AREA MAPPING ROBOT WITH FOUR WHEELS, LiDAR AND CAMERA

Submitted By

K.S. PAVAL (CB.EN.U4AIE20047)

SHREYA SANGHAMITRA(CB.EN.U4AIE20066)

....

For the Completion of

19AIE213 - ROBOTIC OPERATING SYSTEMS

CEN

11th July 2022

INDEX

Introduction

Area Mapping Robot with Four Wheels, LiDAR and Camera

In this project, a robot is created which helps to map the area in real-time with a sensor known as the LiDAR sensor. LiDAR (**Li**ght **D**etection **A**nd **R**anging) is a sensor which works when the laser light from a source (transmitter) is reflected from the objects. It is used for determining ranges by targeting an object or a surface with a laser and measuring the time for the reflected light to return to the receiver. A camera is attached so that the user can see the object/area that the LiDAR sensor is detecting.

Objective

To design a robot with four wheels with LiDAR sensor and camera for area mapping.

Components Used

- 4 wheel-robot
- LiDAR sensor
- Camera

Tools

The simulation of the robot is done in GAZEBO and the prototype can be viewed in RVIZ where every object/path detected by the camera and the LiDAR sensor can also be viewed.

Launch Files Used

- **world.launch**

In this launch file we spawn our robot in the world that we created.

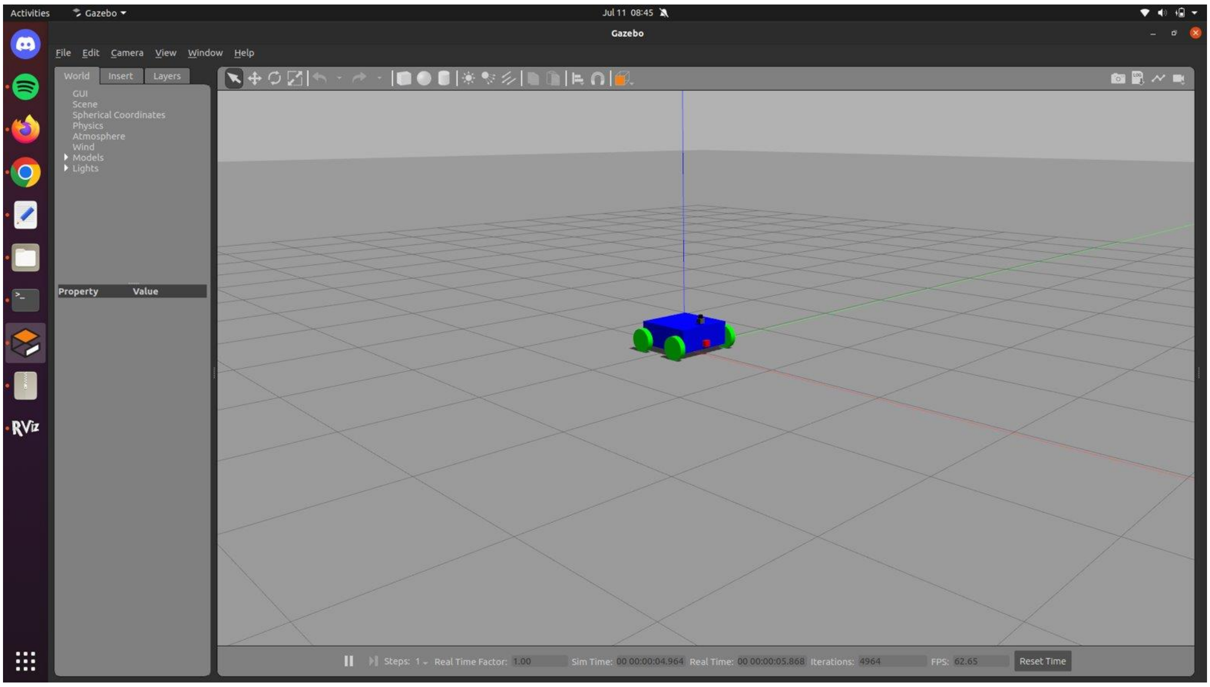
```xml
<?xml version="1.0" encoding="UTF-8"?>

<launch>
	<!-- Robot pose -->
	<arg name="x" default="0"/>
	<arg name="y" default="0"/>
	<arg name="z" default="0"/>
	<arg name="roll" default="0"/>
	<arg name="pitch" default="0"/>
	<arg name="yaw" default="0"/>
	<arg name="robot_name" default="atom"/>

	<!-- Launch other relevant files-->
	<include file="$(find atom)/launch/robot_description.launch"/>

	<!-- World File -->
	<arg name="world_file" default="$(find atom)/worlds/empty.world"/>

	<!-- Launch Gazebo World -->
	<include file="$(find gazebo_ros)/launch/empty_world.launch">
		<arg name="use_sim_time" value="true"/>
		<arg name="verbose" value="false"/>
		<arg name="debug" value="false"/>
		<arg name="gui" value="true" />
		<arg name="world_name" value="$(arg world_file)"/>
	</include>

	<!-- Spawn My Robot -->
	<node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
		args="-urdf -param robot_description -model atom
			-x $(arg x) -y $(arg y) -z $(arg z)
			-R $(arg roll) -P $(arg pitch) -Y $(arg yaw)"/>

	<!--launch rviz-->
	<node name="rviz" pkg="rviz" type="rviz" respawn="false"
		args="-d $(find atom)/default.rviz"/>

</launch>
```

