**Department of Electrical Engineering**

**Faculty Member: Date:**

**Semester:**  **Group:**

# EE381 Robotics

**Lab 5: Gazebo Simulator and Twist Messages**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **PLO4 – CLO4** | **PLO5 - CLO5** | | **PLO8-CLO6** | **PLO9-CLO7** |
| **Name** | **Reg. No** | **Viva/Quiz/Demo** | **Analysis of Data in Lab Report** | **Modern Tool Usage** | **Ethics** | **Individual and Team Work** |
|  |  | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** | **5 Marks** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

## Introduction

This laboratory exercise is intended to introduce Gazebo, a popular physics simulator used widely with ROS. The major advantage of a simulator is that it allows for robot programming without having to buy an actual robot. The physics of the simulator closely approximates the real-life hardware implementation so it is useful for tests that risk hardware damage. This lab will utilize the robot simulation in Gazebo and will focus on “Twist” messages which are used to command the robot to move.

## Objectives

The following are the main objectives of this lab:

* Execution of Gazebo Simulator with ROS
* Cloning of an existing package from a GitHub repository
* Performing teleoperation of a robot in Gazebo
* Publishing twist messages for commanding actuator velocities

## Lab Conduct

* Respect faculty and peers through speech and actions
* The lab faculty will be available to assist the students. In case some aspect of the lab experiment is not understood, the students are advised to seek help from the faculty.
* In the tasks, there are commented lines such as #YOUR CODE STARTS HERE# where you have to provide the code. You must put the code between the #START and #END parts of these commented lines. Do NOT remove the commented lines.
* Use the tab key to provide the indentation in python.
* When you provide the code in the report, keep the font size at 12

**Theory**

Gazebo is a physics simulator that contains the robot models, the environment and its objects. Various aspects of physics such as dynamics, collision, lighting etc. are simulated in Gazebo and such simulations can be made to closely approximate the behaviors of real robots. This lab will focus on implementing a robot simulation and commanding the robot to move in its environment.

The terminal commands are given as:

**cd <directory>** change directory

**cd..** go back to previous directory

**pwd**  print the current directory

**ls**  list the contents of the current directory

**mkdir <directory>** create a directory

**python3 <script.py>** execute python script

A brief summary of the terminal commands needed for working in ROS are provided below:

* **colcon build**

build the packages whenever a node is created or modified

* **colcon build --packages-select <package\_name>**

build a specific package whenever a node is created or modified

* **. install/setup.bash**

make the terminal “aware” of the workspace (notice the dot and the space)

* **ros2 pkg create --build-type ament\_python <package\_name>**

create a new package (must be done in src directory)

* **ros2 pkg create --build-type ament\_python <package\_name> --dependencies rclpy std\_msgs geometry\_msgs**

create a new package with dependencies

* **ros2 run <package\_name> <node\_name>**

execute a node

**Lab Task 1 – Simulation Startup \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2]**

In this task, we will start Gazebo, clone the robot simulator packages from the repository and spawn the robot in the simulator.

Go to the link of the repository for the robot simulator. You will find the terminal commands for the installation which you must execute. Once complete, a workspace for the robot simulator will be created containing the required packages.

Start up the Gazebo simulator with the command given below. You need to be in the workspace root for this. (Also, ensure you have built the packages and sourced the workspace).

**gazebo**

You can install Gazebo from the following link if it is not installed:

**https://classic.gazebosim.org/tutorials?tut=install\_ubuntu&cat=install**

Follow the commands for installing the simulator packages (you will be given instructions for the TurtleBot3 package). After installing, launch the simulation and place the *Waffle* model. The robot will start with a laser span of 360 degrees. You need to take the screenshot of this robot and add it as follows.

***### 360 DEGREE LASER SCREENSHOT STARTS HERE ###***

*### 360 DEGREE LASER SCREENSHOT ENDS HERE ###*

The 360 degree laser is highly useful for practical tasks. Due to the introductory nature of the lab, we will modify the laser scanner to make the visuals easier to see. (The 360 degree span will be revisited in a later lab). You need to modify the laser before proceeding to the next tasks. Go to the models file in the package:

*turtlebot3\_simulations/turtlebot3\_gazebo/models/…/model.sdf*

Open the model.sdf file and find the <sensor name = “…” type = “ray”> tag of the laser scanner. Inside the tag, you will find the <samples>, <min\_angle> and <max\_angle> sub-tags. Lower the samples to 36 and change the span of laser from 45-180 degrees. Save the file, build the package and rerun the simulation. Now the robot will have the modified laser. You need to take the screenshot and add it to your report.

***### MODIFIED LASER SCREENSHOT STARTS HERE ###***

*### MODIFIED LASER SCREENSHOT ENDS HERE ###*

**Lab Task 2 – Teleoperation \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [1]**

In the simulation (with the modified laser), open a new terminal and launch the following command to start the teleoperation node:

**ros2 run teleop\_twist\_keyboard teleop\_twist\_keyboard**

With the terminal still active, press the mentioned keys to move the robot around. You need to make a video showing the teleop command being given then moving the robot around. Ensure that your group names are also shown on the terminal. The video will be named lab5\_teleop for the submission.

**Lab Task 3 – Linear and Angular Velocities \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2]**

Before publishing velocity commands to the robot via topics, it is helpful to gain insights into the concepts of linear and angular velocities. For this, the TurtleSim program will be used. Open the TurtleSim node and start up the teleoperation node (which you will need to alter the angular orientation).

