University of Prince Mugrin
College of Computer and Cyber Sciences
Artificial Intelligence Department
Abubakar Waziri -4220056

AI 407 – Introduction to Robotics

Lab 5 Manual

Introduction to 3D Simulation with Gazebo & TurtleBot3

❖ **Lab objectives**
- ✓ **Overview of 3D Simulation in Gazebo**
- ✓ **TurtleBot3 Robot Worlds & Models**
- ✓ **Controlling TurtleBot3: Teleoperation & Python Scripting**
- ✓ **Reading and Processing LIDAR Data**

❖ **Lab Requirements**
- ✓ **Software:** Ubuntu 22.04 LTS, ROS 2 Humble
- ✓ **Hardware:** Students should work on Lab Devices

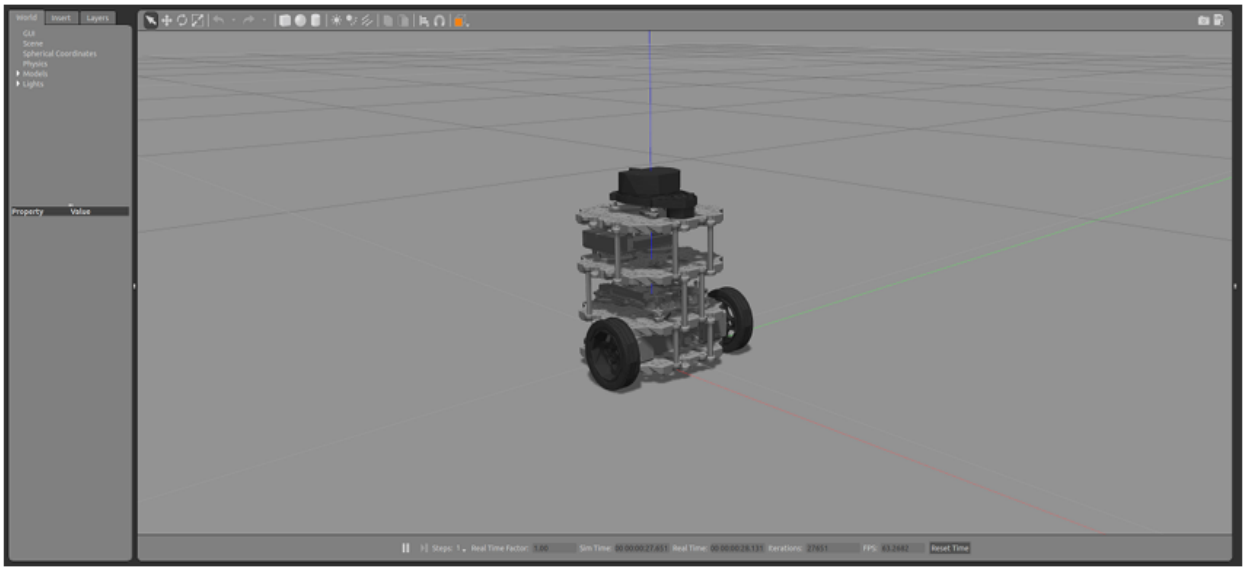
▪ **Before You Start**

Kindly read the manual, review the references if any, before beginning implementation.

