

Introduction to URDF: Mobile Robot Design and Control

Install the necessary packages required

Execute in Terminal #1

```
sudo apt-get install ros-foxy-teleop-twist-keyboard  
sudo apt-get install ros-foxy-joint-state-publisher*
```

```
sudo apt install gazebo11 libgazebo11-dev  
sudo apt install ros-foxy-gazebo-ros-pkgs  
sudo apt install ros-foxy-robot-state-publisher*
```

Execute in Terminal #1

```
cd ros2_ws/src  
ros2 pkg create lab4 --build-type ament_python --dependencies rclpy  
cd src/lab4/  
mkdir urdf  
mkdir launch  
cd urdf  
touch three_wheeled_robot.urdf
```

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

```
<robot name = "three_wheeled_robot">
```

```
  <link name="base">  
    <visual>  
      <geometry>  
        <box size="0.75 0.4 0.1"/>  
      </geometry>  
      <material name="gray">  
        <color rgba=".2 .2 .2 1" />  
      </material>  
    </visual>  
  
    <inertial>  
      <mass value="1" />  
      <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01" iyz="0" izz="0.01" />  
    </inertial>  
  
    <collision>  
      <geometry>  
        <box size="0.75 0.4 0.1"/>  
      </geometry>  
    </collision>
```

```
</link>
```

```
<link name="wheel_right_link">
  <inertial>
    <mass value="2" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

  <visual>
    <geometry>
      <cylinder radius="0.15" length="0.1"/>
    </geometry>
    <material name="white">
      <color rgba="1 1 1 1"/>
    </material>
  </visual>

  <collision>
    <geometry>
      <cylinder radius="0.15" length="0.1"/>
    </geometry>
    <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
  </collision>
</link>
```

```
<joint name="wheel_right_joint" type="continuous">
  <origin xyz="0.2 0.25 0.0" rpy="1.57 0.0 0.0"/>
  <parent link="base"/>
  <child link="wheel_right_link"/>
  <axis xyz="0.0 0.0 1.0"/>
</joint>
```

```
<link name="wheel_left_link">
  <inertial>
    <mass value="2" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

  <visual>
    <geometry>
      <cylinder radius="0.15" length="0.1"/>
    </geometry>
    <material name="white">
      <color rgba="1 1 1 1"/>
    </material>
  </visual>

  <collision>
```

```

    <geometry>
      <cylinder radius="0.15" length="0.1"/>
    </geometry>
    <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
  </collision>
</link>

```

```

<joint name="wheel_left_joint" type="continuous">
  <origin xyz="0.2 -0.25 0.0" rpy="1.57 0.0 0.0"/>
  <parent link="base"/>
  <child link="wheel_left_link"/>
  <axis xyz="0.0 0.0 1.0"/>
</joint>

```

```

<link name="caster">
  <inertial>
    <mass value="1" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

```

```

  <visual>
    <geometry>
      <sphere radius=".08" />
    </geometry>
    <material name="white" />
  </visual>

```

```

  <collision>
    <origin/>
    <geometry>
      <sphere radius=".08" />
    </geometry>
  </collision>
</link>

```

```

<joint name="caster_joint" type="continuous">
  <origin xyz="-0.3 0.0 -0.07" rpy="0.0 0.0 0.0"/>
  <axis xyz="0 0 1" />
  <parent link="base"/>
  <child link="caster"/>
</joint>

```

```

<link name="camera">
  <inertial>
    <mass value="0.1" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

```

```

<visual>
  <geometry>
    <box size="0.1 0.1 0.05"/>
  </geometry>
  <material name="white">
    <color rgba="1 1 1 1"/>
  </material>
</visual>

<collision>
  <geometry>
    <box size="0.1 0.1 0.05"/>
  </geometry>
</collision>
</link>

<joint name="camera_joint" type="fixed">
  <origin xyz="-0.35 0 0.01" rpy="0 0.0 3.14"/>
  <parent link="base"/>
  <child link="camera"/>
  <axis xyz="0.0 0.0 1.0"/>
</joint>

<link name="lidar">
  <inertial>
    <mass value="0.5" />
    <inertia ixx="0.01" ixy="0.0" ixz="0"
      iyy="0.01" iyz="0" izz="0.01" />
  </inertial>

  <visual>
    <geometry>
      <cylinder radius="0.1" length="0.05"/>
    </geometry>
    <material name="white">
      <color rgba="1 1 1 1"/>
    </material>
  </visual>

  <collision>
    <geometry>
      <box size="0.1 0.1 0.1"/>
    </geometry>
  </collision>
</link>

<joint name="lidar_joint" type="fixed">
  <origin xyz="-0.285 0 0.075" rpy="0 0.0 1.57"/>
  <parent link="base"/>
  <child link="lidar"/>
  <axis xyz="0.0 0.0 1.0"/>

