ROS Lab 4

Vipul Dinesh, 220929024, MTE-A-09

General:

- **URDF (Unified Robot Description Format):**
  A XML format used in ROS2 to describe the physical configuration of robots, including the structure, joints, and sensors. It helps simulate and visualize the robot in different tools.
- **Gazebo:**
  A simulation tool that integrates with ROS2 to provide a realistic environment where robots can be tested and developed. It simulates physics, sensors, and interactions with the environment.
- **RViz:**
  A visualization tool in ROS2 used to visualize the robot's state, sensor data, and environment in real-time. It helps developers understand what the robot perceives and how it is acting.

1. Create my_sim package
- `cd ~/ros2_ws/src`
- `ros2 pkg create --build-type ament_python my_sim --dependencies rclpy`

2. Make urdf and launch sub-folders in my_sim package folder
- `mkdir ~/ros2_ws/src/my_sim/urdf`
- `mkdir ~/ros2_ws/src/my_sim/launch`

3. Create a urdf file called `three_wheeled_robot.urdf` in the urdf directory just created and fill content as follows

`three_wheeled_robot.urdf`

```
<?xml version="1.0" ?>
<!-- Define the robot model and name it "three_wheeled_robot" -
->
<robot name="three_wheeled_robot">

   <!-- Define the base link of the robot -->
   <link name="base">
       <!-- Visual properties of the base link -->
       <visual>
           <geometry>
               <!-- The base is represented as a box with the
specified dimensions -->
               <box size="0.75 0.4 0.1"/>
           </geometry>
           <material name="gray">
```

```
                    <!-- Set the color of the base to gray -->
                    <color rgba=".2 .2 .2 1" />
                </material>
            </visual>
            <!-- Inertial properties of the base link -->
            <inertial>
                <!-- Mass of the base link -->
                <mass value="1" />
                <!-- Inertia tensor of the base link -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>


            <!-- Collision properties of the base link -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <box size="0.75 0.4 0.1"/>
                </geometry>
            </collision>
        </link>


        <!-- Define the right wheel link of the robot -->
        <link name="wheel_right_link">
            <!-- Inertial properties of the right wheel -->
            <inertial>
                <!-- Mass of the right wheel -->
                <mass value="2" />
                <!-- Inertia tensor of the right wheel -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>


            <!-- Visual properties of the right wheel -->
            <visual>
                <geometry>
                    <!-- The right wheel is represented as a
cylinder with the specified radius and length -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <material name="white">
```

```
                    <!-- Set the color of the right wheel to white -
->
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the right wheel -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
            </collision>
        </link>



        <!-- Define the joint for the right wheel -->
        <joint name="wheel_right_joint" type="continuous">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="0.2 0.25 0.0" rpy="1.57 0.0 0.0"/>
            <!-- The parent link of this joint -->
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="wheel_right_link"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define the left wheel link of the robot -->
        <link name="wheel_left_link">
            <!-- Inertial properties of the left wheel -->
            <inertial>
                <!-- Mass of the left wheel -->
                <mass value="2" />
                <!-- Inertia tensor of the left wheel -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>
            <!-- Visual properties of the left wheel -->
            <visual>
                <geometry>
```

```
                    <!-- The left wheel is represented as a cylinder
with the specified radius and length -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the left wheel to white --
>
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the left wheel -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
            </collision>
        </link>

        <!-- Define the joint for the left wheel -->
        <joint name="wheel_left_joint" type="continuous">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="0.2 -0.25 0.0" rpy="1.57 0.0 0.0"/>
            <!-- The parent link of this joint -->
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="wheel_left_link"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define the caster link of the robot -->
        <link name="caster">
            <!-- Inertial properties of the caster -->
            <inertial>
                <!-- Mass of the caster -->
                <mass value="1" />
                <!-- Inertia tensor of the caster -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>
            <!-- Visual properties of the caster -->
```

```
            <visual>
                <geometry>
                    <!-- The caster is represented as a sphere with
the specified radius -->
                    <sphere radius=".08" />
                </geometry>
                <material name="white" />
            </visual>
            <!-- Collision properties of the caster -->
            <collision>
                <origin/>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <sphere radius=".08" />
                </geometry>
            </collision>
        </link>

        <!-- Define the joint for the caster -->
        <joint name="caster_joint" type="continuous">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="-0.3 0.0 -0.07" rpy="0.0 0.0 0.0"/>
            <axis xyz="0 0 1" />
            <!-- The parent link of this joint -->
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="caster"/>
        </joint>

        <!-- Define the camera link of the robot -->
        <link name="camera">
            <!-- Inertial properties of the camera -->
            <inertial>
                <!-- Mass of the camera -->
                <mass value="0.1" />
                <!-- Inertia tensor of the camera -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>
            <!-- Visual properties of the camera -->
            <visual>
                <geometry>
```

```
                    <!-- The camera is represented as a box with the
specified dimensions -->
                    <box size="0.1 0.1 0.05"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the camera to white -->
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the camera -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <box size="0.1 0.1 0.05"/>
                </geometry>
            </collision>
        </link>

        <!-- Define the joint for the camera -->
        <joint name="camera_joint" type="fixed">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="-0.35 0 0.01" rpy="0 0.0 3.14"/>
            <!-- The parent link of this joint -->
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="camera"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define the lidar link of the robot -->
        <link name="lidar">
            <!-- Inertial properties of the lidar -->
            <inertial>
                <!-- Mass of the lidar -->
                <mass value="0.5" />
                <!-- Inertia tensor of the lidar -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>
            <!-- Visual properties of the lidar -->
            <visual>
                <geometry>
```

```
                    <!-- The lidar is represented as a cylinder with
the specified radius and length -->