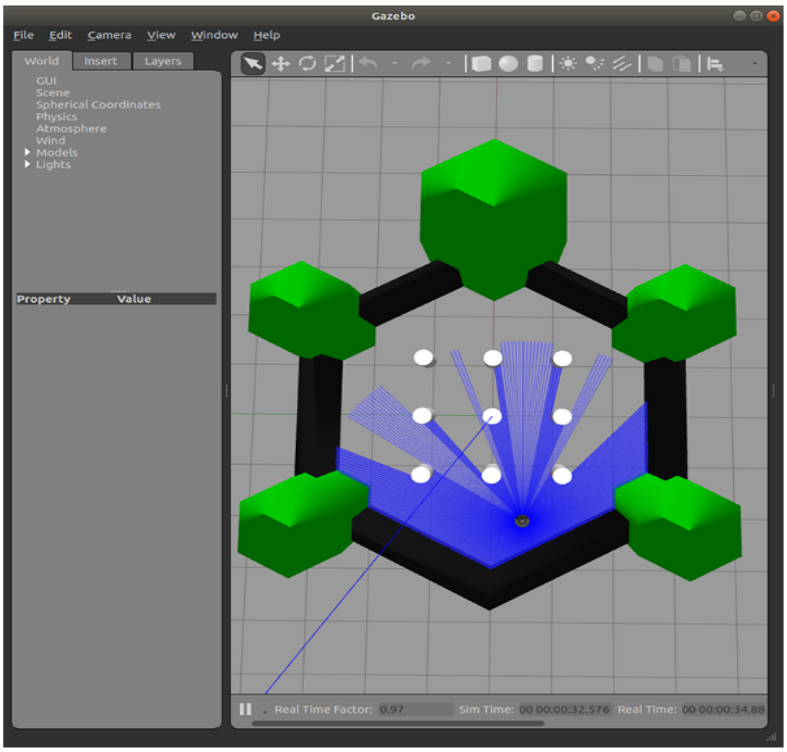
❖ Introduction to 3D Simulation with Gazebo

Gazebo is a powerful 3D simulator for testing and developing robotic applications in a realistic virtual environment. It provides physics modeling, sensor simulation, and ROS integration, enabling users to control robots like TurtleBot3 for tasks such as navigation and mapping without needing physical hardware. Three simulation environments are prepared for TurtleBot3.