```

```
</joint>
```

```
<!--http://wiki.ros.org/simulator_gazebo/Tutorials/ListOfMaterials-->
```

```
<gazebo reference="base">
```

```
  <material>Gazebo/WhiteGlow</material>
```

```
</gazebo>
```

```
<gazebo reference="wheel_left_link">
```

```
  <material>Gazebo/SkyBlue</material>
```

```
</gazebo>
```

```
<gazebo reference="wheel_right_link">
```

```
  <material>Gazebo/SkyBlue </material>
```

```
</gazebo>
```

```
<gazebo reference="caster">
```

```
  <material>Gazebo/Grey</material>
```

```
</gazebo>
```

```
<gazebo reference="lidar">
```

```
  <material>Gazebo/Blue</material>
```

```
</gazebo>
```

```
<gazebo reference="camera">
```

```
  <material>Gazebo/Red</material>
```

```
</gazebo>
```

```
<!-- differential robot-->
```

```
<gazebo>
```

```
  <plugin filename="libgazebo_ros_diff_drive.so" name="gazebo_base_controller">
```

```
    <odometry_frame>odom</odometry_frame>
```

```
    <commandTopic>cmd_vel</commandTopic>
```

```
    <publish_odom>true</publish_odom>
```

```
    <publish_odom_tf>true</publish_odom_tf>
```

```
    <update_rate>15.0</update_rate>
```

```
    <left_joint>wheel_left_joint</left_joint>
```

```
    <right_joint>wheel_right_joint</right_joint>
```

```
    <wheel_separation>0.5</wheel_separation>
```

```
    <wheel_diameter>0.3</wheel_diameter>
```

```
    <max_wheel_acceleration>0.7</max_wheel_acceleration>
```

```
    <max_wheel_torque>8</max_wheel_torque>
```

```
    <robotBaseFrame>base</robotBaseFrame>
```

```
  </plugin>
```

```
</gazebo>
```

```
<!-- camera plugin-->
```

```
<gazebo reference="camera">
```

```
  <sensor type="camera" name="camera1">
```

```
    <visualize>true</visualize>
```

```
    <update_rate>30.0</update_rate>
```

```
    <camera name="head">
```

```
      <horizontal_fov>1.3962634</horizontal_fov>
```

```
      <image>
```

```

    <width>800</width>
    <height>800</height>
    <format>R8G8B8</format>
  </image>
  <clip>
    <near>0.02</near>
    <far>300</far>
  </clip>
</camera>
<plugin name="camera_controller" filename="libgazebo_ros_camera.so">
  <alwaysOn>true</alwaysOn>
  <updateRate>60.0</updateRate>
  <cameraName>/camera1</cameraName>
  <imageTopicName>image_raw</imageTopicName>
  <cameraInfoTopicName>info_camera</cameraInfoTopicName>
  <frameName>camera</frameName>
  <hackBaseline>0.07</hackBaseline>
</plugin>
</sensor>
</gazebo>

```

```

<!--lidar plugin-->
<gazebo reference="lidar">
  <sensor name="lidar" type="ray">
    <visualize>true</visualize>
    <update_rate>12.0</update_rate>
    <plugin filename="libgazebo_ros_ray_sensor.so" name="gazebo_lidar">
      <output_type>sensor_msgs/LaserScan</output_type>
      <frame_name>lidar</frame_name>
    </plugin>
    <ray>
      <scan>
        <horizontal>
          <samples>360</samples>
          <resolution>1</resolution>
          <min_angle>0.00</min_angle>
          <max_angle>3.14</max_angle>
        </horizontal>
      </scan>
      <range>
        <min>0.120</min>
        <max>3.5</max>
        <resolution>0.015</resolution>
      </range>
    </ray>
  </sensor>
</gazebo>

```

```

</robot>

```

```

cd ..
cd launch

```

touch gazebo.launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node
from launch.actions import ExecuteProcess
def generate_launch_description():
    urdf = '/home/asha/ros2_ws/src/lab4/urdf/three_wheeled_robot.urdf'
    return LaunchDescription([

        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            arguments=[urdf]),
        Node(
            package='joint_state_publisher',
            executable='joint_state_publisher',
            name='joint_state_publisher',
            arguments=[urdf]),
        # Gazebo related stuff required to launch the robot in simulation
        ExecuteProcess(
            cmd=['gazebo', '--verbose', '-s', 'libgazebo_ros_factory.so'],
            output='screen'),
        Node(
            package='gazebo_ros',
            executable='spawn_entity.py',
            name='urdf_spawner',
            output='screen',
            arguments=["-topic", "/robot_description", "-entity", "lab4"])
    ])
```

touch rviz.launch.py

