin_ws/src/mbot_description/launch/xacro | Save

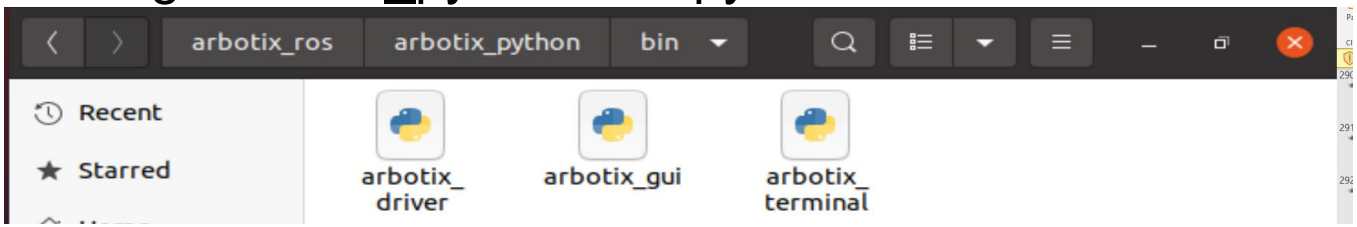
```
1  <launch>
2          <arg name="model" default="$(find xacro)/xacro --inorder '$(find
   mbot_description)/urdf/xacro/mbot_with_camera.xacro'" />
3          <arg name="gui" default="false" />
4
5          <param name="robot_description" command="$(arg model)" />
6
7
8          <param name="use_gui" value="$(arg gui)"/>
9
10
11         <node name="arbotix" pkg="arbotix_python" type="arbotix_driver"
   output="screen" >
12         <rosparam file="$(find mbot_description)/config/-
   fake_mbot_arbotix.yaml" command="load" />
13         <param name="sim" value="true"/>
14       </node>
15
16         <node name="joint_state_publisher" pkg="joint_state_publisher"
   type="joint_state_publisher" />
17
18     <node name="robot_state_publisher" pkg="robot_state_publisher"
   type="robot_state_publisher" />
19         <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
   mbot_description)/config/mbot_arbotix.rviz" required="true" />
20
21 </launch>
22
```

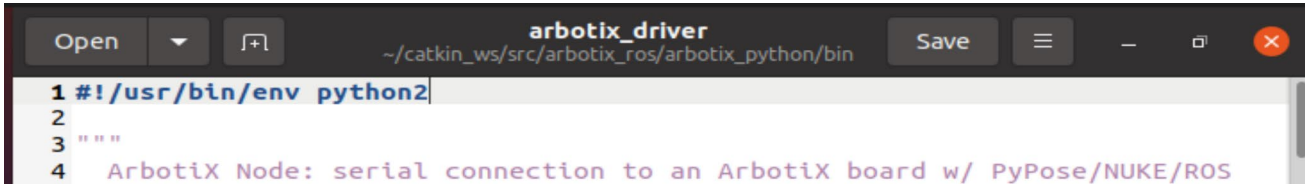
Create fake_mbot_arbotix.yaml



roslaunch

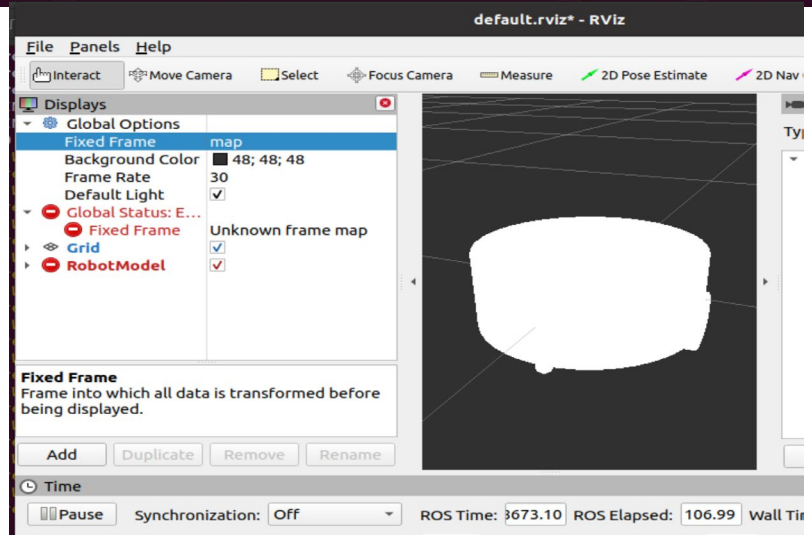
Change arbotix_python 3 to python2



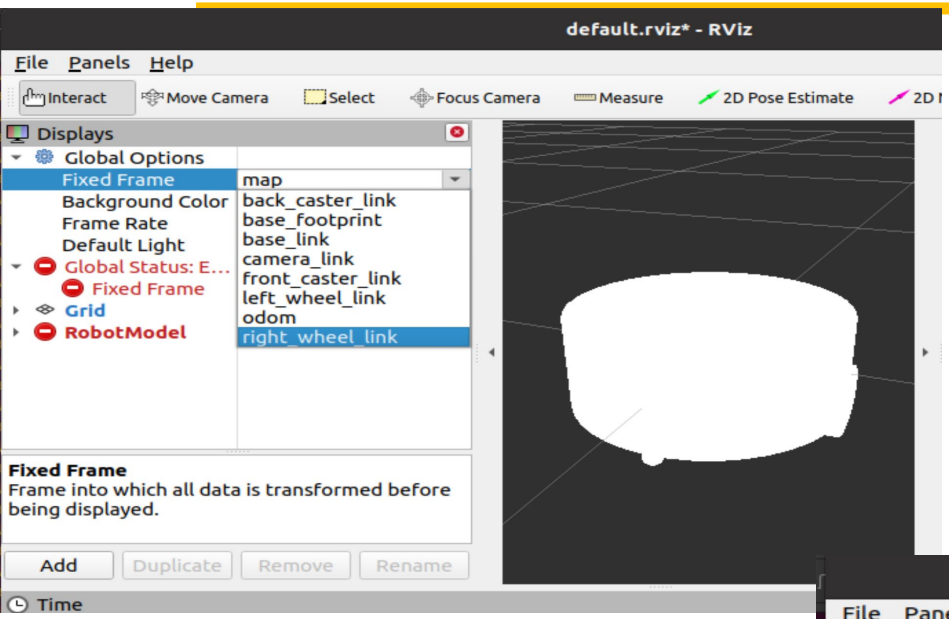


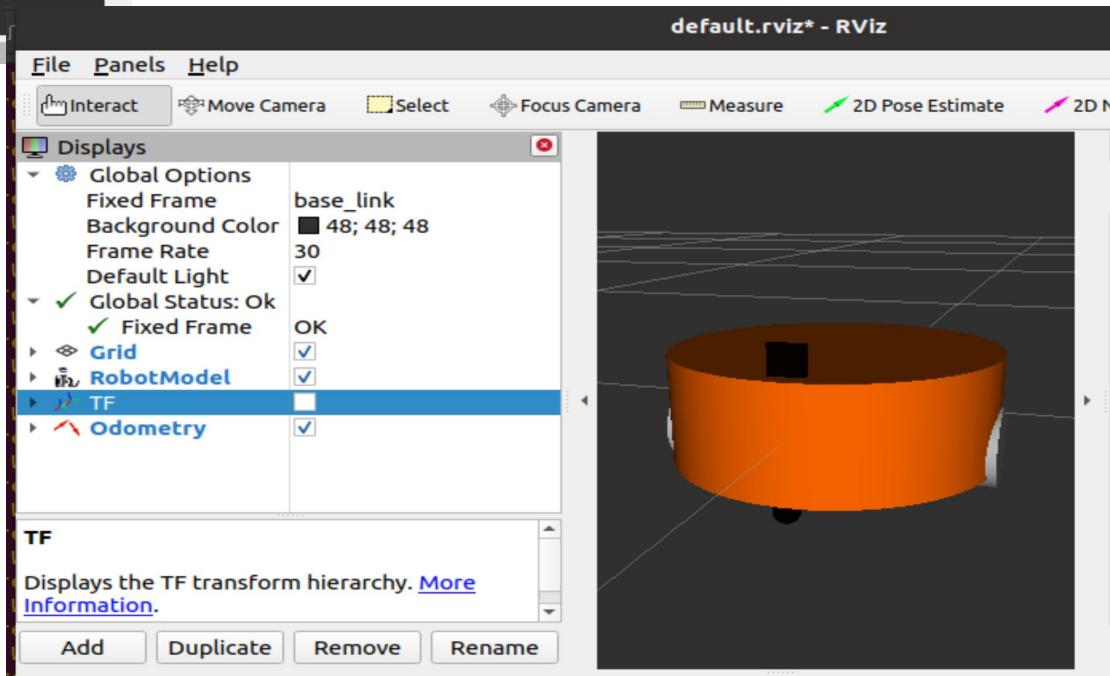

```
ab@ab-c:~/catkin_ws$ catkin_make
```

```
ab@ab-c:~$ roslaunch mbot_description arbotix_mbot_with_camera_xacro.launch
```