- **Empty.world**

  Here a world environment is created where a ground and sun has been included.

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

<sdf version="1.4">

        <world name="default">
                <include>
                        <uri>model://ground_plane</uri>
                </include>


                <!-- Light source -->
                <include>
                        <uri>model://sun</uri>
                </include>

                <!-- World camera -->
                <gui fullscreen='0'>
                        <camera name='world_camera'>
                                <pose>4.927360 -4.376610 3.740080 0.000000 0.275643 2.356190</pose>
                                <view_controller>orbit</view_controller>
                        </camera>
                </gui>
        </world>
</sdf>
```

And here we can see the robot in gazebo



- **Robot_description.launch**

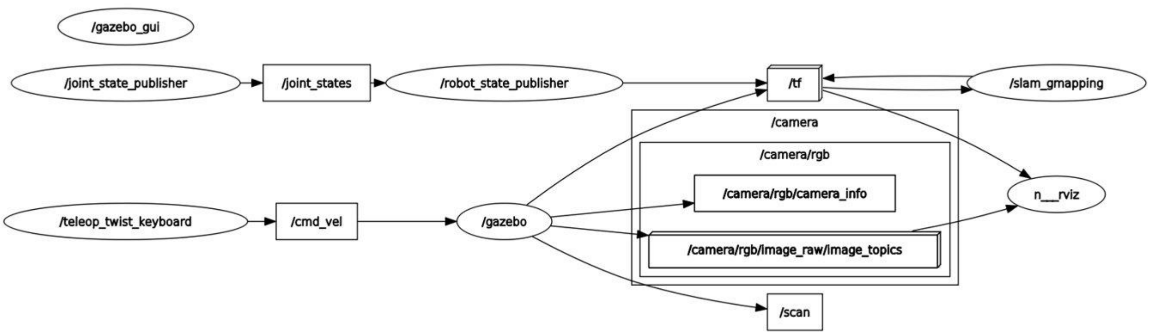
```xml
<?xml version="1.0"?>
<launch>

    <!-- send urdf to param server -->
    <param name="robot_description" command="$(find xacro)/xacro --inorder '$(find atom)/urdf/atom.xacro'" />

    <!-- Send fake joint values-->
    <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher">
        <param name="use_gui" value="false"/>
    </node>

    <!-- Send robot states to tf -->
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" respawn="false" output="screen"/>

</launch>

```

- **Slam_gmapping.launch**

```xml
<?xml version="1.0"?>
<launch>
  <node name="slam_gmapping" pkg="gmapping" type="slam_gmapping">
    <remap from="/scan" to="/atom/sensor_laser/scan"/>
    <param name="base_frame" value="base_footprint"/>
  </node>
</launch>
```

rqt_graph