                    <cylinder radius="0.1" length="0.05"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the lidar to white -->
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the lidar -->
            <collision>
                <geometry>
                    <!-- The collision shape is a box with the
specified dimensions -->
                    <box size="0.1 0.1 0.1"/>
                </geometry>
            </collision>
        </link>

        <!-- Define the joint for the lidar -->
        <joint name="lidar_joint" type="fixed">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="-0.285 0 0.075" rpy="0 0.0 1.57"/>
            <!-- The parent link of this joint -->
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="lidar"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define Gazebo-specific properties for the base link -->
        <gazebo reference="base">
            <!-- Set the material of the base link in Gazebo -->
            <material>Gazebo/WhiteGlow</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the left wheel
link -->
        <gazebo reference="wheel_left_link">
            <!-- Set the material of the left wheel link in Gazebo -
->
            <material>Gazebo/SkyBlue</material>
        </gazebo>
```

```
        <!-- Define Gazebo-specific properties for the right wheel
link -->
        <gazebo reference="wheel_right_link">
            <!-- Set the material of the right wheel link in Gazebo
-->
            <material>Gazebo/SkyBlue</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the caster -->
        <gazebo reference="caster">
            <!-- Set the material of the caster in Gazebo -->
            <material>Gazebo/Grey</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the lidar -->
        <gazebo reference="lidar">
            <!-- Set the material of the lidar in Gazebo -->
            <material>Gazebo/Blue</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the camera -->
        <gazebo reference="camera">
            <!-- Set the material of the camera in Gazebo -->
            <material>Gazebo/Red</material>
        </gazebo>

        <!-- Define the Gazebo plugin for differential drive control
-->
        <gazebo>
            <plugin filename="libgazebo_ros_diff_drive.so"
name="gazebo_base_controller">
                <!-- Odometry frame for the plugin -->
                <odometry_frame>odom</odometry_frame>
                <!-- Command topic for velocity commands -->
                <commandTopic>cmd_vel</commandTopic>
                <!-- Publish odometry data -->
                <publish_odom>true</publish_odom>
                <!-- Publish odometry transform -->
                <publish_odom_tf>true</publish_odom_tf>
                <!-- Update rate for the plugin -->
                <update_rate>15.0</update_rate>
                <!-- Left wheel joint name -->
                <left_joint>wheel_left_joint</left_joint>
                <!-- Right wheel joint name -->
```

```
                <right_joint>wheel_right_joint</right_joint>
                <!-- Wheel separation distance -->
                <wheel_separation>0.5</wheel_separation>
                <!-- Wheel diameter -->
                <wheel_diameter>0.3</wheel_diameter>
                <!-- Maximum wheel acceleration -->
                <max_wheel_acceleration>0.7</max_wheel_acceleration>
                <!-- Maximum wheel torque -->
                <max_wheel_torque>8</max_wheel_torque>
                <!-- Base frame of the robot -->
                <robotBaseFrame>base</robotBaseFrame>
            </plugin>
        </gazebo>

        <!-- Define the Gazebo plugin for the camera -->
        <gazebo reference="camera">
            <!-- Define the camera sensor -->
            <sensor type="camera" name="camera1">
                <!-- Visualize the camera output in Gazebo -->
                <visualize>true</visualize>
                <!-- Update rate for the camera sensor -->
                <update_rate>30.0</update_rate>
                <camera name="head">
                    <!-- Horizontal field of view for the camera -->
                    <horizontal_fov>1.3962634</horizontal_fov>
                    <image>
                        <!-- Image width in pixels -->
                        <width>800</width>
                        <!-- Image height in pixels -->
                        <height>800</height>
                        <!-- Image format -->
                        <format>R8G8B8</format>
                    </image>
                    <clip>
                        <!-- Near clipping distance -->
                        <near>0.02</near>
                        <!-- Far clipping distance -->
                        <far>300</far>
                    </clip>
                </camera>
                <plugin name="camera_controller"
filename="libgazebo_ros_camera.so">
                    <!-- Always keep the camera on -->
                    <alwaysOn>true</alwaysOn>
                    <!-- Update rate for the camera controller -->
```

```
                    <updateRate>60.0</updateRate>
                    <!-- Camera name for the controller -->
                    <cameraName>/camera1</cameraName>
                    <!-- Image topic name -->
                    <imageTopicName>image_raw</imageTopicName>
                    <!-- Camera info topic name -->

<cameraInfoTopicName>info_camera</cameraInfoTopicName>
                    <!-- Frame name for the camera -->
                    <frameName>camera</frameName>
                    <!-- Baseline for stereo camera setup (not used
here) -->
                    <hackBaseline>0.07</hackBaseline>
                </plugin>
            </sensor>
        </gazebo>

        <!-- Define the Gazebo plugin for the lidar -->
        <gazebo reference="lidar">
            <!-- Define the lidar sensor -->
            <sensor name="lidar" type="ray">
                <!-- Visualize the lidar output in Gazebo -->
                <visualize>true</visualize>
                <!-- Update rate for the lidar sensor -->
                <update_rate>12.0</update_rate>
                <plugin filename="libgazebo_ros_ray_sensor.so"
name="gazebo_lidar">
                    <!-- Output type for the lidar sensor -->
                    <output_type>sensor_msgs/LaserScan</output_type>
                    <!-- Frame name for the lidar sensor -->
                    <frame_name>lidar</frame_name>
                </plugin>
                <ray>
                    <scan>
                        <horizontal>
                            <!-- Number of samples for the lidar
scan -->
                            <samples>360</samples>
                            <!-- Resolution of the lidar scan -->
                            <resolution>1</resolution>
                            <!-- Minimum angle for the lidar scan --
>
                            <min_angle>0.00</min_angle>
                            <!-- Maximum angle for the lidar scan --
>
```

```
                            <max_angle>3.14</max_angle>
                        </horizontal>
                    </scan>
                    <range>
                        <!-- Minimum range for the lidar sensor -->
                        <min>0.120</min>
                        <!-- Maximum range for the lidar sensor -->
                        <max>3.5</max>
                        <!-- Resolution of the lidar range -->
                        <resolution>0.015</resolution>
                    </range>
                </ray>
            </sensor>
        </gazebo>

    </robot>
```

4. In the launch folder, create launch files for gazebo and rviz so that they use the `three_wheeled_robot.urdf` file and also publish their status. Name them as `gazebo.launch.py` and `rviz.launch.py` respectively and fill the content as follows