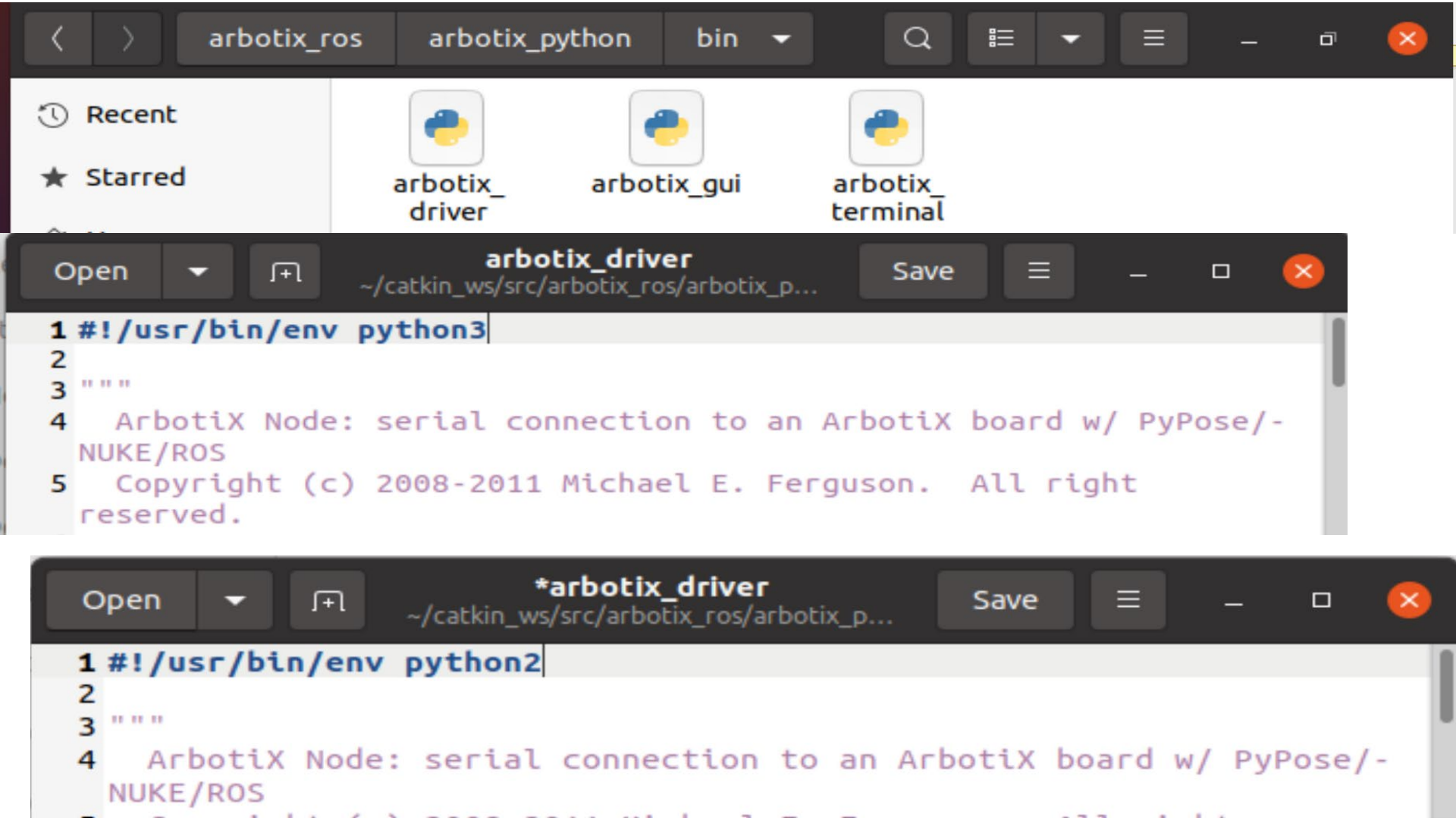
Trouble shooting #1:
RViz ->fixed frame -> “base_link”