```
from launch import LaunchDescription
from launch_ros.actions import Node

def generate_launch_description():
    urdf = '/home/asha/ros2_ws/src/lab4/urdf/three_wheeled_robot.urdf'
    # rviz_config_file=os.path.join(package_dir,'config.rviz')

    return LaunchDescription([
        Node(
            package='robot_state_publisher',
            executable='robot_state_publisher',
            name='robot_state_publisher',
            output='screen',
            arguments=[urdf]),
        Node(
            package='joint_state_publisher_gui',
            executable='joint_state_publisher_gui',
```

```

        name='joint_state_publisher_gui',
        arguments=[urdf]),

    Node(
        package='rviz2',
        executable='rviz2',
        name='rviz2',
        # arguments=['-d',rviz_config_file],
        output='screen'),

])

```

Edit the setup.py file as

```

from setuptools import setup
import os
from glob import glob
package_name = 'lab4 '

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    data_files=[
        ('share/ament_index/resource_index/packages',
         ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('urdf/*')),
        (os.path.join('share', package_name), glob('launch/*'))
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='asha',
    maintainer_email='asha@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        ],
    },
)

```

Execute in Terminal #1

```
colcon build --packages-select lab4
ros2 launch lab4 rviz.launch.py
```

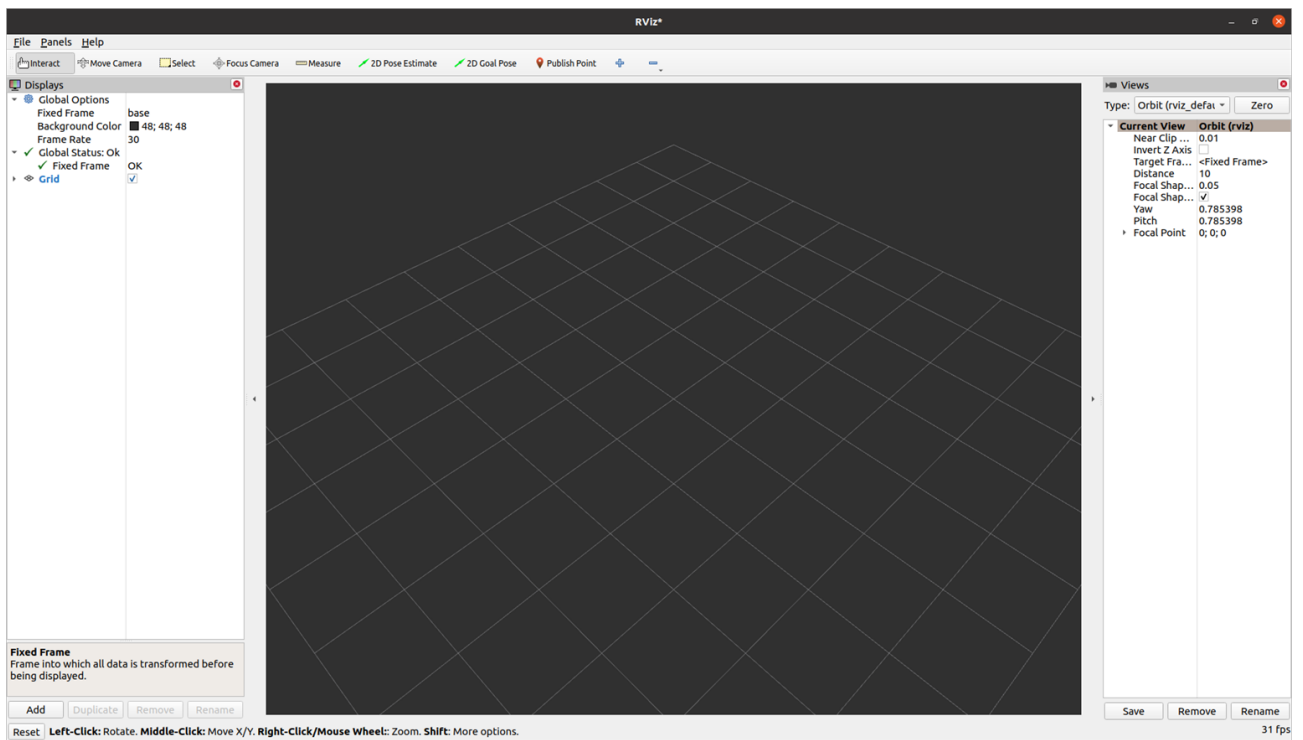
Execute in Terminal #2

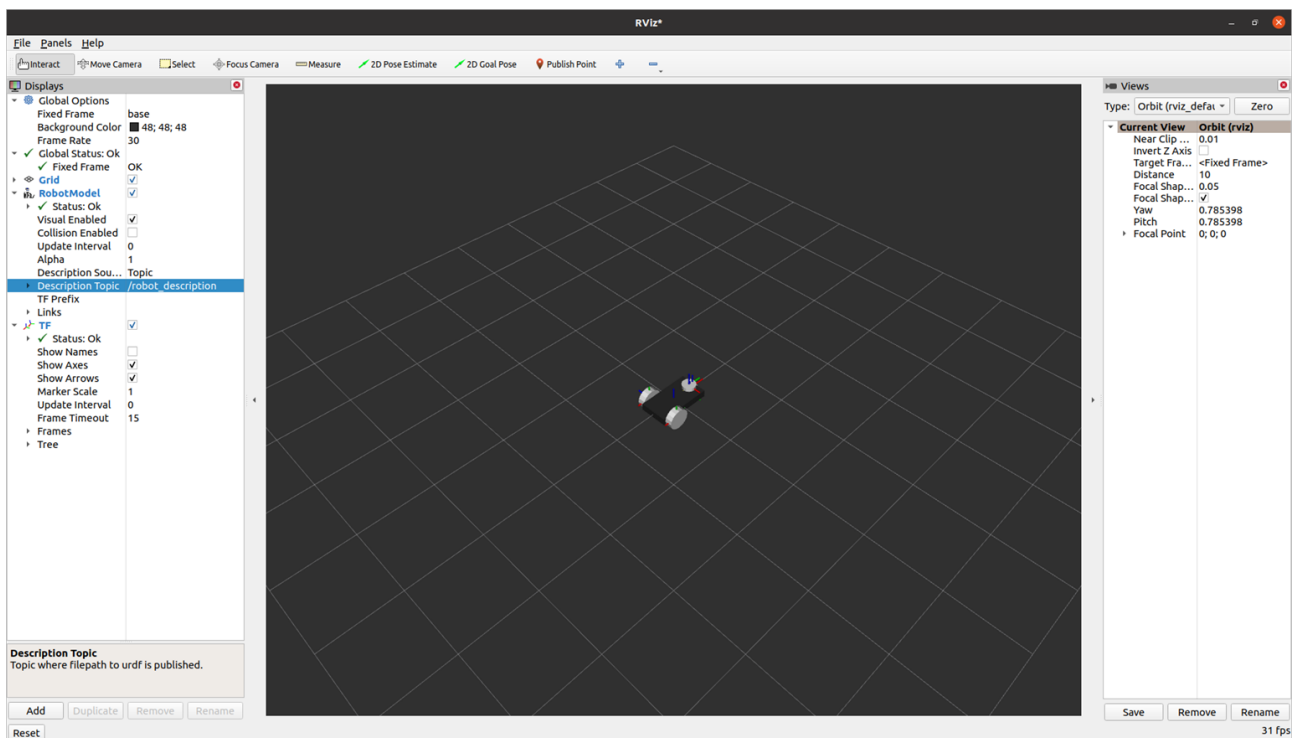
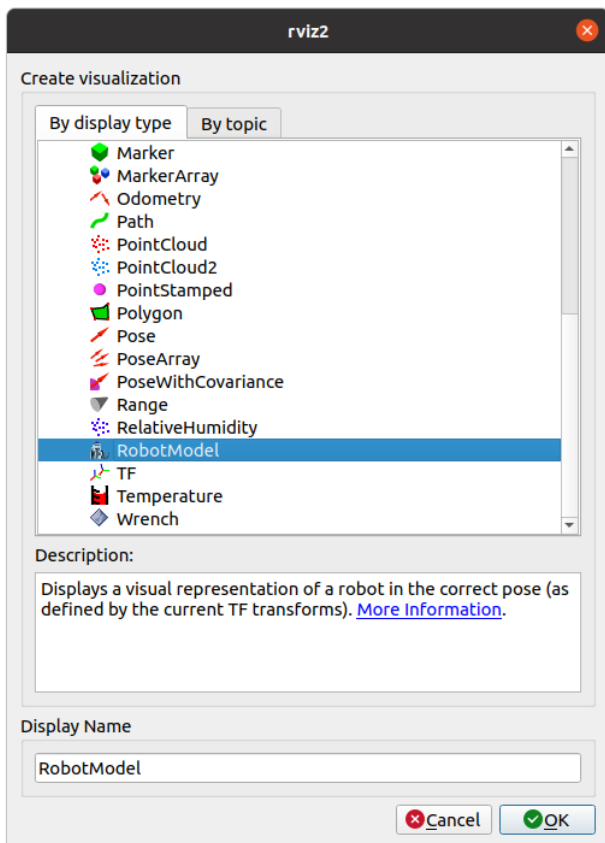
```
killall gzserver