1. Empty World

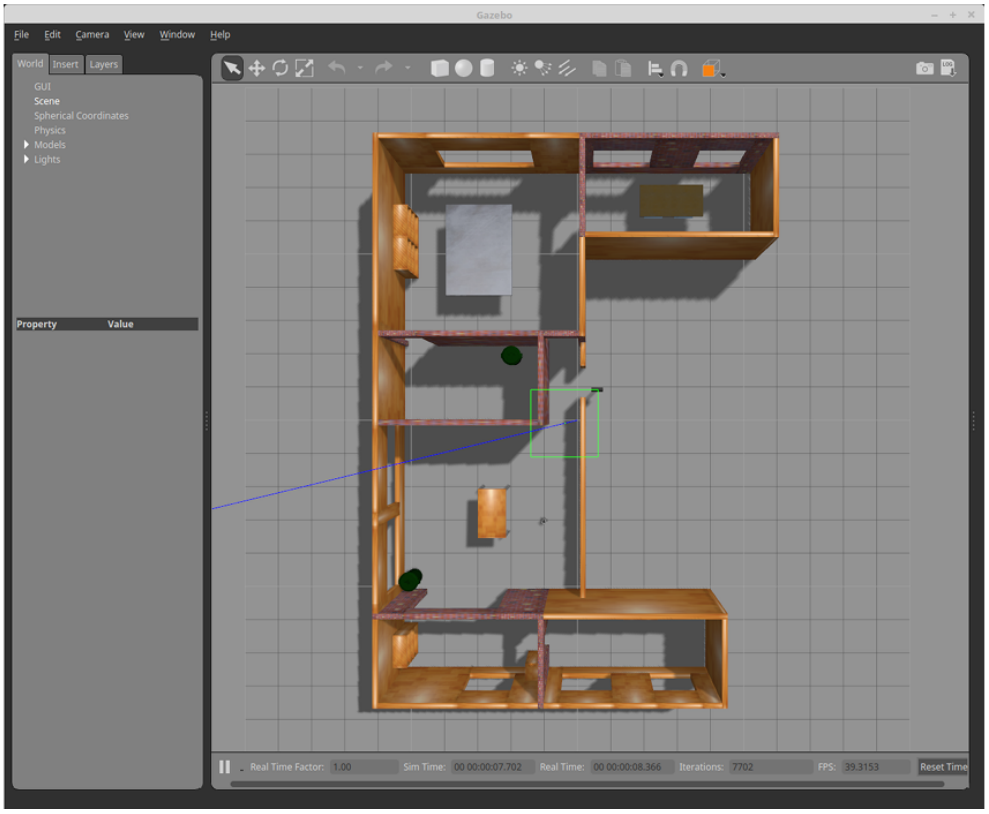


2. TurtleBot3 World



2. TurtleBot3 House

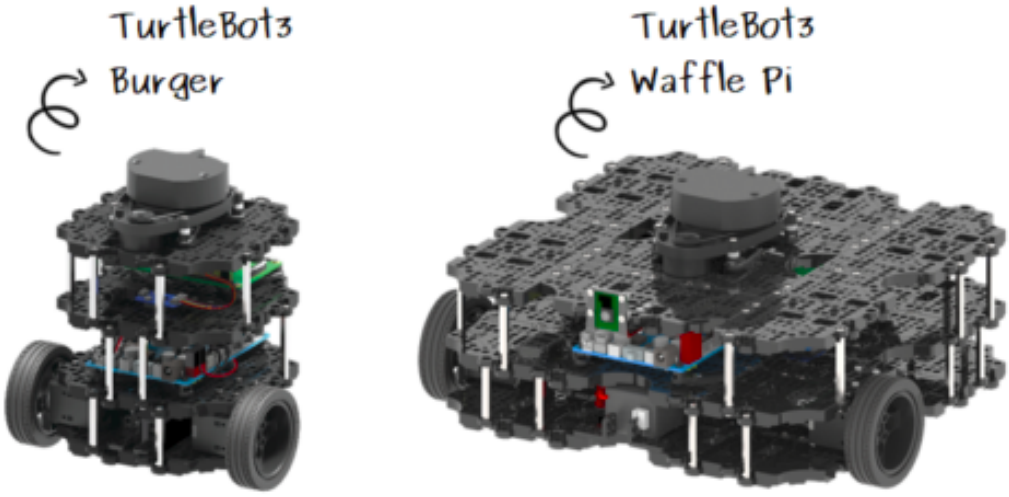
University of Prince Mugrin
College of Computer and Cyber Sciences
Artificial Intelligence Department
Abubakar Waziri -4220056



❖ Introduction to TurtleBot3

TurtleBot3 is a compact, customizable robot designed for robotics education and research, offering affordability and versatility with full ROS compatibility. Available in models like Burger, Waffle, and Waffle Pi, it features various sensors (e.g., LiDAR, cameras) and actuators, enabling tasks like mapping, navigation, and object recognition, making it ideal for students, researchers, and hobbyists.

1. **TurtleBot3 Overview:**
    a. Review the first Lab ==> Lab-1 Manual
2. **TurtleBot3 Models:**
    a. Burger, Waffle, and Waffle Pi