```
parallels@parallels:~/catkin_ws/src/atom$ rosrun rqt_graph rqt_graph
WARNING: Package name "rosActionExample" does not follow the naming conventions.
 It should start with a lower case letter and only contain lower case letters, d
igits, underscores, and dashes.
WARNING: Package name "rosActionExample" does not follow the naming conventions.
 It should start with a lower case letter and only contain lower case letters, d
igits, underscores, and dashes.
```



URDF Files Used

- atom.gazebo

Open

atom.gazebo
~/catkin_ws/src/Robotics_ws-main/atom/urdf

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

  <gazebo>

    <plugin name="skid_steer_drive_controller" filename="libgazebo_ros_skid_steer_drive.so">
      <updateRate>10.0</updateRate>
      <robotNamespace>/</robotNamespace>
      <leftFrontJoint>left_wheel_hinge_front</leftFrontJoint>
      <rightFrontJoint>right_wheel_hinge_front</rightFrontJoint>
      <leftRearJoint>left_wheel_hinge_back</leftRearJoint>
      <rightRearJoint>right_wheel_hinge_back</rightRearJoint>
      <wheelSeparation>0.4</wheelSeparation>
      <wheelDiameter>0.2</wheelDiameter>
      <robotBaseFrame>robot_footprint</robotBaseFrame>
      <torque>10</torque>

      <topicName>cmd_vel</topicName>
      <odometryTopic>odom</odometryTopic>
      <odometryFrame>odom</odometryFrame>

      <commandTopic>cmd_vel</commandTopic>
      <topic_name_twist>cmd_vel</topic_name_twist>
      <topic_name_odometry>odom</topic_name_odometry>
      <topic_name_joint>joint</topic_name_joint>

      <broadcastTF>true</broadcastTF>

      <covariance_x>0.0001</covariance_x>
      <covariance_y>0.0001</covariance_y>
      <covariance_yaw>0.01</covariance_yaw>

    </plugin>

  </gazebo>

```

```xml
  </gazebo>

  <!-- camera -->
  <gazebo reference="camera">
    <sensor type="camera" name="camera1">
      <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>0.0</updateRate>
        <cameraName>camera</cameraName>
        <imageTopicName>rgb/image_raw</imageTopicName>
        <cameraInfoTopicName>rgb/camera_info</cameraInfoTopicName>
        <frameName>camera</frameName>
        <hackBaseline>0.07</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
      </plugin>
    </sensor>
  </gazebo>
```

```xml
  <!-- hokuyo -->
  <gazebo reference="hokuyo">
    <sensor type="ray" name="head_hokuyo_sensor">
      <pose>0 0 0 0 0 0</pose>
      <visualize>false</visualize>
      <update_rate>40</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>720</samples>
            <resolution>1</resolution>
            <min_angle>-1.570796</min_angle>
            <max_angle>1.570796</max_angle>
          </horizontal>
        </scan>
        <range>
          <min>0.10</min>
          <max>30.0</max>
          <resolution>0.01</resolution>
        </range>
        <noise>
          <type>gaussian</type>
          <!-- Noise parameters based on published spec for Hokuyo laser
               achieving "+-30mm" accuracy at range < 10m.  A mean of 0.0m and
               stddev of 0.01m will put 99.7% of samples within 0.03m of the true
               reading. -->
          <mean>0.0</mean>
          <stddev>0.01</stddev>
        </noise>
      </ray>
      <plugin name="gazebo_ros_head_hokuyo_controller" filename="libgazebo_ros_laser.so">
        <topicName>/scan</topicName>
        <frameName>hokuyo</frameName>
      </plugin>
    </sensor>
  </gazebo>


</robot>
```

- **atom.xacro**

Open

*atom.xacro
~/catkin_ws/src/atom/urdf

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

<robot name="atom" xmlns:xacro="http://www.ros.org/wiki/xacro">
        <xacro:property name="robot_name" value="atom" />
        <xacro:property name="robot_chassis_mass" value="15"/>
        <xacro:property name="robot_chassis_length" value="0.2"/>
        <xacro:property name="robot_chassis_radius" value="0.25"/>
        <xacro:property name="robot_caster_wheel_radius" value="0.05"/>
        <xacro:property name="robot_caster_wheel_radius_collision" value="0.0499"/>

        <xacro:property name="robot_wheel_mass" value="5"/>
        <xacro:property name="robot_wheel_length" value="0.05"/>
        <xacro:property name="robot_wheel_radius" value="0.1"/>

        <xacro:property name="camera_mass" value="0.1"/>
        <xacro:property name="hokoyu_mass" value="1e-5"/>