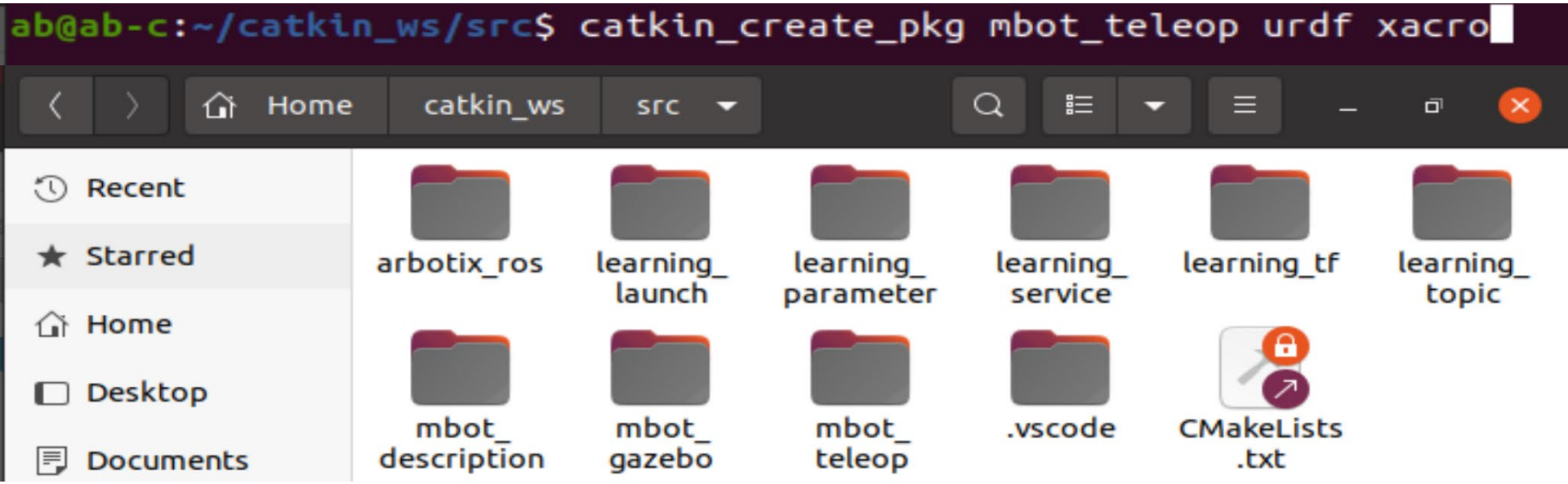


Trouble shooting #2: Arbotix_python -> bin -> python 3

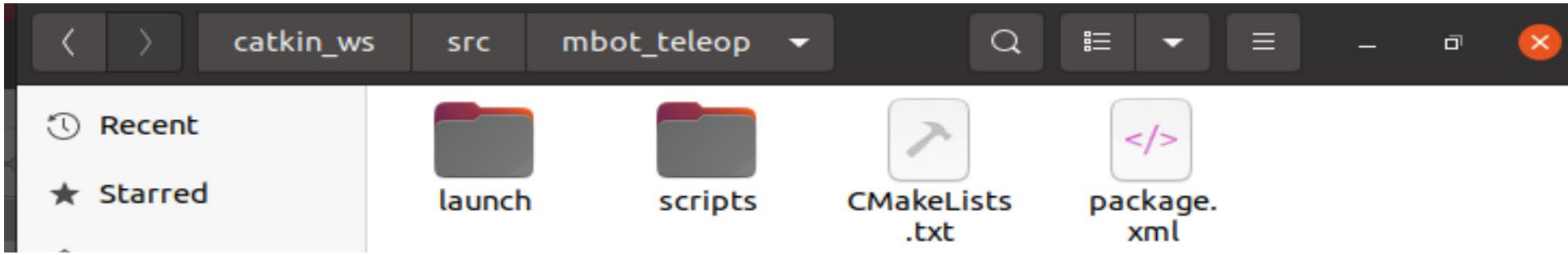
Change python3 to python2



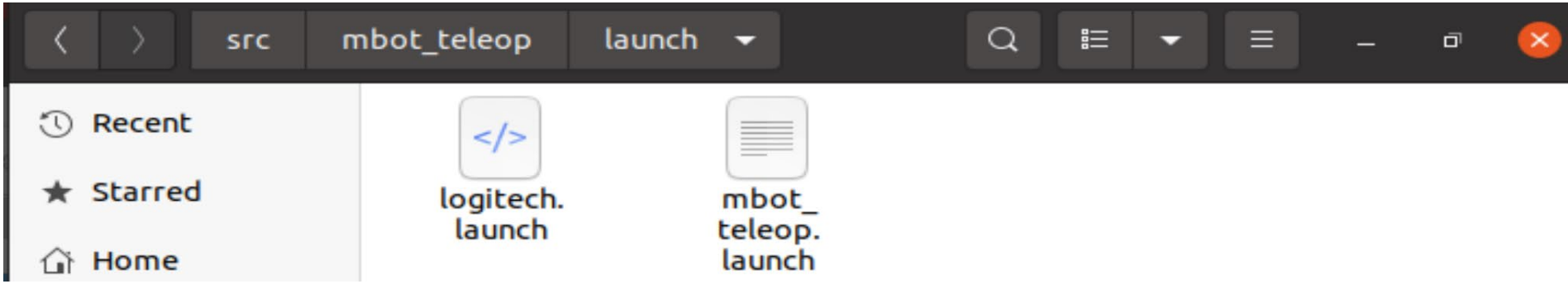
Create mobot_teleop Package



Create New Folders

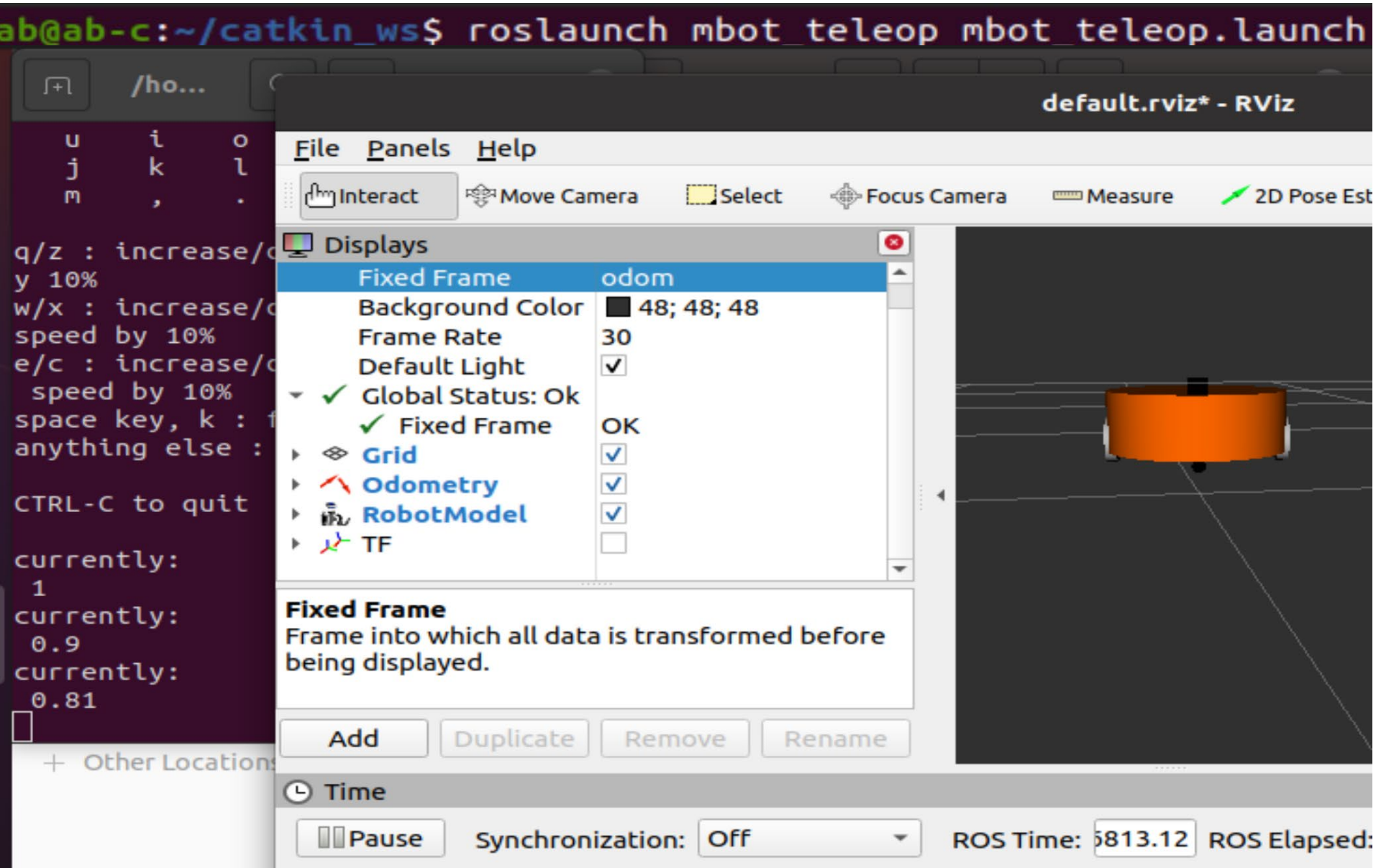


Create mbot_teleop.launch



Create scripts

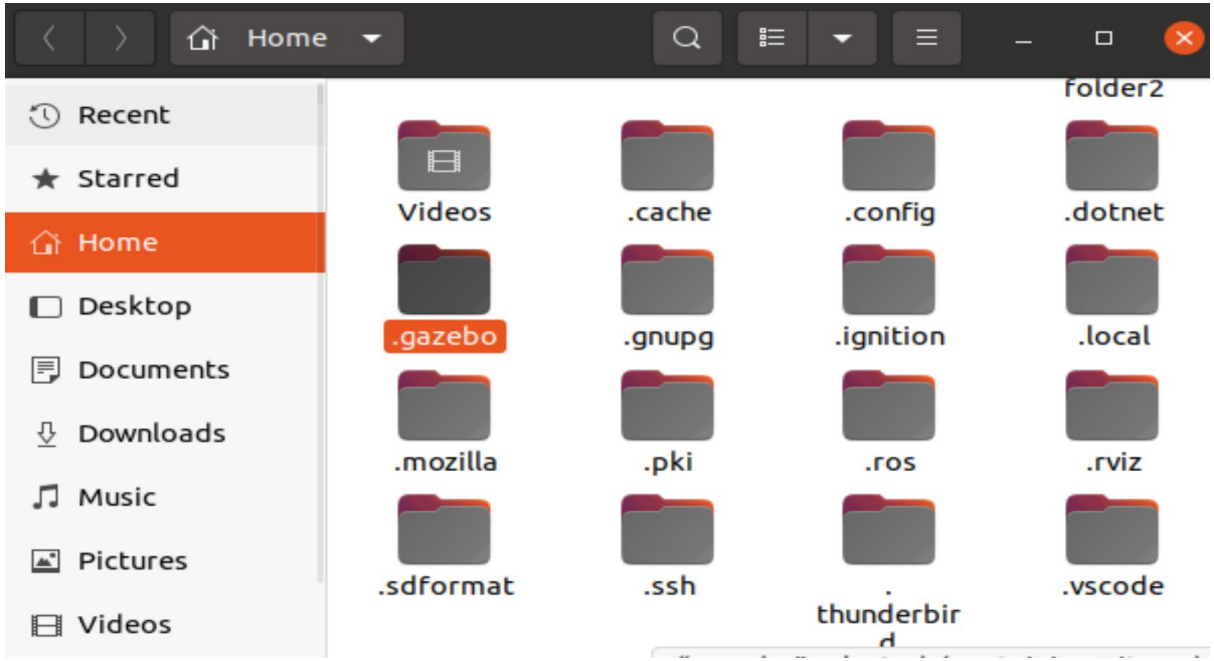
roslaunch


ab@ab-c:~/catkin_ws$ roslaunch mbot_teleop mbot_teleop.launch
u i o
j k l
m , .
q/z : increase/
y 10%
w/x : increase/
speed by 10%
e/c : increase/
speed by 10%
space key, k :
anything else :
CTRL-C to quit
currently:
1
currently:
0.9
currently:
0.81
Other Locations
default.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Est
Displays
Fixed Frame odom
Background Color 48; 48; 48
Frame Rate 30
Default Light
Global Status: Ok
Fixed Frame OK
Grid
Odometry
RobotModel
TF
Fixed Frame
Frame into which all data is transformed before being displayed.
Add
Duplicate
Remove
Rename
Time
Pause
Synchronization: Off
ROS Time: 5813.12
ROS Elapsed:

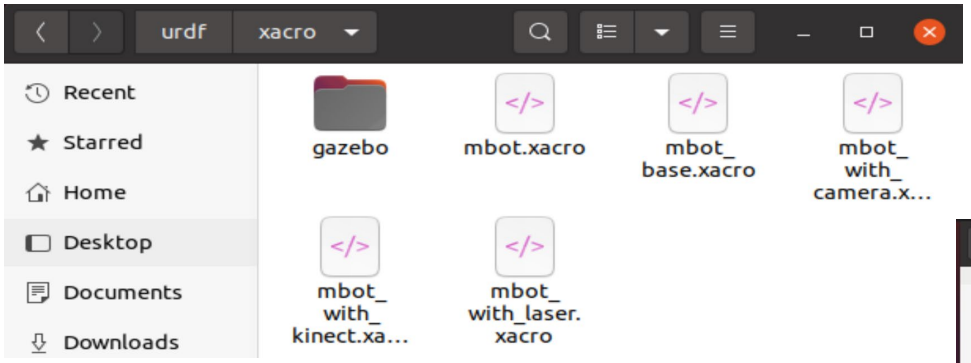
Lecture 18: Gazebo

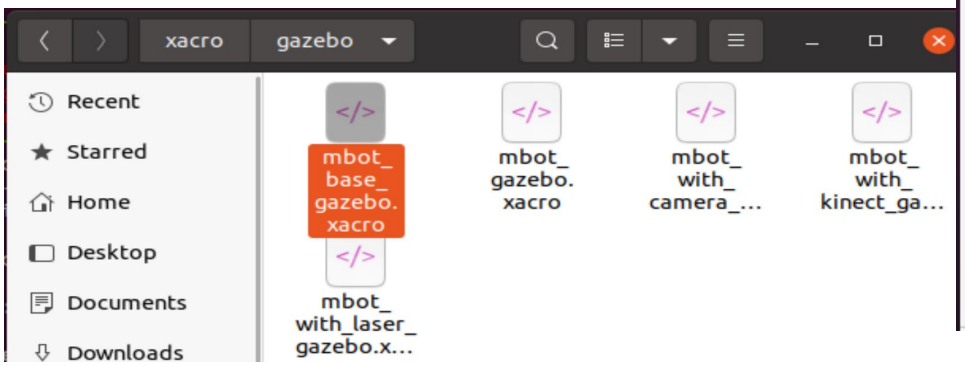
Install Gazebo_models

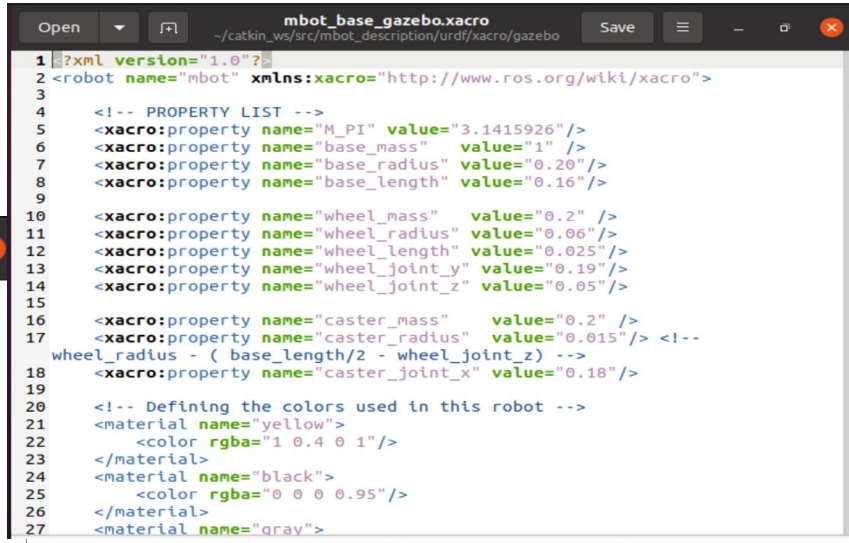
Download:
https://github.com/osrf/gazebo_models



Create the Physical Simulation Model mbot_base_gazebo.xacro







Create the Physical Simulation Model

1. Add link of “collision” and “inertial”

```
        <collision>
            <origin xyz="0 0 0" rpy="${M_PI/2} 0 0" />
            <geometry>
                <cylinder radius="${wheel_radius}" length = "$-
{wheel_length}"/>
            </geometry>
        </collision>
        <cylinder_inertial_matrix  m="${wheel_mass}" r="$-
{wheel_radius}" h="${wheel_length}" />
    </link>
```

```
<!-- Macro for inertia matrix -->
<xacro:macro name="sphere_inertial_matrix" params="m r">
    <inertial>
        <mass value="${m}" />
        <inertia ixx="${2*m*r*r/5}" ixy="0" ixz="0"
            iyy="${2*m*r*r/5}" iyz="0"
            izz="${2*m*r*r/5}" />
    </inertial>
</xacro:macro>

<xacro:macro name="cylinder_inertial_matrix" params="m r h">
    <inertial>
        <mass value="${m}" />
        <inertia ixx="${m*(3*r*r+h*h)/12}" ixy = "0" ixz = "0"
            iyy="${m*(3*r*r+h*h)/12}" iyz = "0"
            izz="${m*r*r/2}" />
    </inertial>
</xacro:macro>
```

Create the Physical Simulation Model

2. Add “gazebo” tag

```
<gazebo reference="${prefix}_wheel_link">
    <material>Gazebo/Gray</material>
</gazebo>

<gazebo reference="base_footprint">
    <turnGravityOff>false</turnGravityOff>
</gazebo>
```

Create the Physical Simulation Model

3. Add transmission to link the joints and the controller

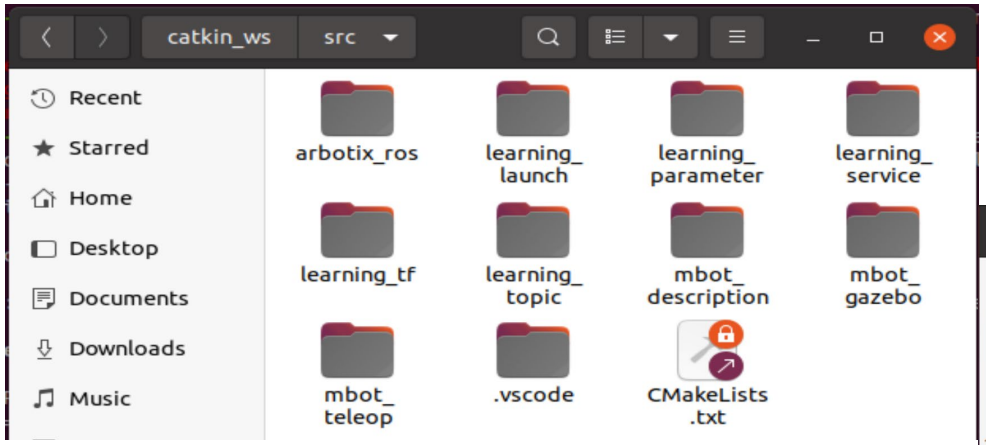
```
        <!-- Transmission is important to link the joints and the
controller -->
        <transmission name="${prefix}_wheel_joint_trans">
            <type>transmission_interface/SimpleTransmission</type>
            <joint name="${prefix}_wheel_joint" >
                <hardwareInterface>hardware_interface/-
VelocityJointInterface</hardwareInterface>
            </joint>
            <actuator name="${prefix}_wheel_joint_motor">
                <hardwareInterface>hardware_interface/-
VelocityJointInterface</hardwareInterface>
                <mechanicalReduction>1</mechanicalReduction>
            </actuator>
        </transmission>
```