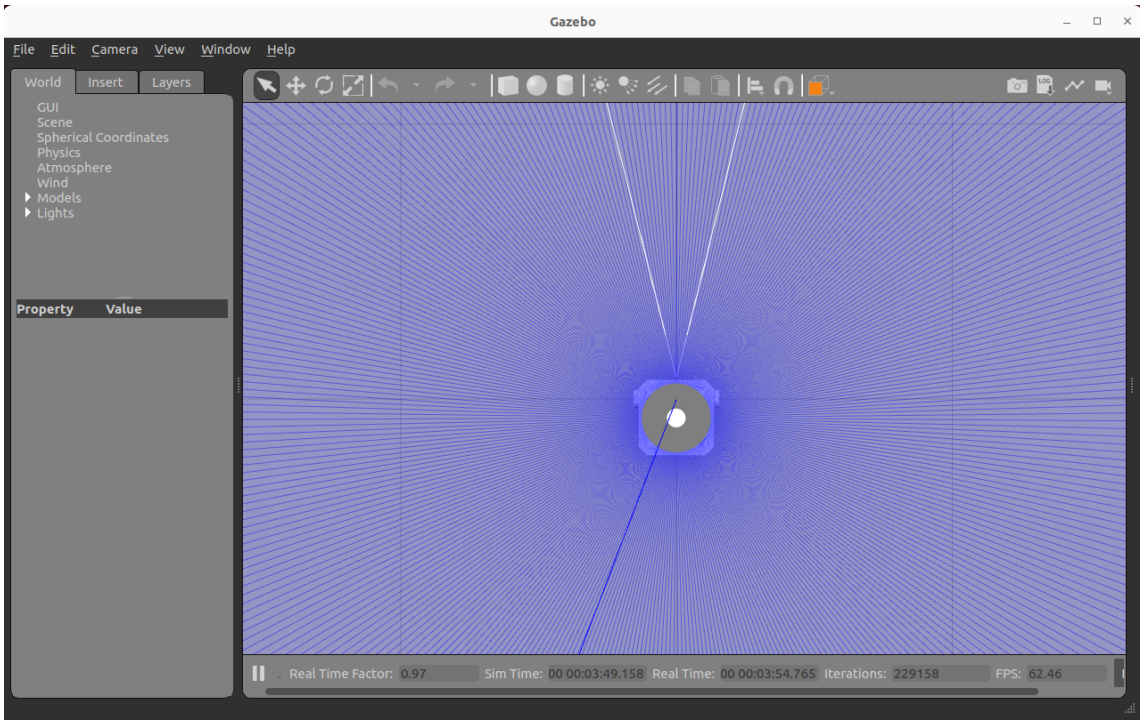


TurtleBot3's integration with ROS allows users to extensively control and interact with the robot using ROS topics, services, and actions. Here's how TurtleBot3 can be utilized to understand robotics concepts:

1. ROS Topics for Movement Control: Users can publish velocity commands to ROS topics (e.g., /cmd_vel) to control the robot's linear and angular velocities. This direct method of controlling the robot's movement is fundamental for understanding robot locomotion and the effects of different commands on the robot's trajectory.

2. Sensor Data Acquisition: By subscribing to various sensor topics (e.g., /scan for LiDAR data), users can access real-time sensor information. This data can be used for tasks like obstacle avoidance, environment mapping, and navigation, offering hands-on experience with sensor integration and data processing in robotics.

University of Prince Mugrin
College of Computer and Cyber Sciences
Artificial Intelligence Department
Abubakar Waziri -4220056

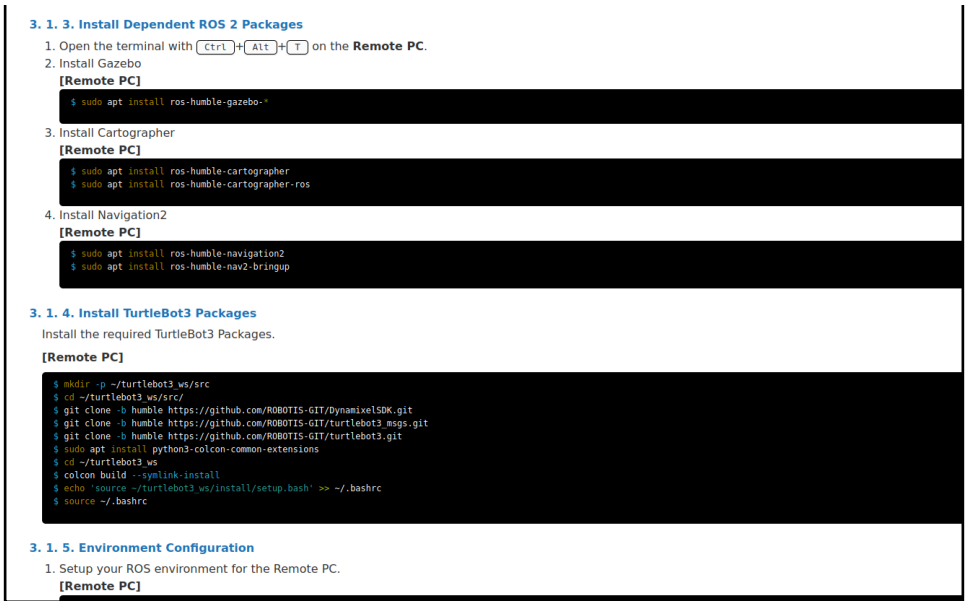


❖ Exercise 1: Getting Started with Gazebo Simulation

Step 1: Install TurtleBot3 & Gazebo Packages

Please follow the instructions in the documentation carefully, starting from Step 3.1.3 through Step 3.1.5. Copy each command from the documentation and execute them in the terminal one by one.

https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/



3. 1. 3. Install Dependent ROS 2 Packages

1. Open the terminal with Ctrl+Alt+T on the **Remote PC**.
2. Install Gazebo

[Remote PC]

```
$ sudo apt install ros-humble-gazebo-*
```

3. Install Cartographer

[Remote PC]

```
$ sudo apt install ros-humble-cartographer
$ sudo apt install ros-humble-cartographer-ros
```

4. Install Navigation2

[Remote PC]

```
$ sudo apt install ros-humble-navigation2
$ sudo apt install ros-humble-nav2-bringup
```

3. 1. 4. Install TurtleBot3 Packages

Install the required TurtleBot3 Packages.