ros2 launch lab4 gazebo.launch.py
```

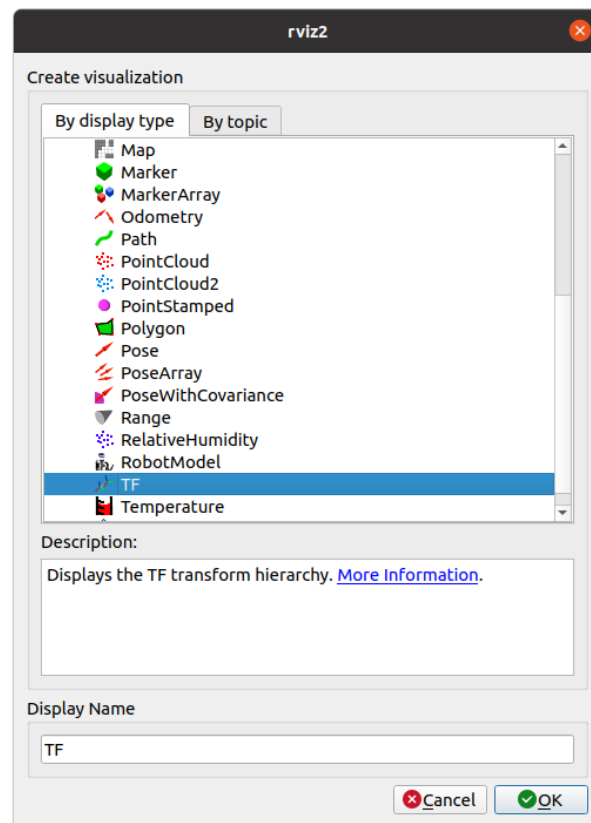
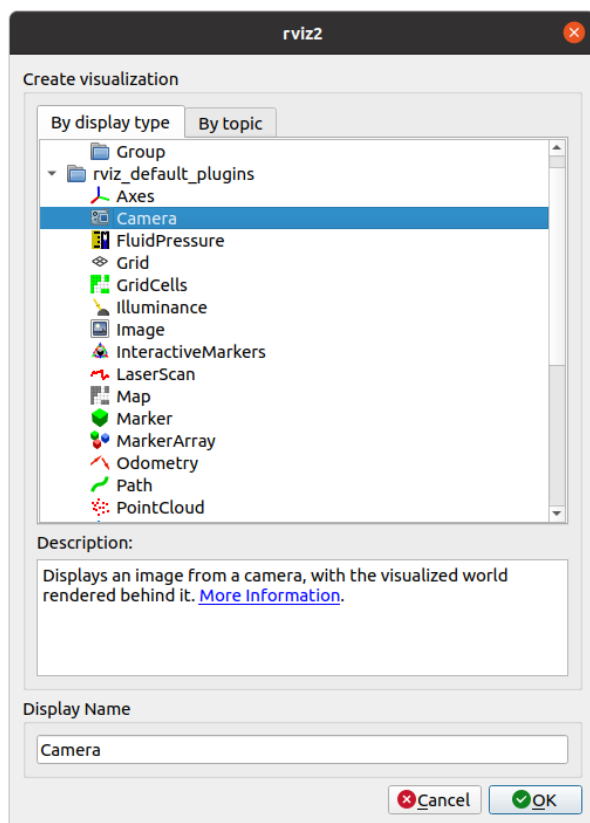
Execute in Terminal #3

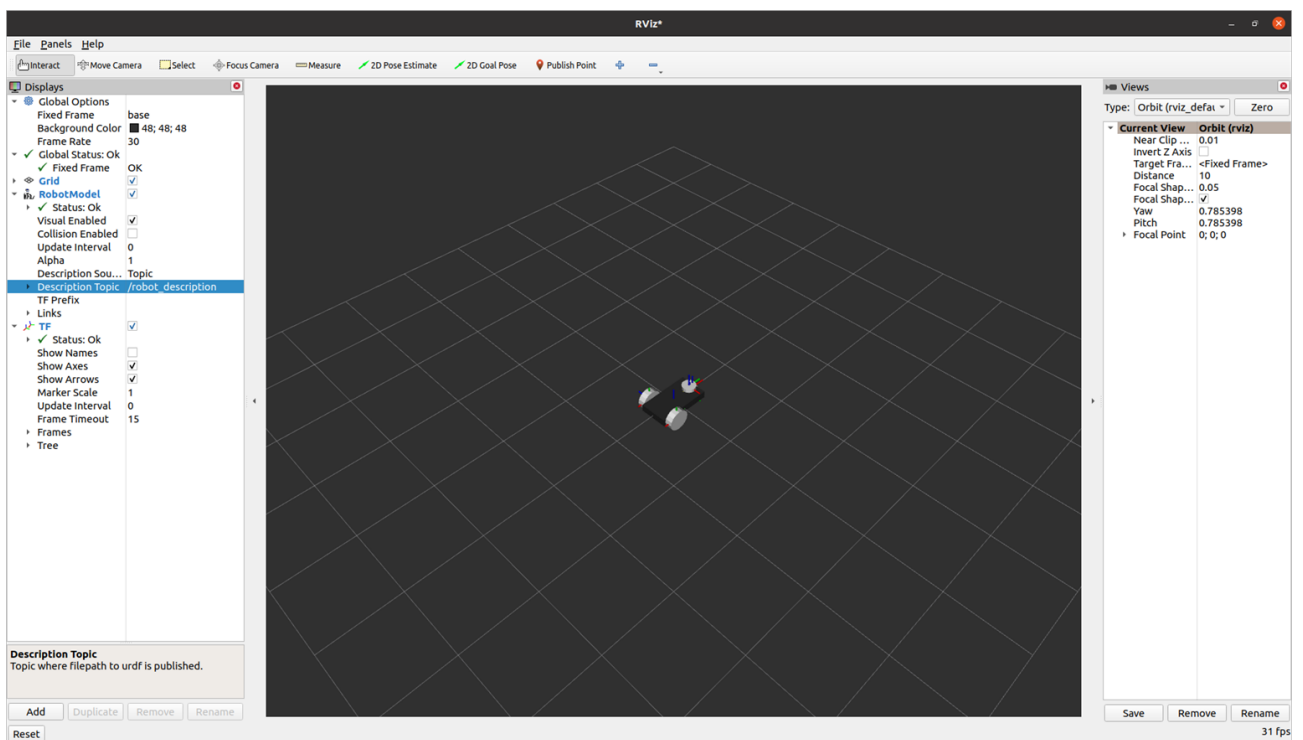
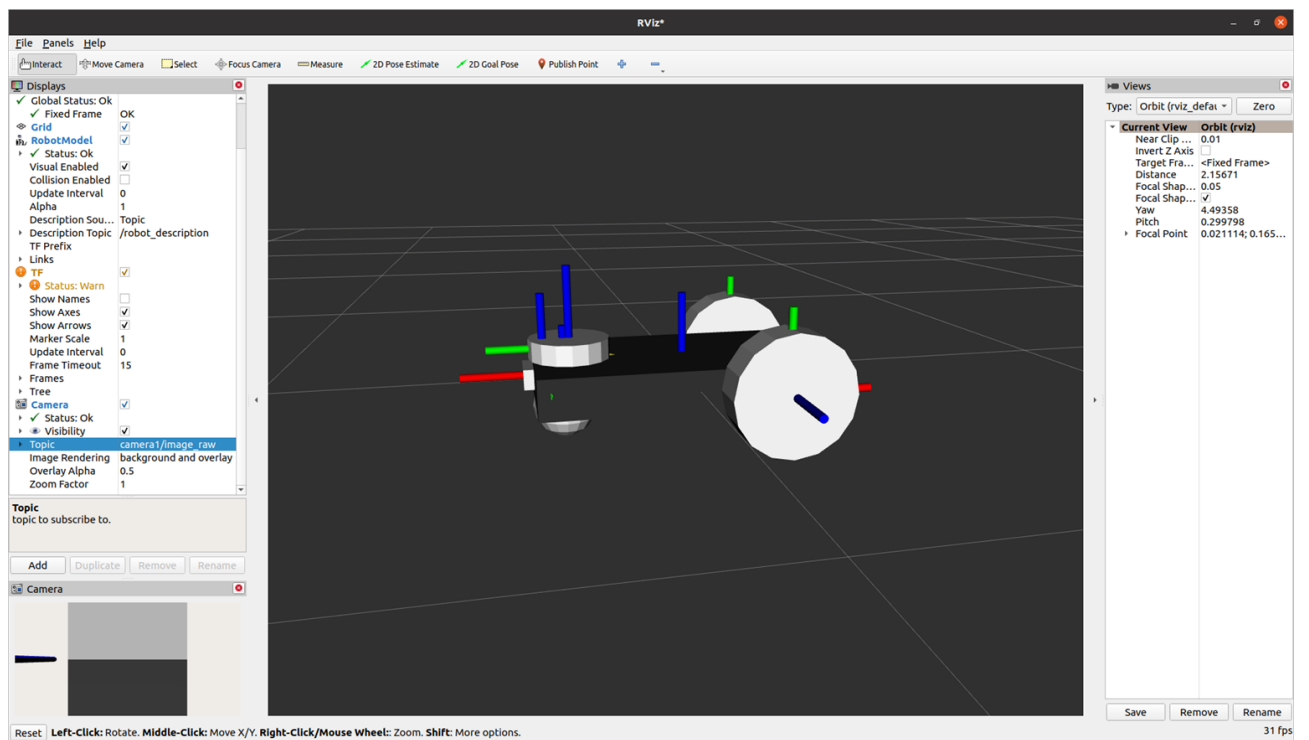
```
ros2 run teleop_twist_keyboard teleop_twist_keyboard
```

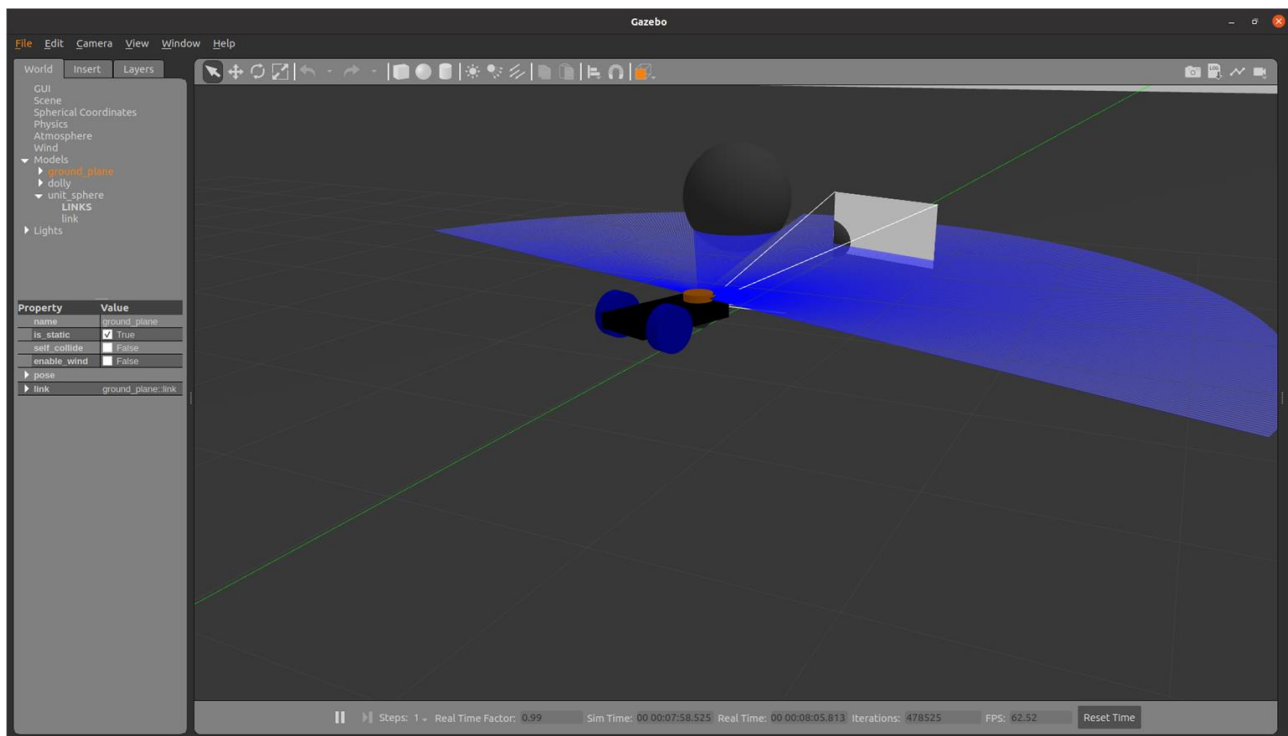
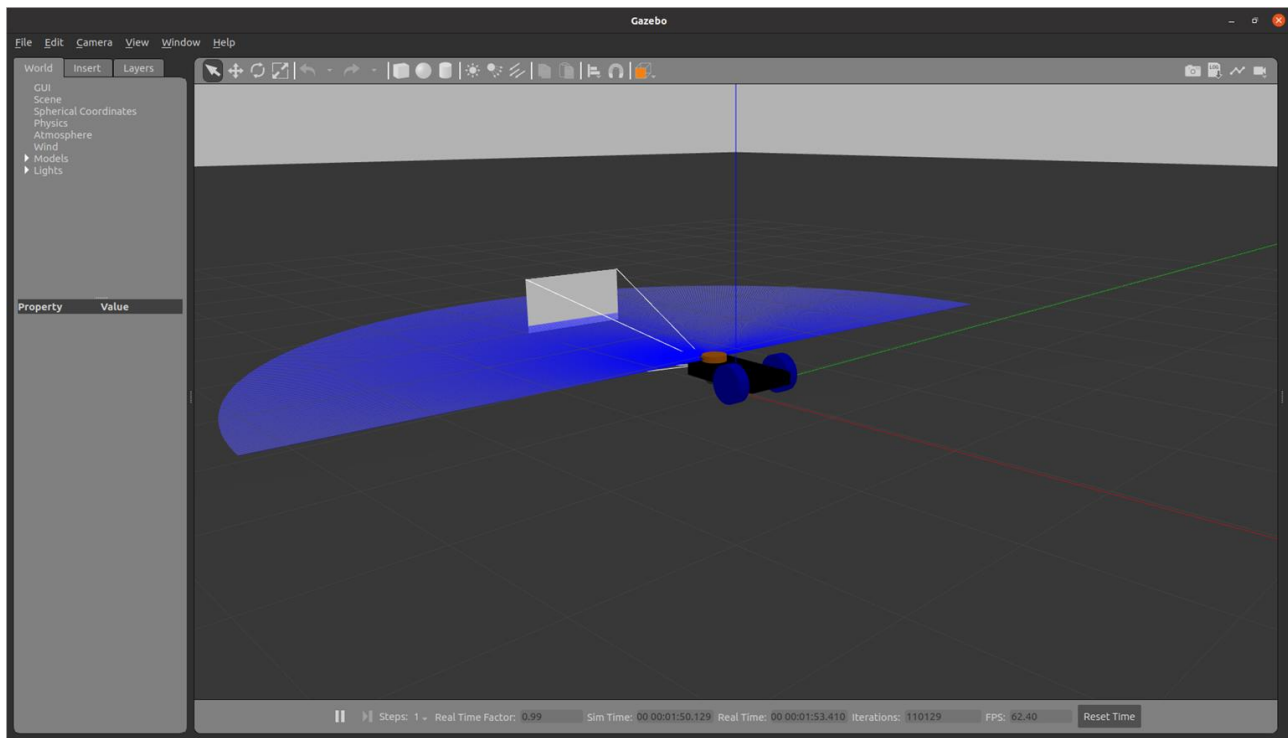

Do these changes in the rviz window:











```
sudo apt-get install python3-pip
pip3 install transforms3d
```

Edit move_robot.py

```
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist
from nav_msgs.msg import Odometry
import transforms3d
import math

class GotoGoalNode(Node):
    def __init__(self):
        super().__init__("move_robot")
        self.target_x = 2
        self.target_y = 2
        self.publisher = self.create_publisher(Twist, "cmd_vel", 10)
        self.subscriber = self.create_subscription(Odometry, "odom", self.control_loop, 10)

    def control_loop(self, msg):

        dist_x = self.target_x - msg.pose.pose.position.x
        dist_y = self.target_y - msg.pose.pose.position.y
        print('current position: {}'.format(msg.pose.pose.position.x,msg.pose.pose.position.y))
        distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)
        print('distance : {}'.format(round(distance, 3)))

        goal_theta = math.atan2(dist_y, dist_x)
        quat = msg.pose.pose.orientation
        roll, pitch, yaw = transforms3d.euler.quat2euler([quat.w,quat.x,quat.y,quat.z])
        diff = math.pi - round(yaw, 2) + round(goal_theta, 2)
        print('yaw: {}'.format(round(yaw, 2)))
        print('target angle: {}'.format(round(goal_theta, 2)))

        if diff > math.pi:
            diff -= 2*math.pi
        elif diff < -math.pi:
            diff += 2*math.pi
        print('orientation : {}'.format(round(diff, 2)))

        vel = Twist()

        if abs(diff) > 0.2:
            vel.linear.x = 0.0
            vel.angular.z = 0.4*round(diff, 2)

        else:
```

```

        if abs(distance) > 0.2:
            vel.linear.x = 0.3*round(distance, 3)
            vel.angular.z = 0.0

        else:
            vel.linear.x = 0.0
            vel.angular.z = 0.0

    print('speed : {}'.format(vel))
    self.publisher.publish(vel)

```

```

def main(args=None):
    rclpy.init(args=args)
    node = GotoGoalNode()
    rclpy.spin(node)
    rclpy.shutdown()

```

```

if __name__ == "__main__":
    main()

```

Execute in Terminal #1

```

colcon build --packages-select lab4
ros2 launch lab4 rviz.launch.py

```

Execute in Terminal #2

```

killall gzserver
ros2 launch lab4 gazebo.launch.py

```

Execute in Terminal #3

```

ros2 run lab4 controller

```

Edit setup.py

```

entry_points={
    'console_scripts': [
        'controller = lab4.move_robot:main'
    ],
}

```

Exercise 1: Write a python code to move the robot to along the wall using the laser scan data.

Exercise 2: Modify the URDF code to change to 4 wheeled robot and run the program.