Lab 4 – Report

Meet Kansara – 220929270 Roll no. 54

Aim: To explore the fundamentals of creating and controlling a mobile robot using URDF in ROS2.

Code Execution and analysis:

URDF file:

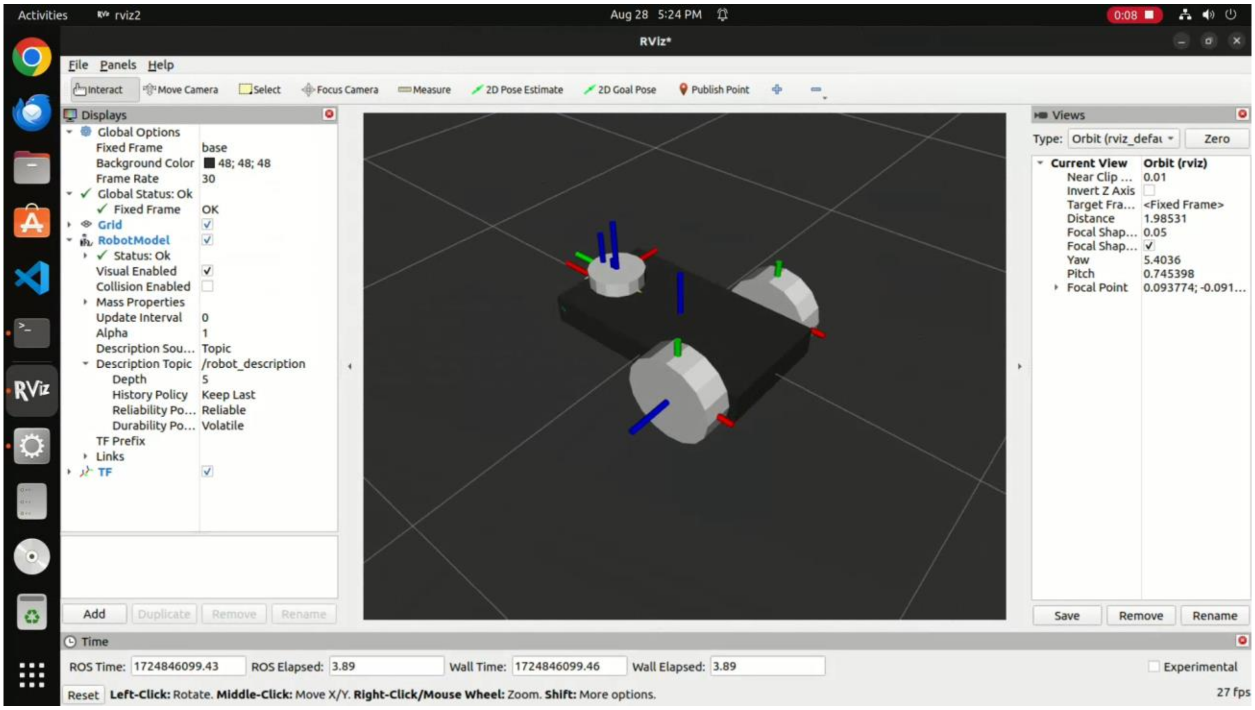
```
<?xml version="1.0" ?>
<robot name="three_wheeled_robot">
  <link name="base">
    <visual>
      <geometry>
        <box size="0.75 0.4 0.1"/>
      </geometry>
      <material name="gray">
        <color rgba=".2 .2 .2 1"/>
      </material>
    </visual>
    <inertial>
      <mass value="1"/>
      <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>
```

```xml
  <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>
```