[Remote PC]

```
$ mkdir -p ~/turtlebot3_ws/src
$ cd ~/turtlebot3_ws/src/
$ git clone -b humble https://github.com/ROBOTIS-GIT/DynamixelSDK.git
$ git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
$ git clone -b humble https://github.com/ROBOTIS-GIT/turtlebot3.git
$ sudo apt install python3-colcon-common-extensions
$ cd ~/turtlebot3_ws
$ colcon build --symlink-install
$ echo 'source ~/turtlebot3_ws/install/setup.bash' >> ~/.bashrc
$ source ~/.bashrc
```

3. 1. 5. Environment Configuration

1. Setup your ROS environment for the Remote PC.

[Remote PC]

Step 2: **Install Gazebo Simulation Package**

Please follow the instructions in the documentation carefully, starting from Step 6.1.1 through Step 6.1.2. Copy each command from the documentation and execute them in the terminal one by one.

https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#gazebo-simulation

Make sure to source this workspace in the .bashrc file. To do that open a new terminal and write the command

gedit .bashrc

Once the file open go and add the following at the end:

source ~/turtlebot3_ws/install/setup.bash
export ROS_DOMAIN_ID=30 #TURTLEBOT3
source /usr/share/gazebo/setup.sh
export TURTLEBOT3_MODEL=waffle_pi

Step 3: **Launch Simulation World (Terminal #1)**

export TURTLEBOT3_MODEL=waffle_pi

ros2 launch turtlebot3_gazebo empty_world.launch.py

Step 4: **Investigate Gazebo Topic and Nodes (Terminal #2)**

To understand how Gazebo communicates with ROS 2, use the following commands:

ros2 topic list

University of Prince Mugrin
College of Computer and Cyber Sciences
Artificial Intelligence Department
Abubakar Waziri -4220056

ros2 node list

Step 5: Operate TurtleBot3 (Terminal #2)

To teleoperate the TurtleBot3 with a keyboard, launch the teleoperation node with the command below in a new terminal window.

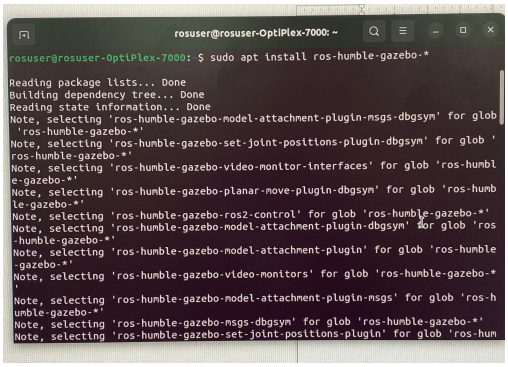
export TURTLEBOT3_MODEL=waffle_pi

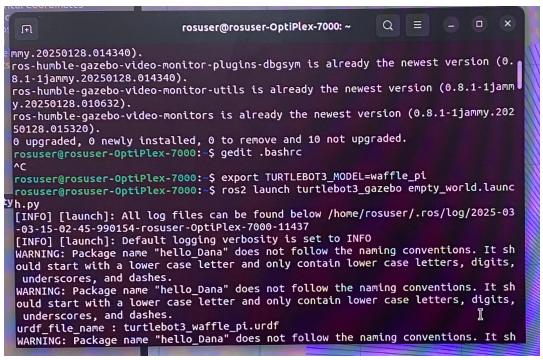
ros2 run turtlebot3_teleop teleop_keyboard