         <xacro:property name="laser_size_x" value="0.03"/>
         <xacro:property name="laser_size_y" value="0.03"/>
         <xacro:property name="laser_size_z" value="0.04"/>
         <xacro:property name="laser_origin_x" value="0.065"/>
         <xacro:property name="laser_origin_y" value="0"/>
         <xacro:property name="laser_origin_z" value="0.035"/>

        <!-- Make Chassis of Bot -->
        <link name="chassis">
                <pose>0 0 0.1 0 0 0</pose>

                <inertial>
                        <mass value="${robot_chassis_mass}"/>
                <origin xyz="0.0 0 0" rpy=" 0 0 0"/>

                <inertia
                        ixx="0.147116667" ixy="0" ixz="0"
                        iyy="0.334951167" iyz="0"
                        izz="0.3978345"
                />
                </inertial>

                <collision name="collision">
                        <origin xyz="0 0 0.05" rpy=" 0 0 0"/>
                        <geometry>
                                <box size="0.5 0.5 0.2"/>
                        </geometry>
                </collision>

                <visual name="chassis_visual">
                        <origin xyz="0 0 0.05" rpy=" 0 0 0"/>
                        <geometry>
                                <box size="0.5 0.5 0.2"/>
                        </geometry>

                </visual>

        </link>


```

*atom.xacro
~/catkin_ws/src/atom/urdf

```xml
	<!-- Right Wheel Back -->
	<link name="right_wheel_back">
		<inertial>
			<mass value="${robot_wheel_mass}"/>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<inertia
				ixx="0.1" ixy="0.0" ixz="0.0"
				iyy="0.1" iyz="0.0"
				izz="0.1"
			/>
		</inertial>

		<visual>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<geometry>
				<cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
			</geometry>
		</visual>

		<collision>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<geometry>
				<cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
			</geometry>
		</collision>

	</link>

	<!-- Right Wheel Front-->
	<link name="right_wheel_front">
		<inertial>
			<mass value="${robot_wheel_mass}"/>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<inertia
				ixx="0.1" ixy="0.0" ixz="0.0"
				iyy="0.1" iyz="0.0"
				izz="0.1"
			/>
		</inertial>

		<visual>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<geometry>
				<cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
			</geometry>
		</visual>

		<collision>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<geometry>
				<cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
			</geometry>
		</collision>

	</link>
```

*atom.xacro
~/catkin_ws/src/atom/urdf

```xml
	<!-- Left wheel Back-->
	<link name="left_wheel_back">
		<inertial>
			<mass value="${robot_wheel_mass}"/>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<inertia
				ixx="0.1" ixy="0.0" ixz="0.0"
				iyy="0.1" iyz="0.0"
				izz="0.1"
			/>
		</inertial>

		<visual>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<geometry>
				<cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
			</geometry>
		</visual>

		<collision>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<geometry>
				<cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
			</geometry>
		</collision>

	</link>

	<!-- Left wheel Front-->
	<link name="left_wheel_front">
		<inertial>
			<mass value="${robot_wheel_mass}"/>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<inertia
				ixx="0.1" ixy="0.0" ixz="0.0"
				iyy="0.1" iyz="0.0"
				izz="0.1"
			/>
		</inertial>

		<visual>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<geometry>
				<cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
			</geometry>
		</visual>

		<collision>
			<origin xyz="0 0 0" rpy="0 1.5707 1.5707"/>
			<geometry>
				<cylinder radius="${robot_wheel_radius}" length="${robot_wheel_length}"/>
			</geometry>
		</collision>

	</link>
```

Open *atom.xacro ~/catkin_ws/src/atom/urdf Save

```xml

	<!-- Camera -->
	<link name="camera">
		<inertial>
			<mass value="${camera_mass}"/>
			<origin xyz="0 0 0" rpy="0 0 0"/>
			<inertia
			ixx="1e-6" ixy="0.0" ixz="0.0"
			iyy="1e-6" iyz="0.0"
			izz="1e-6"
		       />
		</inertial>

		<visual>
			<origin xyz="0 0 0" rpy="0 0 0"/>
			<geometry>
				<box size="0.05 0.05 0.05"/>
			</geometry>
		</visual>

		<collision>
			<origin xyz="0 0 0" rpy="0 0 0"/>
			<geometry>
				<box size="0.05 0.05 0.05"/>
			</geometry>
		</collision>
	</link>

	<!-- Hokuyo Lidar -->
	<link name="hokuyo">
		<inertial>
			<mass value="${hokoyu_mass}"/>
			<origin xyz="0 0 0" rpy="0 0 0"/>