`gazebo.launch.py`

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

    def generate_launch_description():
        urdf =
'/home/vipul/ros2_ws/src/my_sim/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", "my_sim"])
            ])
```

rviz.launch.py

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

      def generate_launch_description():
         urdf =
'/home/vipul/ros2_ws/src/my_sim/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'),
         ])
```

5. Install the required python package and then create an executable/node called `move_robot.py` in the directory `~/ros2_ws/src/my_sim/my_sim`

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

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()
```

6. Make changes in `setup.py` to set `move_robot.py` as an executable named `controller`, initialize the gazebo and rviz launch files, and also recognize the urdf file

```
    from setuptools import find_packages, setup
    from glob import glob
    import os

    package_name = 'my_sim'

    setup(
        name=package_name,
        version='0.0.0',
        packages=find_packages(exclude=['test']),
        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='vipul',
        maintainer_email='vipul@todo.todo',
        description='TODO: Package description',
        license='TODO: License declaration',
```

```
        tests_require=['pytest'],
        entry_points={
            'console_scripts': [
            'controller=my_sim.move_robot:main'
            ],
        },
    )
```

7. Rebuild the package and source it again

- `cd ~/ros2_ws/`
- `colcon build --packages-select my_sim`
- `source ~/ros2_ws/install/setup.zsh`

8. Launch rviz using the launch file and make the following changes

`terminal_1`

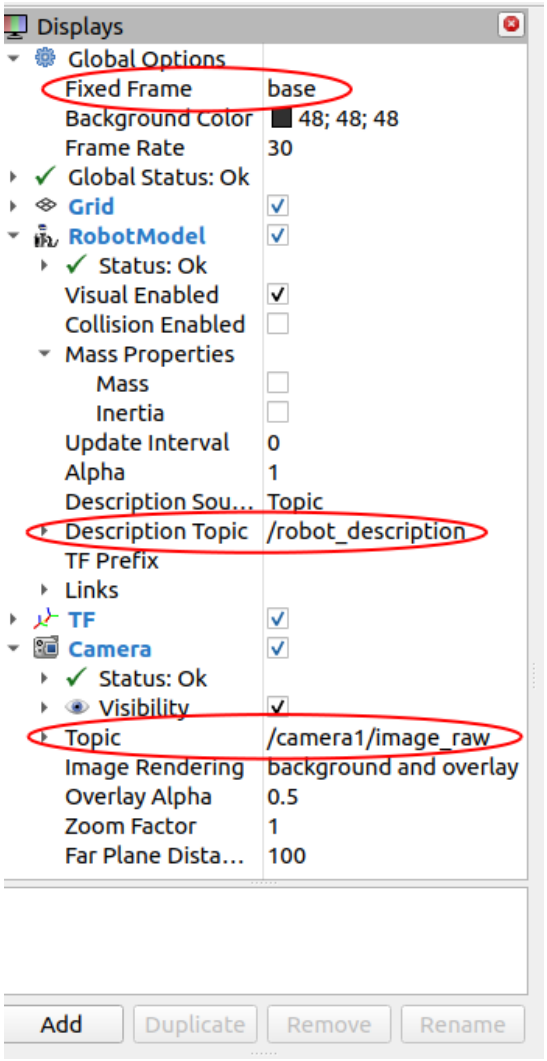
- `ros2 launch my_sim rviz.launch.py`

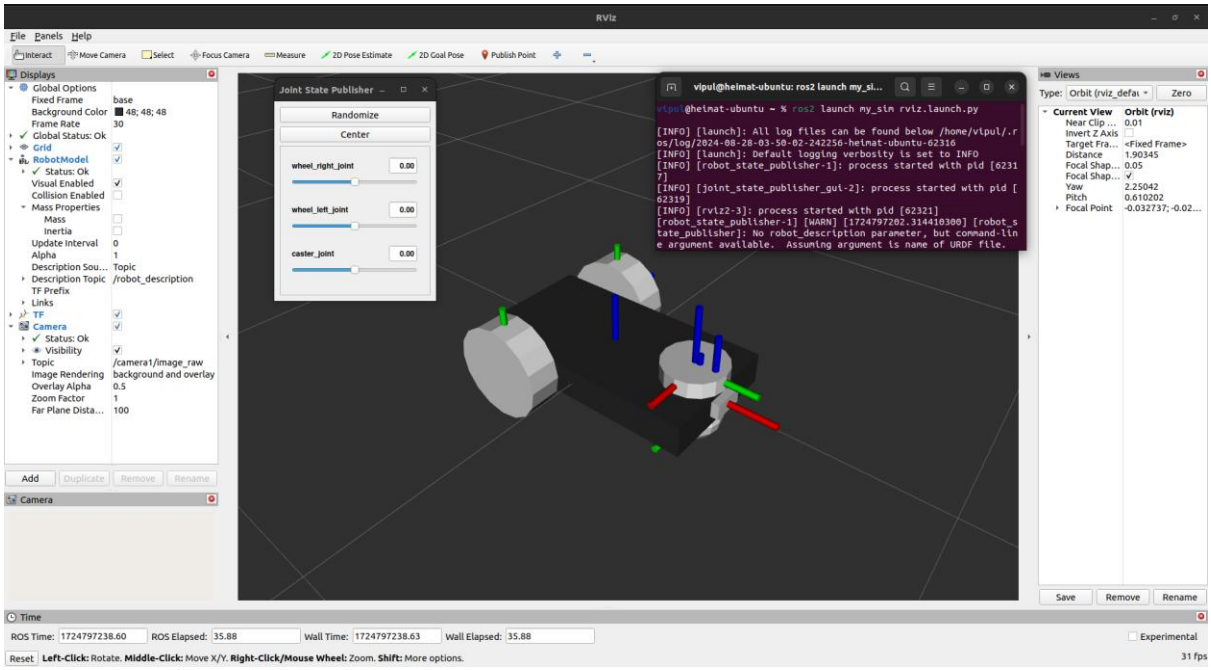
8.1. Under `Global Options`, set `Fixed Frame` as `base`

8.2. Click on `Add` at the bottom of the window and include `RobotModel`, `TF` and `Camera` (one at a time)

8.3. Under `RobotModel`, set `Description Topic` as `/robot_description`

8.4. Under `Camera`, set `Topic` as `camera1/image_raw`

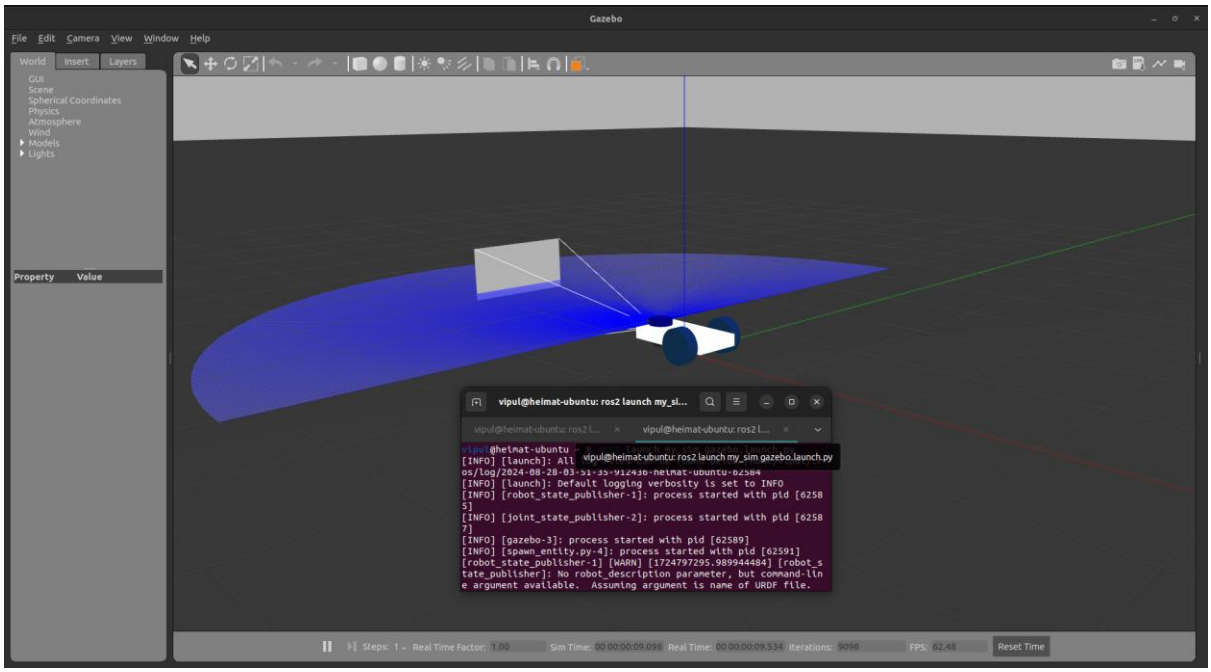




9. Launch gazebo using the launch file

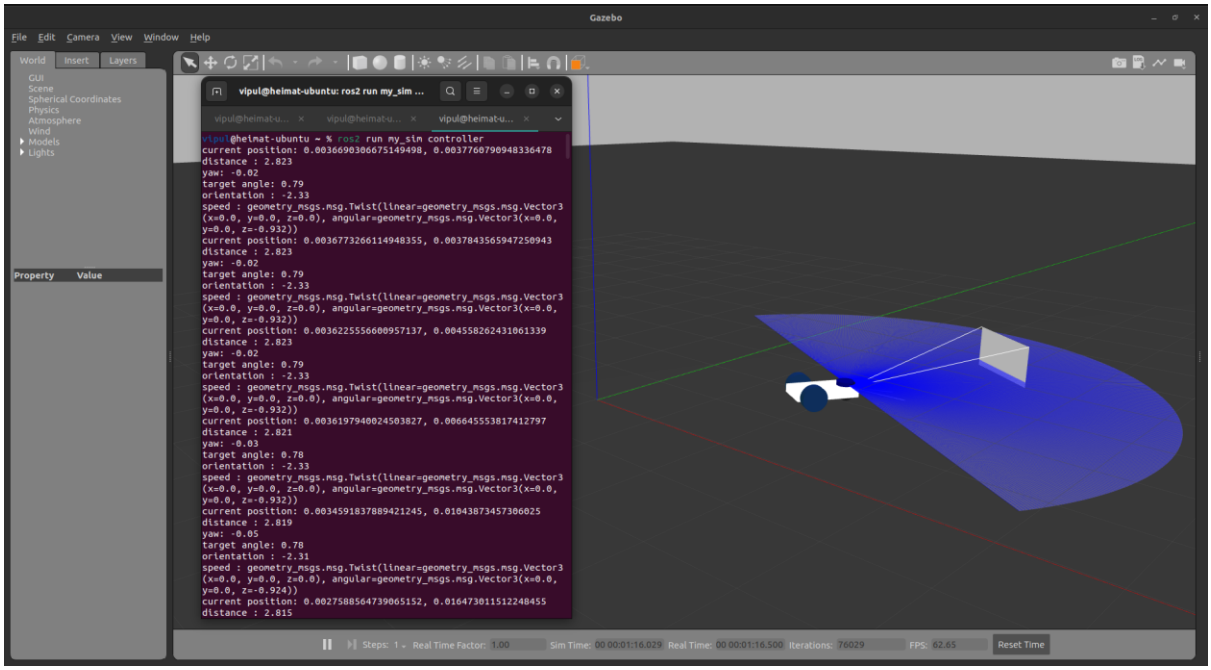
`terminal_2`

- `killall gzserver`
  Note: Used to kill any ghost/old gazebo servers, ignore if `gzserver: no process found` error is shown
- `ros2 launch my_sim gazebo.launch.py`
  Note: The URDF robot in gazebo can be teleoperated by using `ros2 run teleop_twist_keyboard teleop_twist_keyboard` in a third terminal



10. Run `my_sim:controller` in a new terminal while rviz and gazebo are open
    `terminal_3`

- `ros2 run my_sim controller`



11. Create a urdf file called `four_wheeled_robot.urdf` in the urdf directory just created and fill content as follows

four_wheeled_robot.urdf