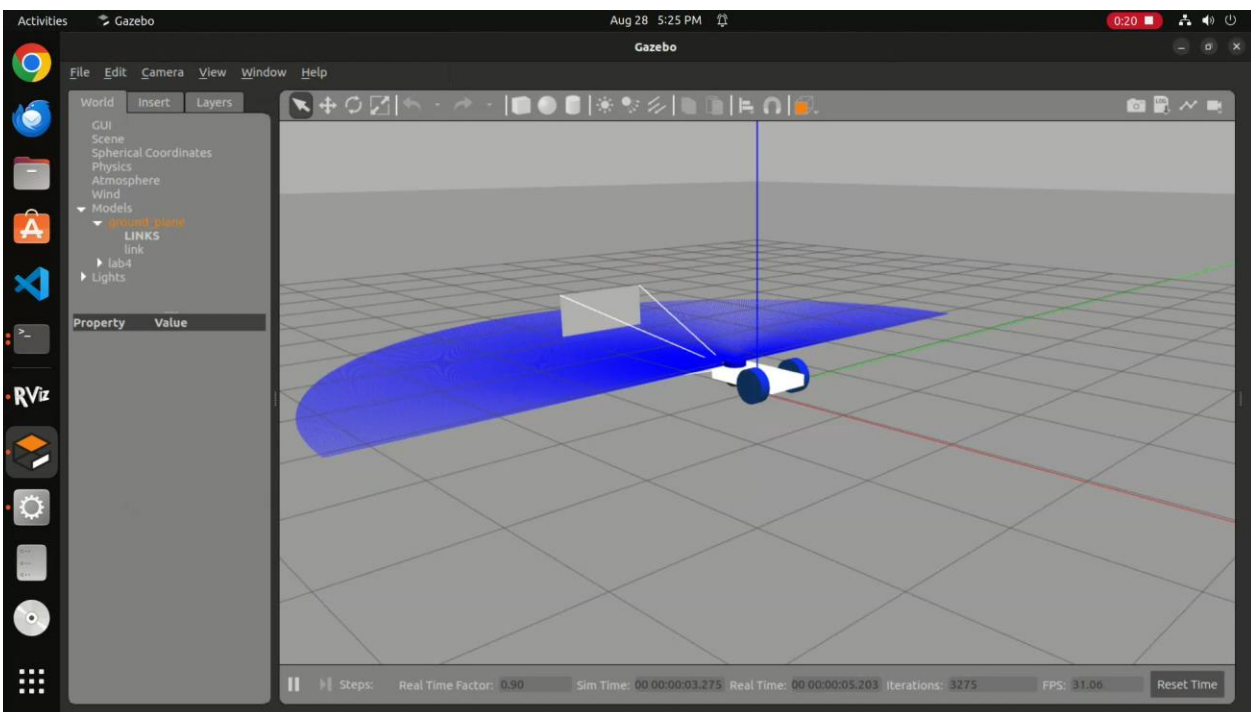
```xml
        <box size="0.1 0.1 0.05"/>
      </geometry>
    </collision>
  </link>

  <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>

  <!-- Material Assignments -->
  <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 Drive Plugin -->
  <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>
```

```xml
      <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>
```

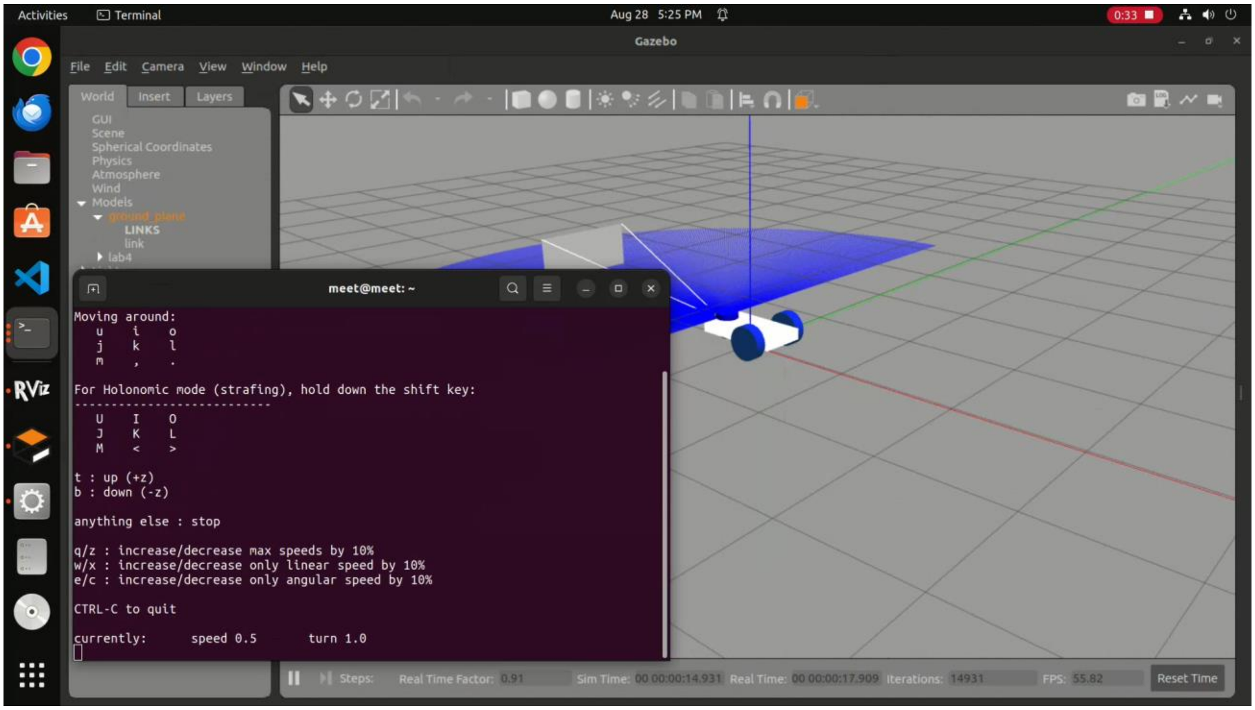
URDF Model in Rviz:



URDF Model in Gazebo:



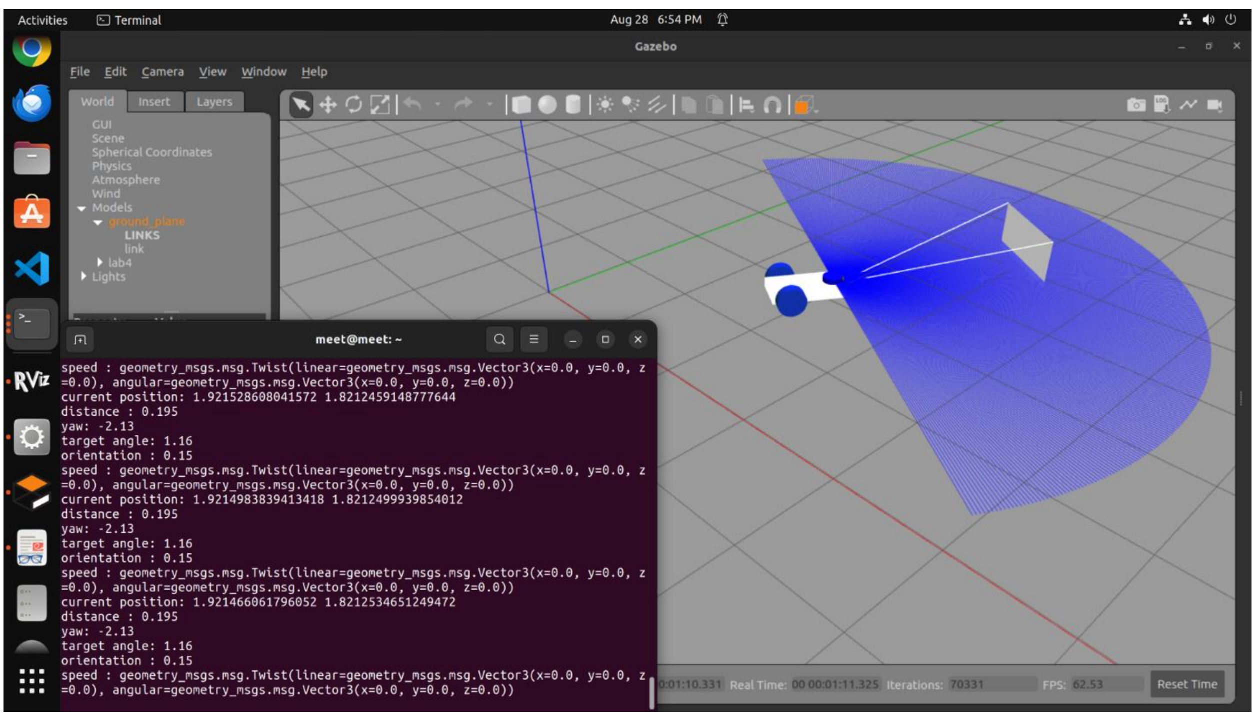
1. Teleop Control of a Three-Wheeled Robot:



2. P-Controller for a Three-Wheeled Robot:

```python
#!/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)))
```

```python
        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()
```



Conclusion: Responsive control of the three-wheeled robot using teleop keys and a P controller has been achieved.