			<inertia
			ixx="1e-6" ixy="0.0" ixz="0.0"
			iyy="1e-6" iyz="0.0"
			izz="1e-6"
		       />
		</inertial>

		<visual>
			<origin xyz="0 0 0" rpy="0 0 0"/>
			<geometry>
				<mesh filename="package://atom/meshes/hokuyo.dae"/>
			</geometry>
		</visual>

		<collision>
			<origin xyz="0 0 0" rpy="0 0 0"/>
			<geometry>
				<box size="0.1 0.1 0.1"/>
			</geometry>
		</collision>
	</link>

```

```
<!-- Project center to the ground -->
<link name="robot_footprint"></link>



<!-- Define Joints -->

<!-- Right Wheel Joint Back-->
<joint type="continuous" name="right_wheel_hinge_back">
    <origin xyz="-0.2 -0.30 0" rpy="0 0 0" />
    <parent link="chassis"/>
    <child link="right_wheel_back" />
    <axis xyz="0 1 0" rpy="0 0 0" />
    <limit effort="10000" velocity="1000" />
    <dynamics damping="1.0" friction="1.0" />
</joint>

<!-- Right Wheel Joint Front-->
<joint type="continuous" name="right_wheel_hinge_front">
    <origin xyz="0.2 -0.30 0" rpy="0 0 0" />
    <parent link="chassis"/>
    <child link="right_wheel_front" />
    <axis xyz="0 1 0" rpy="0 0 0" />
    <limit effort="10000" velocity="1000" />
    <dynamics damping="1.0" friction="1.0" />
</joint>


<!-- Left Wheel Joint Back-->
<joint type="continuous" name="left_wheel_hinge_back">
    <origin xyz="-0.2 0.30 0" rpy="0 0 0" />
    <parent link="chassis"/>
    <child link="left_wheel_back" />
    <axis xyz="0 1 0" rpy="0 0 0" />
    <limit effort="10000" velocity="1000" />
    <dynamics damping="1.0" friction="1.0" />
</joint>

<!-- Left Wheel Joint Front-->
<joint type="continuous" name="left_wheel_hinge_front">
    <origin xyz="0.2 0.30 0" rpy="0 0 0" />
    <parent link="chassis"/>
    <child link="left_wheel_front" />
    <axis xyz="0 1 0" rpy="0 0 0" />
    <limit effort="10000" velocity="1000" />
    <dynamics damping="1.0" friction="1.0" />
</joint>

<!-- Camera Joint -->
<joint name="camera_joint" type="fixed">
        <origin xyz="0.26 0 0" rpy="0 0 0" />
        <parent link="chassis"/>
        <child link="camera" />
        <axis xyz="0 1 0"/>
</joint>

<!-- Hokoyu Joint -->
<joint name="hokuyo_joint" type="fixed">
    <origin xyz="0.2 0 0.2" rpy="0 0 0" />
    <parent link="chassis"/>
    <child link="hokuyo" />
    <axis xyz="0 1 0"/>
</joint>

<joint name="robot_footprint_joint" type="fixed">
        <origin xyz="0 0 0" rpy="0 0 0" />
        <parent link="robot_footprint"/>
        <child link="chassis" />
</joint>
```

```
        <!-- Color of bot -->
        <gazebo reference="left_wheel_front">
                <material>Gazebo/Green</material>
                <kp>1000000.0</kp> <!-- kp and kd for rubber -->
                <kd>100.0</kd>
                <mu1>1.0</mu1>
                <mu2>1.0</mu2>
                <maxVel>1.0</maxVel>
                <minDepth>0.00</minDepth>
        </gazebo>

        <gazebo reference="left_wheel_back">
                <material>Gazebo/Green</material>
                <kp>1000000.0</kp> <!-- kp and kd for rubber -->
                <kd>100.0</kd>
                <mu1>1.0</mu1>
                <mu2>1.0</mu2>
                <maxVel>1.0</maxVel>
                <minDepth>0.00</minDepth>
        </gazebo>