```
<?xml version="1.0" ?>
<!-- Define the robot model and name it "three_wheeled_robot" -
->
<robot name="four_wheeled_robot">

  <!-- Define the base link of the robot -->
  <link name="base">
    <!-- Visual properties of the base link -->
    <visual>
      <geometry>
        <!-- The base is represented as a box with the
specified dimensions -->
        <box size="0.75 0.4 0.1"/>
      </geometry>
      <material name="gray">
        <!-- Set the color of the base to gray -->
        <color rgba=".2 .2 .2 1" />
      </material>
    </visual>
    <!-- Inertial properties of the base link -->
    <inertial>
      <!-- Mass of the base link -->
      <mass value="1" />
      <!-- Inertia tensor of the base link -->
      <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
    </inertial>


    <!-- Collision properties of the base link -->
    <collision>
      <geometry>
        <!-- The collision shape is the same as the
visual shape -->
        <box size="0.75 0.4 0.1"/>
      </geometry>
    </collision>
  </link>

  <!-- Define the rear right wheel link of the robot -->
  <link name="wheel_rear_right_link">
    <!-- Inertial properties of the right wheel -->
    <inertial>
      <!-- Mass of the right wheel -->
```

```
                <mass value="2" />
                <!-- Inertia tensor of the right wheel -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>


            <!-- Visual properties of the right wheel -->
            <visual>
                <geometry>
                    <!-- The right wheel is represented as a
cylinder with the specified radius and length -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the right wheel to white -
->
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the right wheel -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
            </collision>
        </link>

        <!-- Define the joint for the rear right wheel -->
        <joint name="wheel_rear_right_joint" type="continuous">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="0.2 0.25 0.0" rpy="1.57 0.0 0.0"/>
            <!-- The parent link of this joint -->
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="wheel_rear_right_link"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define the rear left wheel link of the robot -->
```

```
        <link name="wheel_rear_left_link">
            <!-- Inertial properties of the left wheel -->
            <inertial>
                <!-- Mass of the left wheel -->
                <mass value="2" />
                <!-- Inertia tensor of the left wheel -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>
            <!-- Visual properties of the left wheel -->
            <visual>
                <geometry>
                    <!-- The left wheel is represented as a cylinder
with the specified radius and length -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the left wheel to white --
>
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the left wheel -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
            </collision>
        </link>

        <!-- Define the joint for the rear left wheel -->
        <joint name="wheel_rear_left_joint" type="continuous">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="0.2 -0.25 0.0" rpy="1.57 0.0 0.0"/>
            <!-- The parent link of this joint -->
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="wheel_rear_left_link"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>
```

```
        <!-- Define the front right wheel link of the robot -->
        <link name="wheel_front_right_link">
            <!-- Inertial properties of the right wheel -->
            <inertial>
                <!-- Mass of the right wheel -->
                <mass value="2" />
                <!-- Inertia tensor of the right wheel -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>


            <!-- Visual properties of the right wheel -->
            <visual>
                <geometry>
                    <!-- The right wheel is represented as a
cylinder with the specified radius and length -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the right wheel to white -
->
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the right wheel -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
            </collision>
        </link>

        <!-- Define the joint for the front right wheel -->
        <joint name="wheel_front_right_joint" type="continuous">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="-0.2 0.25 0.0" rpy="1.57 0.0 0.0"/>
            <!-- The parent link of this joint -->
            <parent link="base"/>
            <!-- The child link of this joint -->
```

```
            <child link="wheel_front_right_link"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define the front left wheel link of the robot -->
        <link name="wheel_front_left_link">
            <!-- Inertial properties of the left wheel -->
            <inertial>
                <!-- Mass of the left wheel -->
                <mass value="2" />
                <!-- Inertia tensor of the left wheel -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>
            <!-- Visual properties of the left wheel -->
            <visual>
                <geometry>
                    <!-- The left wheel is represented as a cylinder
with the specified radius and length -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the left wheel to white --
>
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the left wheel -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <cylinder radius="0.15" length="0.1"/>
                </geometry>
                <contact_coefficients mu="1" kp="1e+13" kd="1.0"/>
            </collision>
        </link>

        <!-- Define the joint for the front left wheel -->
        <joint name="wheel_front_left_joint" type="continuous">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="-0.2 -0.25 0.0" rpy="1.57 0.0 0.0"/>
            <!-- The parent link of this joint -->
```

```
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="wheel_front_left_link"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define the camera link of the robot -->
        <link name="camera">
            <!-- Inertial properties of the camera -->
            <inertial>
                <!-- Mass of the camera -->
                <mass value="0.1" />
                <!-- Inertia tensor of the camera -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>
            <!-- Visual properties of the camera -->
            <visual>
                <geometry>
                    <!-- The camera is represented as a box with the
specified dimensions -->
                    <box size="0.1 0.1 0.05"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the camera to white -->
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the camera -->
            <collision>
                <geometry>
                    <!-- The collision shape is the same as the
visual shape -->
                    <box size="0.1 0.1 0.05"/>
                </geometry>
            </collision>
        </link>

        <!-- Define the joint for the camera -->
        <joint name="camera_joint" type="fixed">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="-0.35 0 0.01" rpy="0 0.0 3.14"/>
            <!-- The parent link of this joint -->
```

```
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="camera"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define the lidar link of the robot -->
        <link name="lidar">
            <!-- Inertial properties of the lidar -->
            <inertial>
                <!-- Mass of the lidar -->
                <mass value="0.5" />
                <!-- Inertia tensor of the lidar -->
                <inertia ixx="0.01" ixy="0.0" ixz="0" iyy="0.01"
iyz="0" izz="0.01" />
            </inertial>
            <!-- Visual properties of the lidar -->
            <visual>
                <geometry>
                    <!-- The lidar is represented as a cylinder with
the specified radius and length -->
                    <cylinder radius="0.1" length="0.05"/>
                </geometry>
                <material name="white">
                    <!-- Set the color of the lidar to white -->
                    <color rgba="1 1 1 1"/>
                </material>
            </visual>
            <!-- Collision properties of the lidar -->
            <collision>
                <geometry>
                    <!-- The collision shape is a box with the
specified dimensions -->
                    <box size="0.1 0.1 0.1"/>
                </geometry>
            </collision>
        </link>