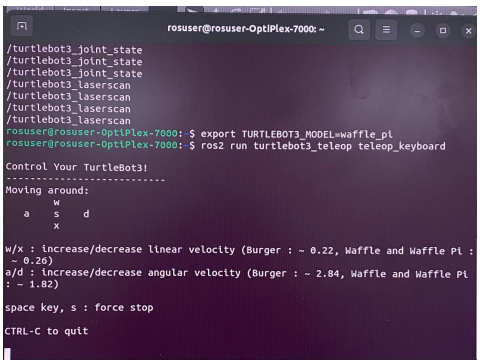
Step 5: Subscribe to the /cmd_vel topic manually (Terminal #3)

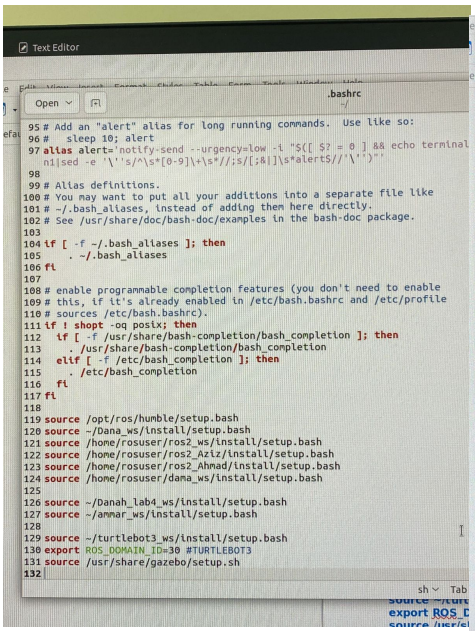
ros2 topic echo /cmd_vel

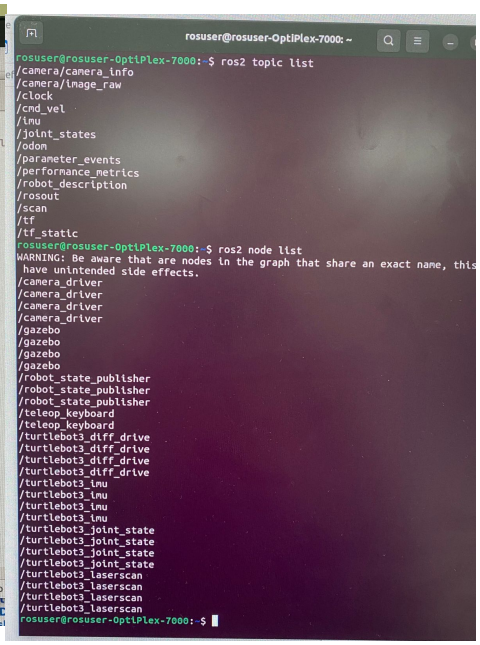
Put screenshots of your work here (for ALL the steps):

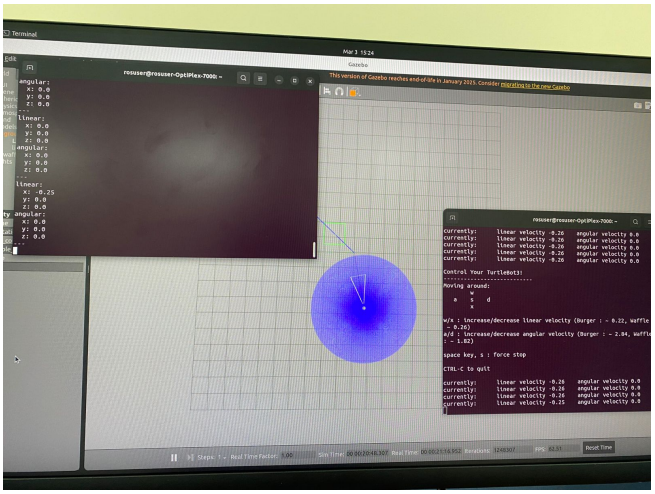












❖ Exercise 2: Controlling Turtlebot3 with Python

<u>Step 1:</u> Understanding the difference between the topics /scan, and /cmd_vel

LaserScan (/scan) → Reads LIDAR data (distances to obstacles).

Twist (/cmd_vel) → Controls robot motion (speed & direction).