        <gazebo reference="right_wheel_front">
                <material>Gazebo/Green</material>
                <kp>1000000.0</kp> <!-- kp and kd for rubber -->
                <kd>100.0</kd>
                <mu1>1.0</mu1>
                <mu2>1.0</mu2>
                <maxVel>1.0</maxVel>
                <minDepth>0.00</minDepth>
        </gazebo>
        <gazebo reference="right_wheel_back">
                <material>Gazebo/Green</material>
                <kp>1000000.0</kp> <!-- kp and kd for rubber -->
                <kd>100.0</kd>
                <mu1>1.0</mu1>
                <mu2>1.0</mu2>
                <maxVel>1.0</maxVel>
                <minDepth>0.00</minDepth>
        </gazebo>
        <!--<gazebo reference="right_wheel">
                <material>Gazebo/Green</material>
        </gazebo>-->

        <gazebo reference="camera">
                <material>Gazebo/Red</material>
        </gazebo>

        <gazebo reference="chassis">
                <material>Gazebo/Blue</material>
        </gazebo>

        <!-- Motor, Camera and Lidar Simulation -->
        <xacro:include filename="$(find atom)/urdf/atom.gazebo" />

</robot>
```

Simulation

We first run roscore

Then in another terminal we open a gazebo world where we want our robot simulation to happen:

```
parallels@parallels:~$ roslaunch my_worlds world1.launch
WARNING: Package name "rosActionExample" does not follow the naming conventions.
 It should start with a lower case letter and only contain lower case letters, d
igits, underscores, and dashes.
... logging to /home/parallels/.ros/log/7f02f652-00c3-11ed-b95b-470ab3f4e0ee/ros
launch-parallels-7721.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

WARNING: Package name "rosActionExample" does not follow the naming conventions.
 It should start with a lower case letter and only contain lower case letters, d
igits, underscores, and dashes.
started roslaunch server http://parallels:44965/
```

Then in another terminal we will put in the commands to spawn our robot in the gazebo world we had opened:

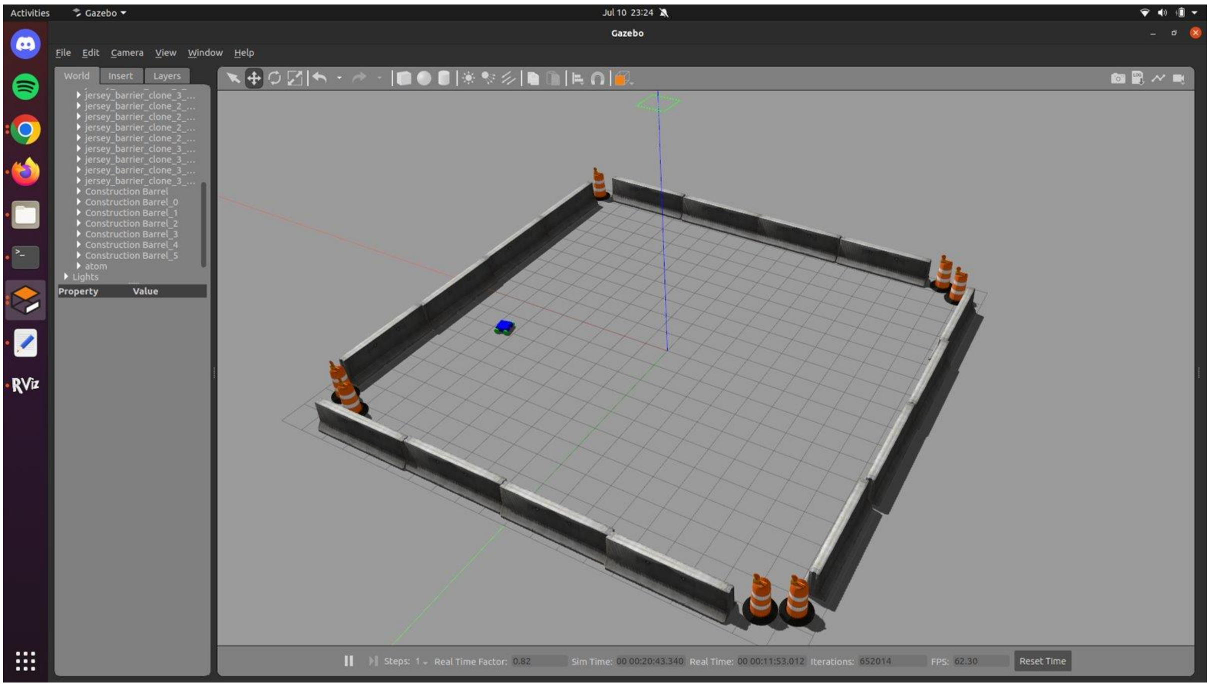
```
parallels@parallels:~$ roslaunch atom world.launch
WARNING: Package name "rosActionExample" does not follow the naming conventions.
 It should start with a lower case letter and only contain lower case letters, d
igits, underscores, and dashes.
... logging to /home/parallels/.ros/log/7f02f652-00c3-11ed-b95b-470ab3f4e0ee/ros
launch-parallels-7928.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