        <!-- Define the joint for the lidar -->
        <joint name="lidar_joint" type="fixed">
            <!-- Origin of the joint in relation to the parent link
(base) -->
            <origin xyz="-0.285 0 0.075" rpy="0 0.0 1.57"/>
            <!-- The parent link of this joint -->
```

```
            <parent link="base"/>
            <!-- The child link of this joint -->
            <child link="lidar"/>
            <!-- The axis of rotation for the joint -->
            <axis xyz="0.0 0.0 1.0"/>
        </joint>

        <!-- Define Gazebo-specific properties for the base link -->
        <gazebo reference="base">
            <!-- Set the material of the base link in Gazebo -->
            <material>Gazebo/WhiteGlow</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the rear left
wheel link -->
        <gazebo reference="wheel_rear_left_link">
            <!-- Set the material of the left wheel link in Gazebo -
->
            <material>Gazebo/SkyBlue</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the rear right
wheel link -->
        <gazebo reference="wheel_rear_right_link">
            <!-- Set the material of the right wheel link in Gazebo
-->
            <material>Gazebo/SkyBlue</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the front left
wheel link -->
        <gazebo reference="wheel_front_left_link">
            <!-- Set the material of the left wheel link in Gazebo -
->
            <material>Gazebo/SkyBlue</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the front right
wheel link -->
        <gazebo reference="wheel_front_right_link">
            <!-- Set the material of the right wheel link in Gazebo
-->
            <material>Gazebo/SkyBlue</material>
        </gazebo>
```

```
        <!-- Define Gazebo-specific properties for the lidar -->
        <gazebo reference="lidar">
            <!-- Set the material of the lidar in Gazebo -->
            <material>Gazebo/Blue</material>
        </gazebo>

        <!-- Define Gazebo-specific properties for the camera -->
        <gazebo reference="camera">
            <!-- Set the material of the camera in Gazebo -->
            <material>Gazebo/Red</material>
        </gazebo>

        <!-- Define the Gazebo plugin for differential drive control
-->
        <gazebo>
            <plugin filename="libgazebo_ros_diff_drive.so"
name="gazebo_base_controller">
                <!-- Odometry frame for the plugin -->
                <odometry_frame>odom</odometry_frame>
                <!-- Command topic for velocity commands -->
                <commandTopic>cmd_vel</commandTopic>
                <!-- Publish odometry data -->
                <publish_odom>true</publish_odom>
                <!-- Publish odometry transform -->
                <publish_odom_tf>true</publish_odom_tf>
                <!-- Update rate for the plugin -->
                <update_rate>15.0</update_rate>
                <!-- Number of wheel pairs -->
                <num_wheel_pairs>2</num_wheel_pairs>
                <!-- Rear Left wheel joint name -->
                <left_joint>wheel_rear_left_joint</left_joint>
                <!-- Rear Right wheel joint name -->
                <right_joint>wheel_rear_right_joint</right_joint>
                <!-- Front Left wheel joint name -->
                <left_joint>wheel_front_left_joint</left_joint>
                <!-- Front Right wheel joint name -->
                <right_joint>wheel_front_right_joint</right_joint>
                <!-- Wheel separation distance -->
                <wheel_separation>0.5</wheel_separation>
                <!-- Wheel diameter -->
                <wheel_diameter>0.3</wheel_diameter>
                <!-- Maximum wheel acceleration -->
                <max_wheel_acceleration>0.7</max_wheel_acceleration>
                <!-- Maximum wheel torque -->
                <max_wheel_torque>8</max_wheel_torque>
```

```
                <!-- Base frame of the robot -->
                <robotBaseFrame>base</robotBaseFrame>
            </plugin>
        </gazebo>

        <!-- Define the Gazebo plugin for the camera -->
        <gazebo reference="camera">
            <!-- Define the camera sensor -->
            <sensor type="camera" name="camera1">
                <!-- Visualize the camera output in Gazebo -->
                <visualize>true</visualize>
                <!-- Update rate for the camera sensor -->
                <update_rate>30.0</update_rate>
                <camera name="head">
                    <!-- Horizontal field of view for the camera -->
                    <horizontal_fov>1.3962634</horizontal_fov>
                    <image>
                        <!-- Image width in pixels -->
                        <width>800</width>
                        <!-- Image height in pixels -->
                        <height>800</height>
                        <!-- Image format -->
                        <format>R8G8B8</format>
                    </image>
                    <clip>
                        <!-- Near clipping distance -->
                        <near>0.02</near>
                        <!-- Far clipping distance -->
                        <far>300</far>
                    </clip>
                </camera>
                <plugin name="camera_controller"
filename="libgazebo_ros_camera.so">
                    <!-- Always keep the camera on -->
                    <alwaysOn>true</alwaysOn>
                    <!-- Update rate for the camera controller -->
                    <updateRate>60.0</updateRate>
                    <!-- Camera name for the controller -->
                    <cameraName>/camera1</cameraName>
                    <!-- Image topic name -->
                    <imageTopicName>image_raw</imageTopicName>
                    <!-- Camera info topic name -->