**ros2 run turtlesim turtlesim\_node**

**ros2 run turtlesim turtle\_teleop\_key**

The velocity commands are given in the *cmd\_vel* topic. The data type of such commands is the “Twist” message which contains the linear x, y, z and the angular x, y, z velocities. The turtle can only move for the linear x and angular z velocities only. It cannot move/strafe sideways in the linear y direction (the other 3 velocities do not exist as the TurtleSim is a 2-D environment). In a new terminal, use the following command to publish a Twist message to move the turtle:

**ros2 topic pub --once turtle1/cmd\_vel geometry\_msgs/msg/Twist "linear:**

**x: 0.0**

**y: 0.0**

**z: 0.0**

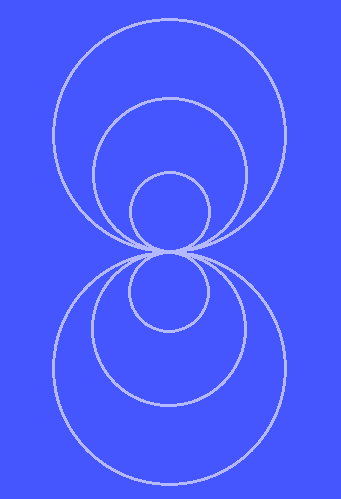
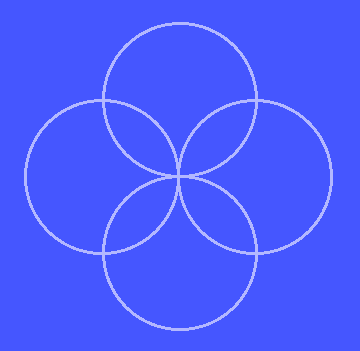
**angular:**

**x: 0.0**

**y: 0.0**

**z: 0.0"**

Experiment with the linear and angular velocities. Then create the following two patterns and take their screenshots showing the turtle. Hint: try experimenting with only the angular velocities first.

* *

***### PATTERN 1 SCREENSHOT STARTS HERE ###***

*### PATTERN 1 SCREENSHOT ENDS HERE ###*

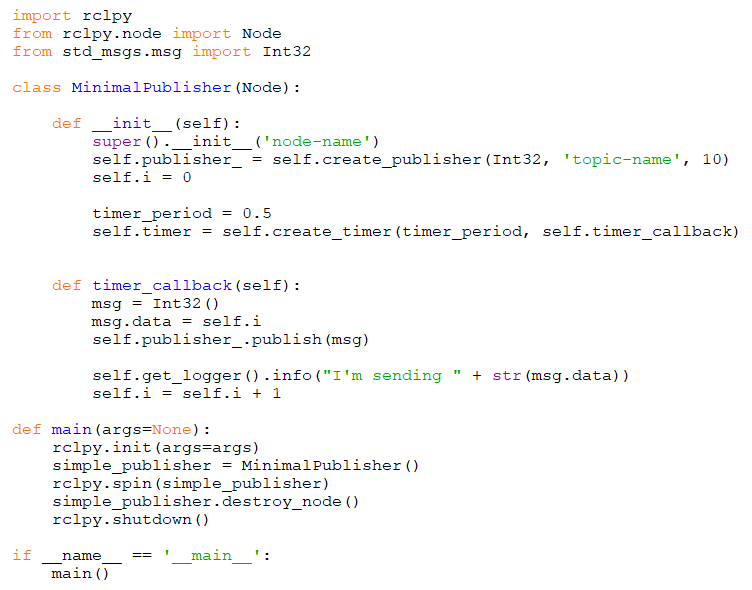
***### PATTERN 2 SCREENSHOT STARTS HERE ###***

*### PATTERN 2 SCREENSHOT ENDS HERE ###*

In the following tasks, you will be required to make videos of your simulation which will be a part of the submission. In all videos, you must show your terminal being executed. **The student names must also appear on the terminal.**

**Lab Task 4 – Turning the Robot \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [1]**

In this task, you will use your understanding of linear and angular velocities in order to publish Twist messages to the robot. Start up the simulation and spawn the robot. Next, create a new package (with rclpy, std\_msgs and geometry\_msgs dependencies) in the workspace called velocity\_package. In the package, create a node called turning.py and place the minimal publisher code:



Next, you need to make the following changes to the node:

Change all of the class and object names corresponding to the following:

**vel\_pub = VelocityPublisher()**

Add the import for the Twist messages:

**from geometry\_msgs.msg import Twist**

To initialize the twist message datatype, use the following command:

**msg = Twist()**

To enter a value for the twist message, use the following commands:

**msg.linear.x = 1.0**

**msg.angular.x = 1.0**

You need to use the above commands to change the node so that it will publish a turning command to the robot (no linear velocity). Provide the class code and also make a video (lab5\_turning) showing execution of the node in the terminal. The video must show both the terminal and gazebo.

***### TURN CODE STARTS HERE ###***

*### TURN CODE ENDS HERE ###*

**Lab Task 5 – Start-Stop \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2]**

Create a second node called redgreen.py which contains the code for publishing Twist messages that will make the robot move forward, then stop, then move forward, then stop and so on. The duration of both the moving and stopping can be 1 second each. Provide the class code and make a video (lab5\_redgreen) showing the execution.

***### START-STOP CODE STARTS HERE ###***

*### START-STOP CODE ENDS HERE ###*

**Lab Task 6 – Moving in a Circle \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [2]**

Create a third node called circles.py. Write the code for publishing Twist messages that will make the robot move in a circular path. Provide the class code and make a video (lab5\_circles) showing the execution.

***### CIRCLE CODE STARTS HERE ###***

*### CIRCLE CODE ENDS HERE ###*