WARNING: Package name "rosActionExample" does not follow the naming conventions.
 It should start with a lower case letter and only contain lower case letters, d
igits, underscores, and dashes.
xacro: in-order processing became default in ROS Melodic. You can drop the optio
n.
started roslaunch server http://parallels:46067/
```

At the end in order to move our robot in the gazebo world we give the following commands in the terminal:

```
parallels@parallels:~$ rosrun teleop_twist_keyboard teleop_twist_keyboard.py

Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

For Holonomic mode (strafing), hold down the shift key:
---------------------------
   U    I    O
   J    K    L
   M    <    >

t : up (+z)
b : down (-z)

anything else : stop

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%
```

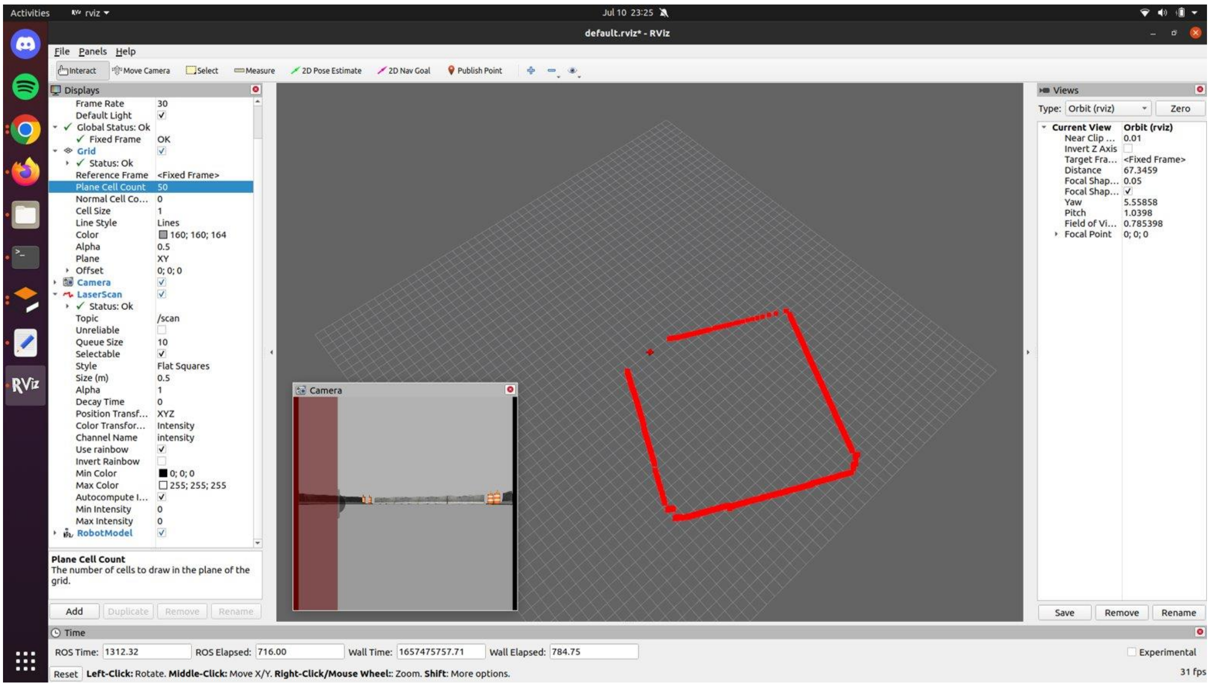
- **Gazebo**

  Here we can operate our robot in the gazebo world with the controller



- **RVIZ**

In RVIZ we can detect the path using LiDAR sensor and and we can see from the robot's point of view using the cameras installed in the robot



Conclusion

Thus, we have successfully implemented a robot with LiDAR sensors and camera which can be used to map the area that it is travelling.