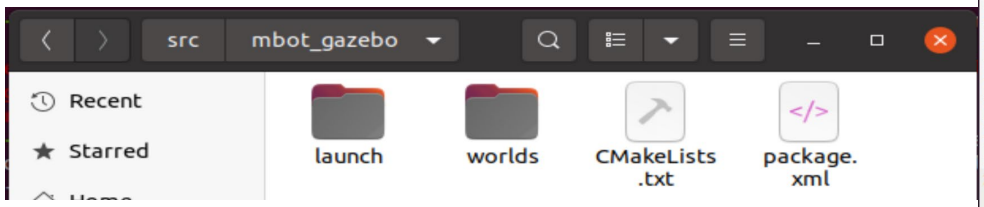
Create the Physical Simulation Model

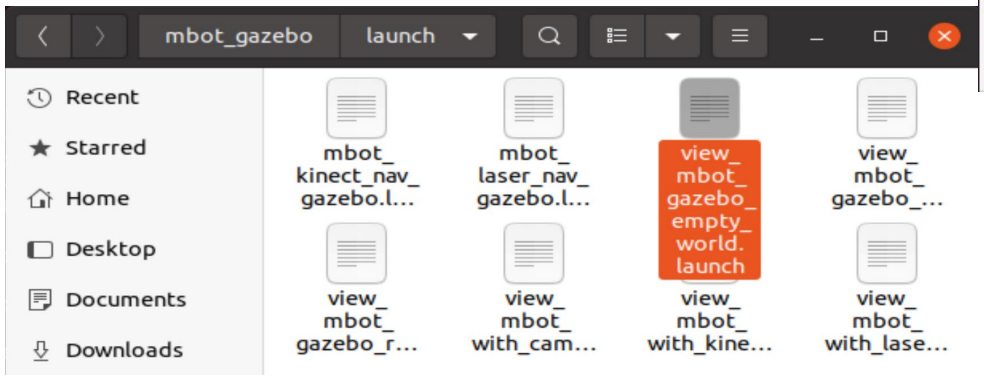
3. Add controller

```
<!-- controller -->
<gazebo>
    <plugin name="differential_drive_controller"
            filename="libgazebo_ros_diff_drive.so">
        <rosDebugLevel>Debug</rosDebugLevel>
        <publishWheelTF>true</publishWheelTF>
        <robotNamespace>/</robotNamespace>
        <publishTf>1</publishTf>
        <publishWheelJointState>true</publishWheelJointState>
        <alwaysOn>true</alwaysOn>
        <updateRate>100.0</updateRate>
        <legacyMode>true</legacyMode>
        <leftJoint>left_wheel_joint</leftJoint>
        <rightJoint>right_wheel_joint</rightJoint>
        <wheelSeparation>${wheel_joint_y*2}</wheelSeparation>
        <wheelDiameter>${2*wheel_radius}</wheelDiameter>
        <broadcastTF>1</broadcastTF>
        <wheelTorque>30</wheelTorque>
        <wheelAcceleration>1.8</wheelAcceleration>
        <commandTopic>cmd_vel</commandTopic>
        <odometryFrame>odom</odometryFrame>
        <odometryTopic>odom</odometryTopic>
        <robotBaseFrame>base_footprint</robotBaseFrame>
    </plugin>
</gazebo>
```