<cameraInfoTopicName>info_camera</cameraInfoTopicName>
                    <!-- Frame name for the camera -->
```

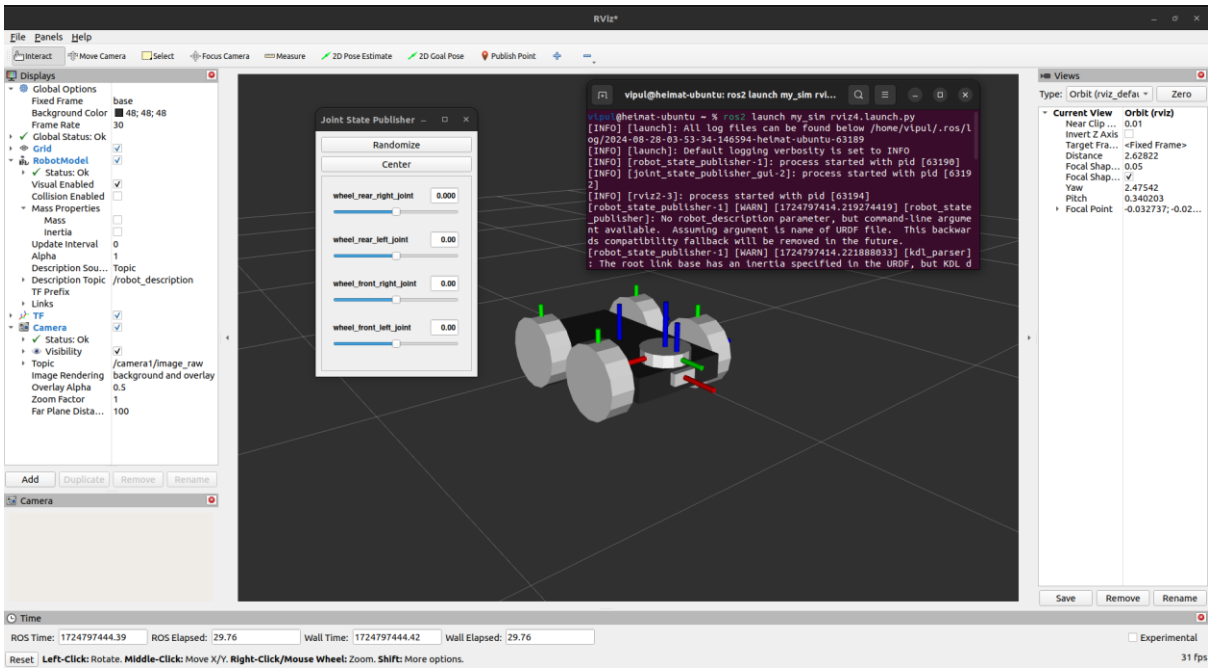
```
                    <frameName>camera</frameName>
                    <!-- Baseline for stereo camera setup (not used
here) -->
                    <hackBaseline>0.07</hackBaseline>
                </plugin>
            </sensor>
        </gazebo>

        <!-- Define the Gazebo plugin for the lidar -->
        <gazebo reference="lidar">
            <!-- Define the lidar sensor -->
            <sensor name="lidar" type="ray">
                <!-- Visualize the lidar output in Gazebo -->
                <visualize>true</visualize>
                <!-- Update rate for the lidar sensor -->
                <update_rate>12.0</update_rate>
                <plugin filename="libgazebo_ros_ray_sensor.so"
name="gazebo_lidar">
                    <!-- Output type for the lidar sensor -->
                    <output_type>sensor_msgs/LaserScan</output_type>
                    <!-- Frame name for the lidar sensor -->
                    <frame_name>lidar</frame_name>
                </plugin>
                <ray>
                    <scan>
                        <horizontal>
                            <!-- Number of samples for the lidar
scan -->
                            <samples>360</samples>
                            <!-- Resolution of the lidar scan -->
                            <resolution>1</resolution>
                            <!-- Minimum angle for the lidar scan --
>
                            <min_angle>0.00</min_angle>
                            <!-- Maximum angle for the lidar scan --
>
                            <max_angle>3.14</max_angle>
                        </horizontal>
                    </scan>
                    <range>
                        <!-- Minimum range for the lidar sensor -->
                        <min>0.120</min>
                        <!-- Maximum range for the lidar sensor -->
                        <max>3.5</max>
                        <!-- Resolution of the lidar range -->
```

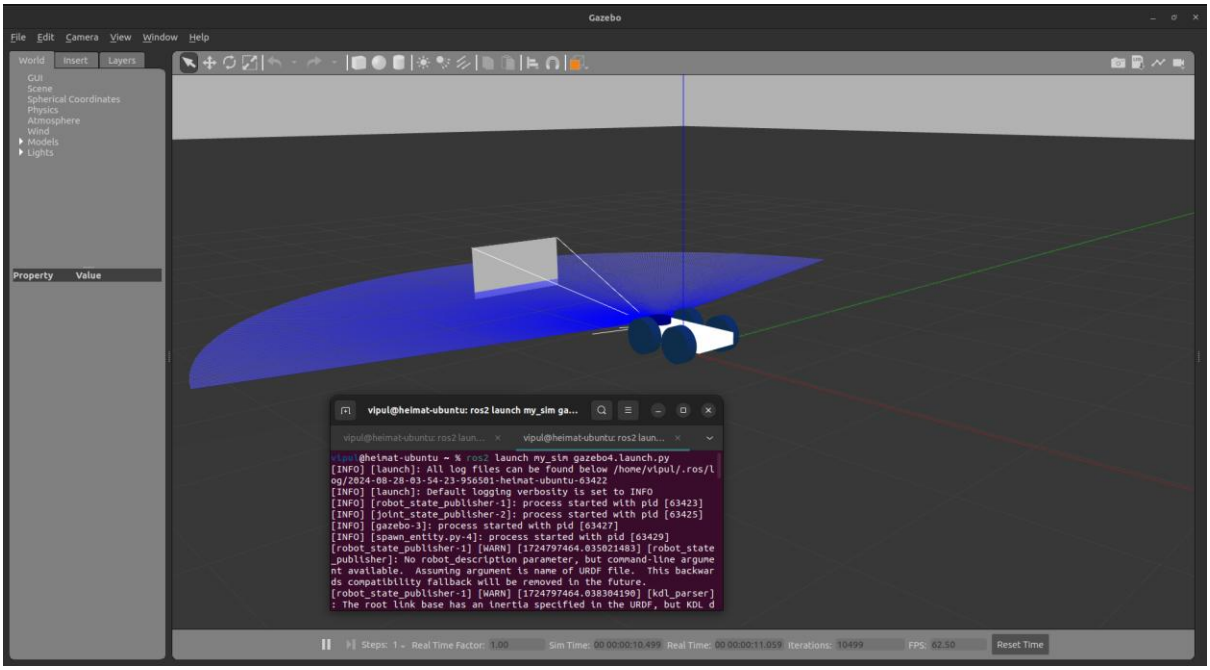
```
                        <resolution>0.015</resolution>
                    </range>
                </ray>
            </sensor>
        </gazebo>