Create launch

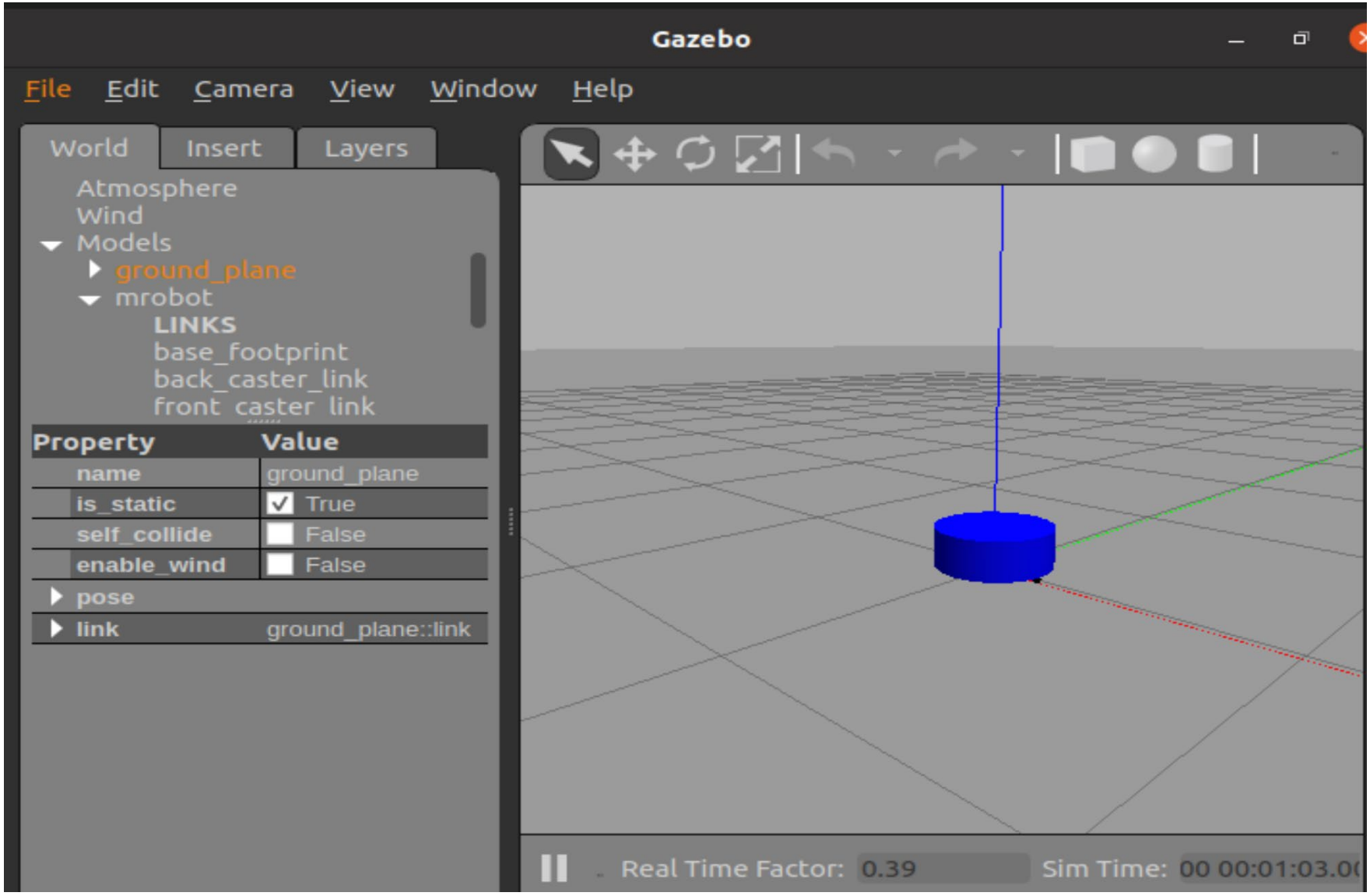




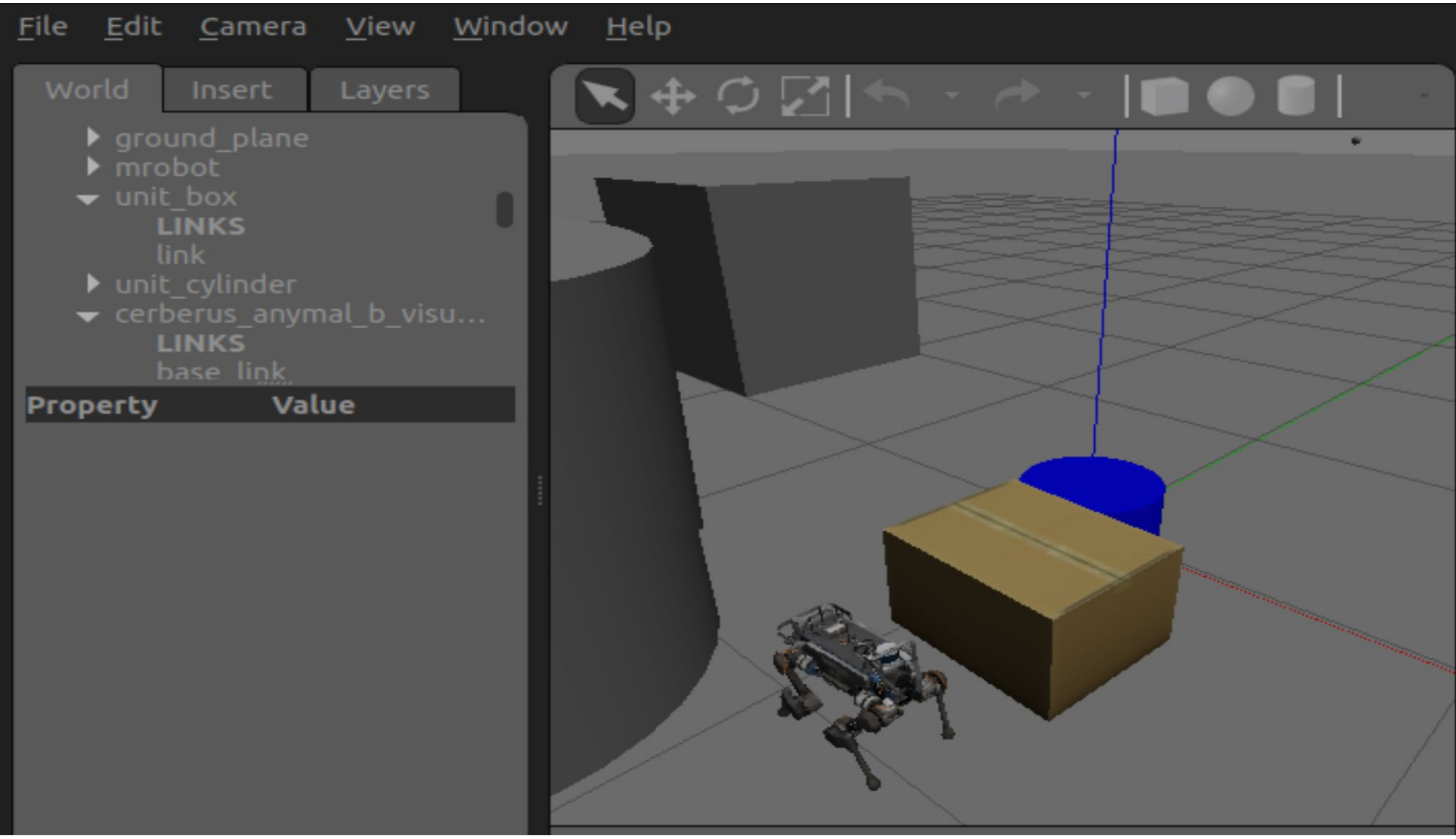


roslaunch

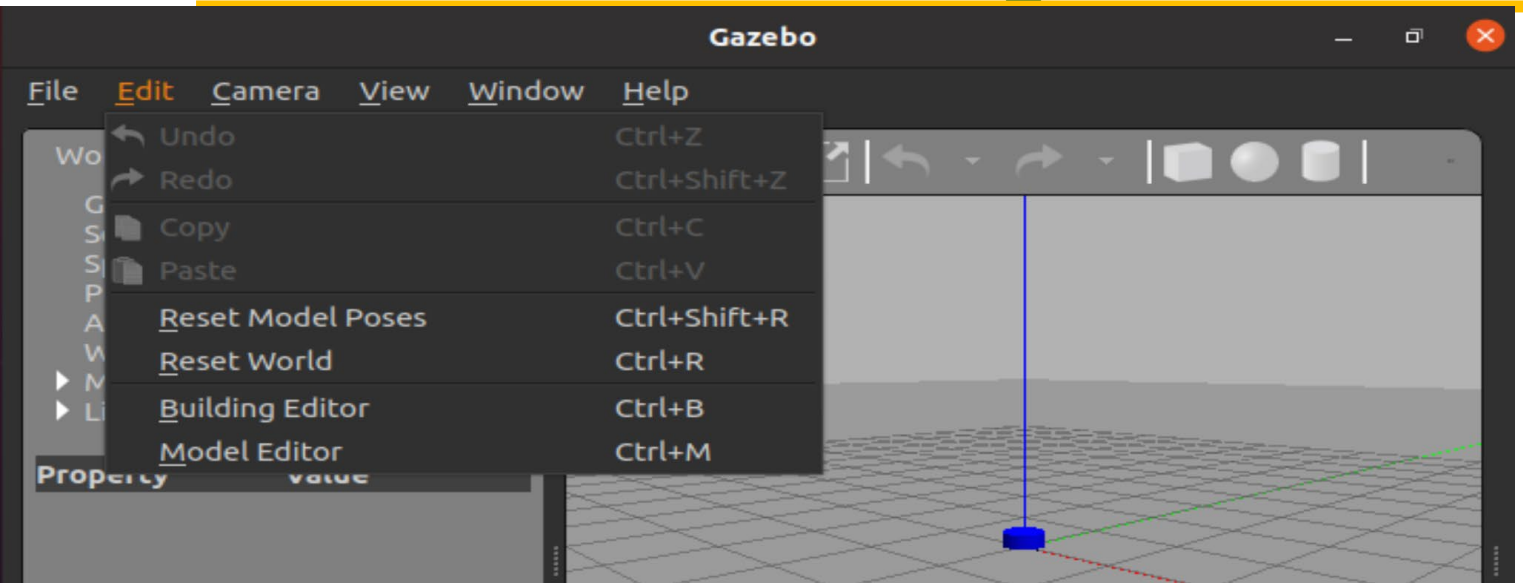
```
ab@ab-c:~/catkin_ws$ roslaunch mbot_gazebo view_mbot_gazebo_empty_world.launch
```

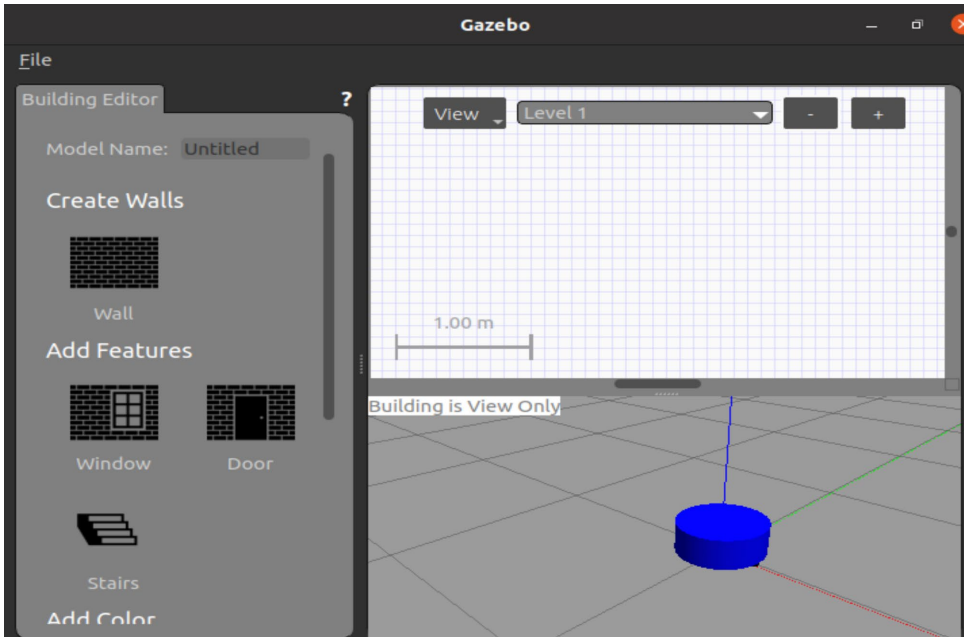


Create Simulation Environment



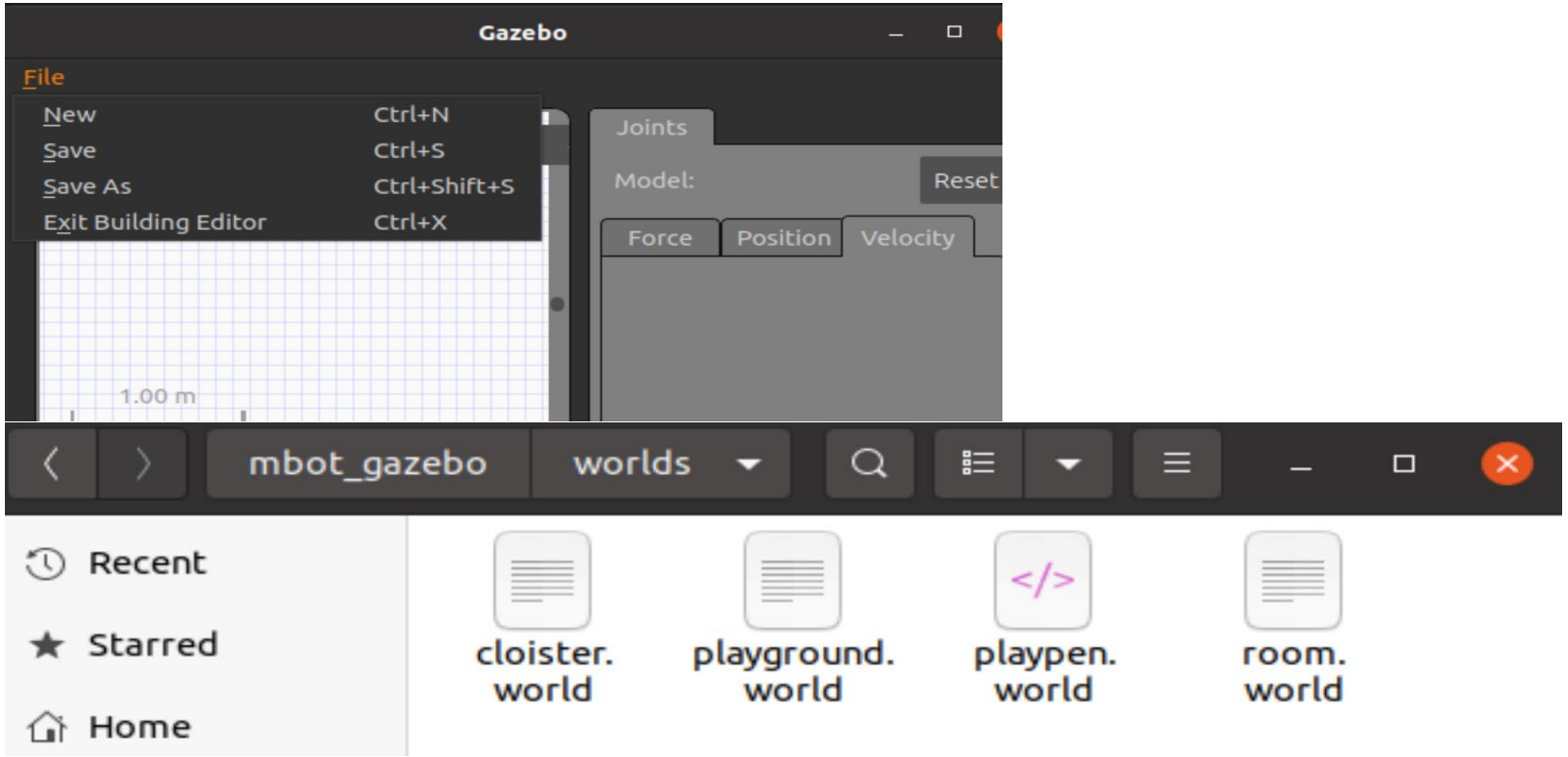
Create Simulation Environment
Building editor





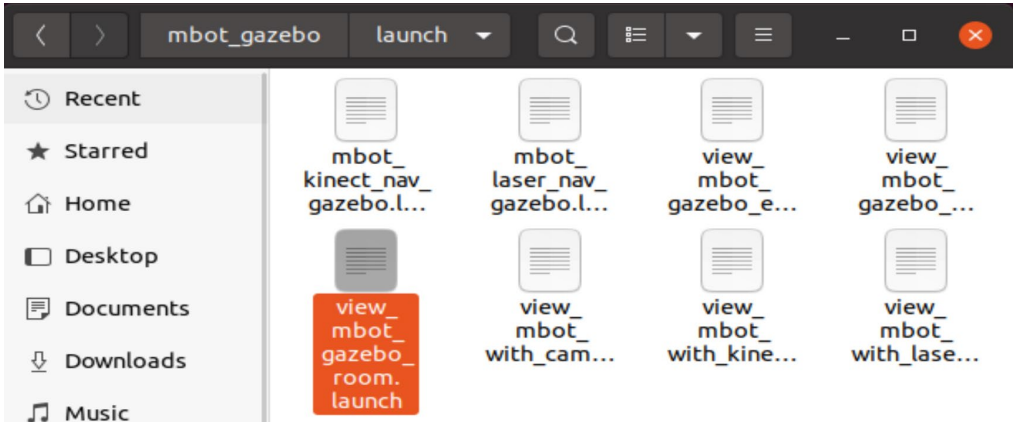
Create Simulation Environment Building editor

Save: room.world



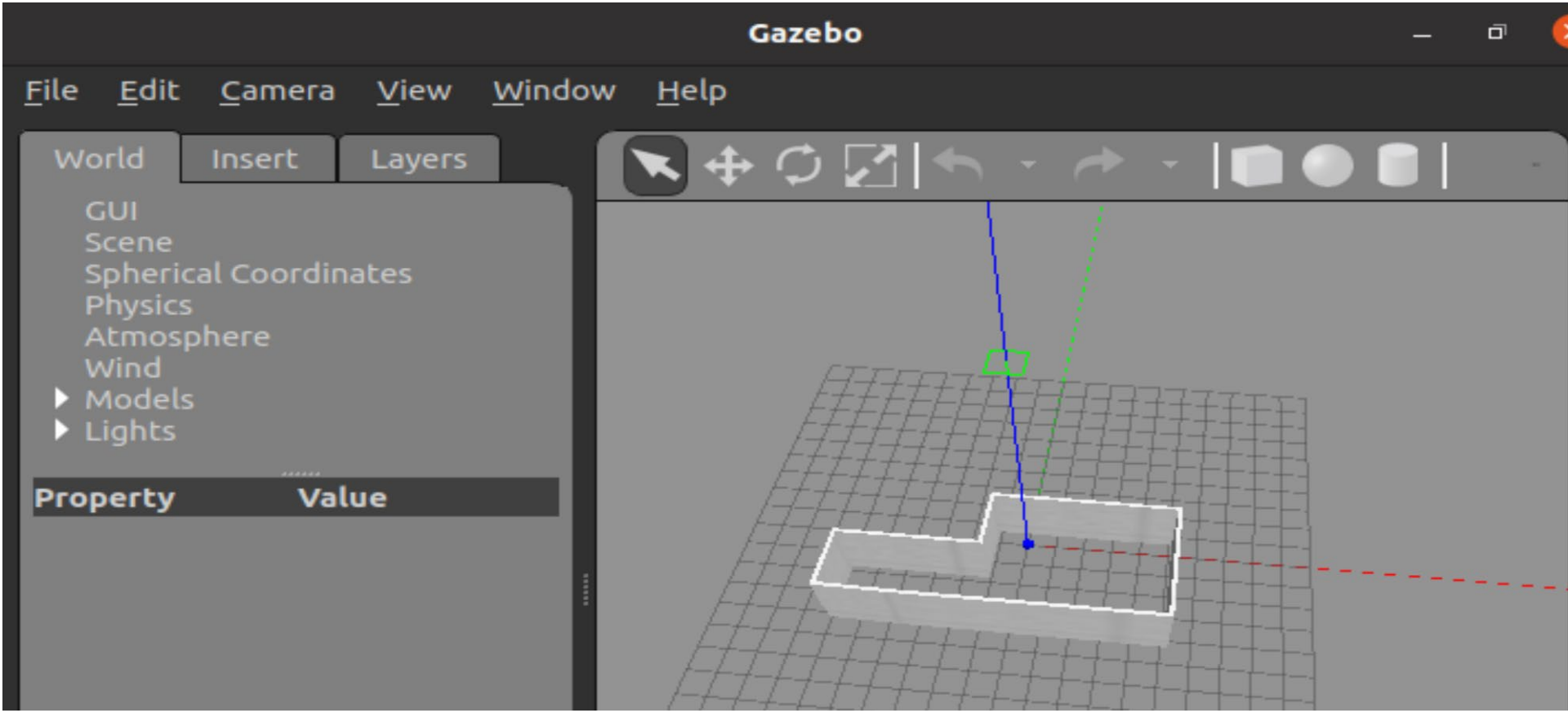
Create Simulation Environment Building editor