    </robot>
```

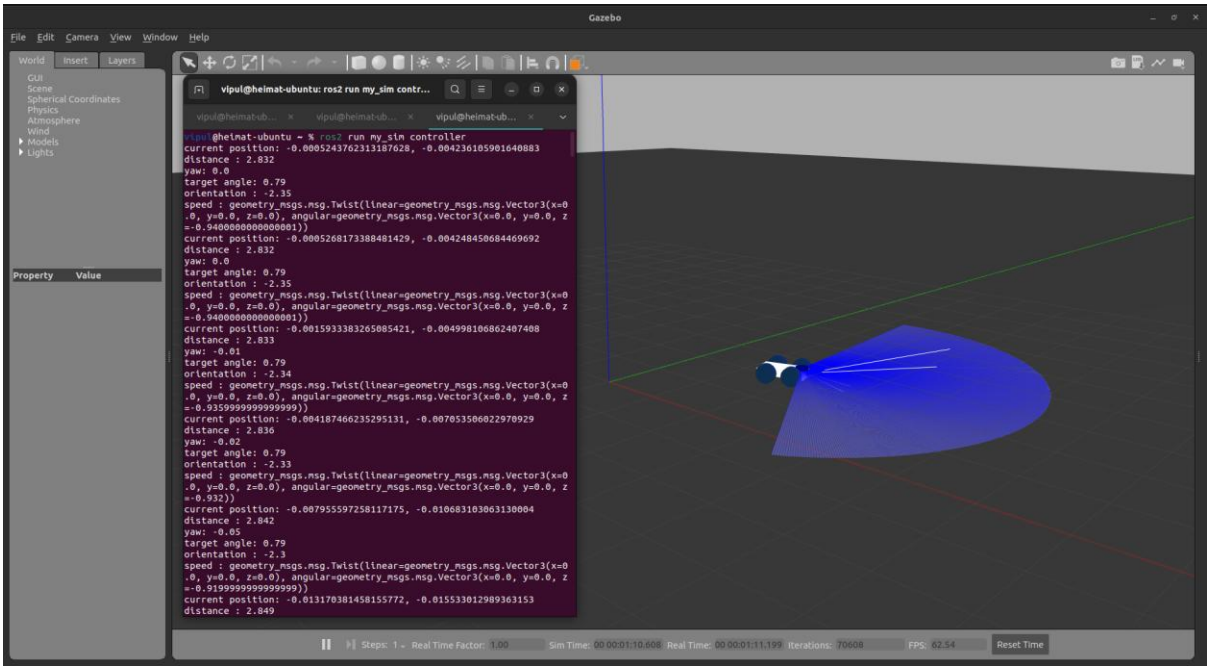
12. In the launch folder, create new launch files for gazebo and rviz so that they use the `four_wheeled_robot.urdf` file and also publish their status. Name them as `gazebo4.launch.py` and `rviz4.launch.py` respectively. Content remains same as `Step 4`, except `urdf = '/home/vipul/ros2_ws/src/my_sim/urdf/three_wheeled_robot.urdf'` is replaced with `urdf = '/home/vipul/ros2_ws/src/my_sim/urdf/three_wheeled_robot.urdf'`

13. Repeat `Step 7-10` to rebuild the package, launch and setup rviz, launch gazebo, and run the controller respectively

    Note: replace `rviz.launch.py` with `rviz4.launch.py` and `gazebo.launch.py` with `gazebo4.launch.py`




Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
vipul@heimat-ubuntu: ros2 launch my_sim ga...
vipul@heimat-ubuntu ~ % ros2 launch my_sim_gazebo4.launch.py
[INFO] [launch]: All log files can be found below /home/vipul/.ros/log/2024-08-28-03-54-23-956501-heimat-ubuntu-63422
[INFO] [launch]: Default logging verbosity is set to INFO
[INFO] [robot_state_publisher-1]: process started with pid [63423]
[INFO] [joint_state_publisher-2]: process started with pid [63425]
[INFO] [gazebo-3]: process started with pid [63427]
[INFO] [spawn_entity.py-4]: process started with pid [63429]
[robot_state_publisher-1] [WARN] [1724797464.035021483] [robot_state_publisher]: No robot_description parameter, but command-line argument available. Assuming argument is name of URDF file. This backwards compatibility fallback will be removed in the future.
[robot_state_publisher-1] [WARN] [1724797464.038304190] [kdl_parser]: The root link base has an inertia specified in the URDF, but KDL d
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:00:10.499 Real Time: 00 00:00:11.059 Iterations: 10499 FPS: 62.50 Reset Time


Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
vipul@heimat-ubuntu: ros2 run my_sim contr...
vipul@heimat-ubuntu ~ % ros2 run my_sim controller
current position: -0.0005243762313187628, -0.004236105901640883
distance : 2.832
yaw: 0.0
target angle: 0.79
orientation : -2.35
speed : geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=-0.9400000000000001))
current position: -0.0005268173388481429, -0.004248450684469692
distance : 2.832
yaw: 0.0
target angle: 0.79
orientation : -2.35
speed : geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=-0.9400000000000001))
current position: -0.0015933383265085421, -0.004998106862407408
distance : 2.833
yaw: -0.01
target angle: 0.79
orientation : -2.34
speed : geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=-0.9359999999999999))
current position: -0.004187466235295131, -0.007053506022970929
distance : 2.836
yaw: -0.02
target angle: 0.79
orientation : -2.33
speed : geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=-0.932))
current position: -0.007955597258117175, -0.010683103063130004
distance : 2.842
yaw: -0.05
target angle: 0.79
orientation : -2.3
speed : geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=-0.9199999999999999))
current position: -0.013170381458155772, -0.015533012989363153
distance : 2.849
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:01:10.608 Real Time: 00 00:01:11.199 Iterations: 70608 FPS: 62.54 Reset Time