Save: room.world



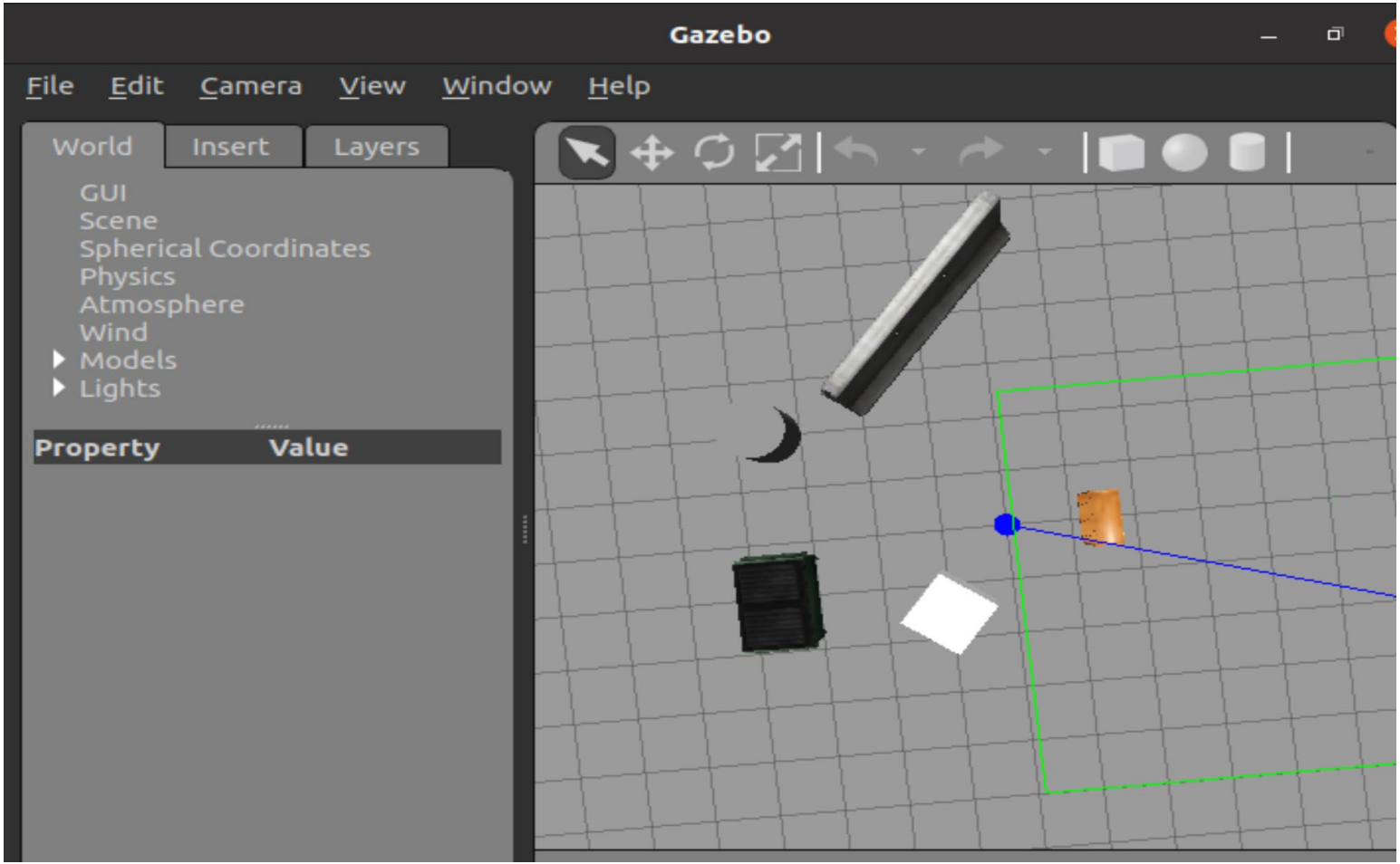
Create Simulation Environment
Building editor

```
ab@ab-c:~/catkin_ws$ catkin_make
ab@ab-c:~/catkin_ws$ source devel/setup.bash
ab@ab-c:~/catkin_ws/src/mbot_gazebo/launch$ roslaunch mbot_gazebo view_mbot_gaz
ebo_room.launch
```

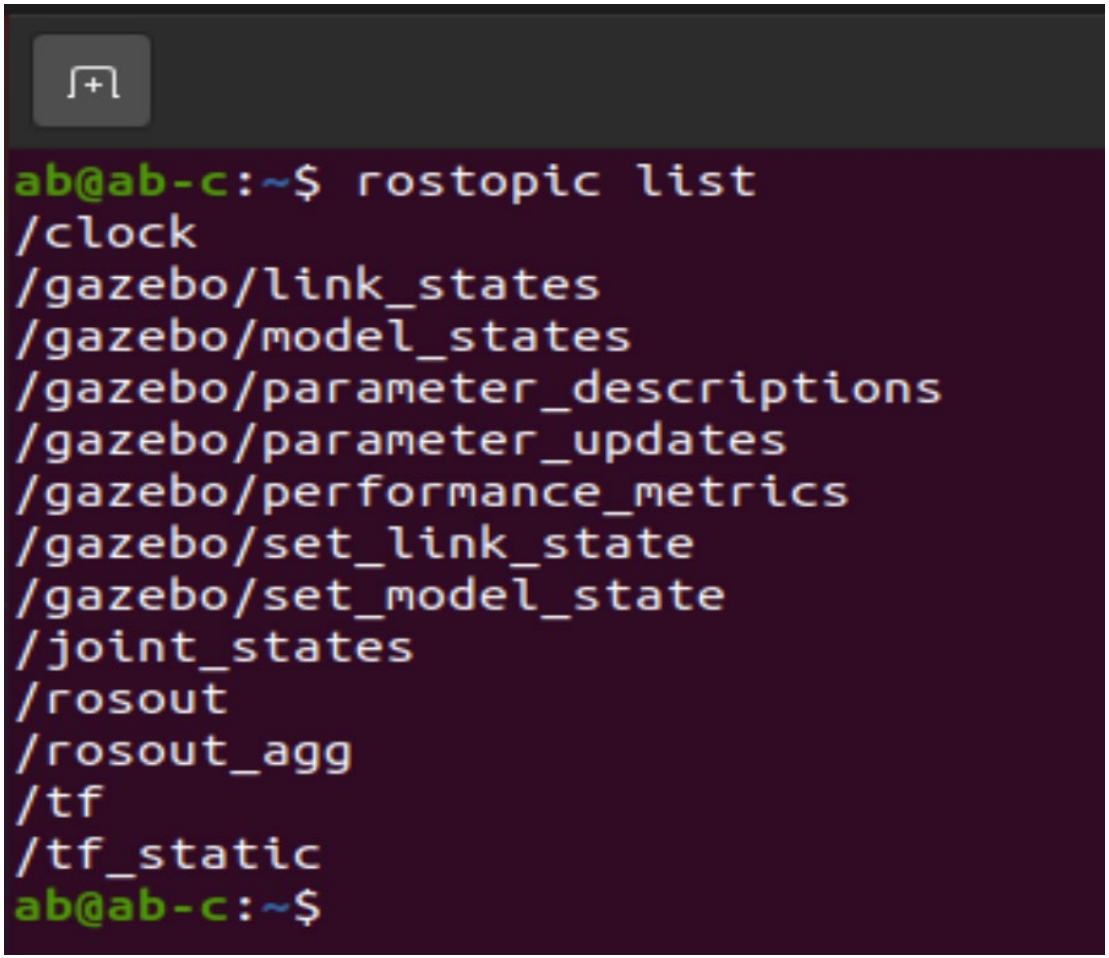


Simulation

```
ab@ab-c:~/catkin_ws$ catkin_make
ab@ab-c:~/catkin_ws$ source devel/setup.bash
ab@ab-c:~/catkin_ws/src/mbot_gazebo/launch$ roslaunch mbot_gazebo view_mbot_gaz
ebo_play_ground.launch
```



Simulation


ab@ab-c:~$ rostopic list
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/joint_states
/rosout
/rosout_agg
/tf
/tf_static
ab@ab-c:~$

Simulation


ab@ab-c:~$ roslaunch mbot_teleop mbot_teleop.launch
/home/ab/catkin_ws/src/mbot_t...
mbot_teleop (mbot_teleop/mbot_teleop.py)
ROS_MASTER_URI=http://localhost:11311
process[mbot_teleop-1]: started with pid [33708]
Control mbot!
Moving around:
u i o
j k l
m , .
q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
space key, k : force stop
anything else : stop smoothly
CTRL-C to quit
currently: speed 0.2 turn 1

Reference

✓ Download link

✓ Language:

English, Chinese, Japanese, Korean

"ROS Robot Programming"

A Handbook is written by TurtleBot3 Developers

Reference

- R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza. Introduction to Autonomous Mobile Robots. MIT Press, 2nd Edition, 2011, ISBN-10: 0262015358.
- Y. Pyo, H. Cho, R. Jung, and T. Lim, ROS Robot Programming, ROBOTIS Co., Ltd., 2017, ISBN 979-11-962307-1-5
- J. O'Kane, A Gentle Introduction to ROS, CreateSpace Independent Publishing Platform, 2013, ISBN-13: 978-1492143239