- **LaserScan Message (LIDAR Data)**

The sensor_msgs/msg/LaserScan message is used to represent LIDAR data in ROS 2. When you run ros2 topic echo /scan, you get data like this:

```yaml
header:
  stamp:
    sec: 196
    nanosec: 191000000
  frame_id: base_scan
angle_min: 0.0
angle_max: 6.28
angle_increment: 0.0175
time_increment: 0.0
scan_time: 0.0
range_min: 0.12
range_max: 3.5
ranges: [inf, inf, 2.1, 1.8, 1.5, 1.2, 0.8, 0.5, 0.3, ...]
intensities: []  # Some LIDARs provide intensity readings
```

- angle_min & angle_max: Define the LIDAR's field of view (e.g., 0 to 360° or 0 to ~6.28 radians).
- angle_increment: The angular resolution (small steps between readings).
- range_min & range_max: The LIDAR's valid sensing range (e.g., 0.12m to 3.5m).
- ranges: An array of distance values, where each value corresponds to an angle. inf means no object detected within range.

This LaserScan message helps robots detect obstacles, create maps (SLAM), and navigate safely.

➢ **Twist Message (Used for Robot Motion)**

The **geometry_msgs/msg/Twist** message is used to control a robot's movement, not for LIDAR data. It has two main parts:

```
Vector3 linear:    float64 x     # Movement along the x-axis
                   float64 y     # Movement along the y-axis
                   float64 z     # Movement along the z-axis

Vector3 angular:   float64 x     # Rotation around the x-axis
                   float64 y     # Rotation around the y-axis
                   float64 z     # Rotation around the z-axis
```

Step 2: Creating a python script to control TurtleBot3 (publish to /cmd_vel)h

In the workspace you created in the last lab (name_lab4_ws) and inside the same package (my_turtle_pkg), create a new Python script to control the TurtleBot3, name it (move_waffle.py).

Add the following code inside the script, then follow the same steps from the previous lab to execute it using ROS 2.

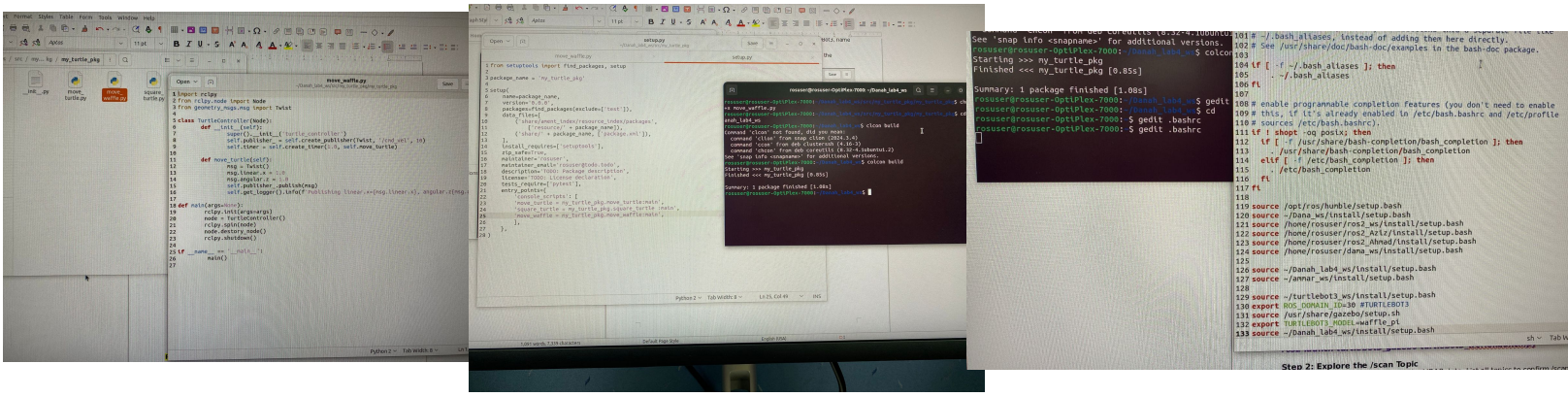
Open

move_waffle.py
~/zainab_lab4_ws/src/my_turtle_pkg/my_turtle_pkg

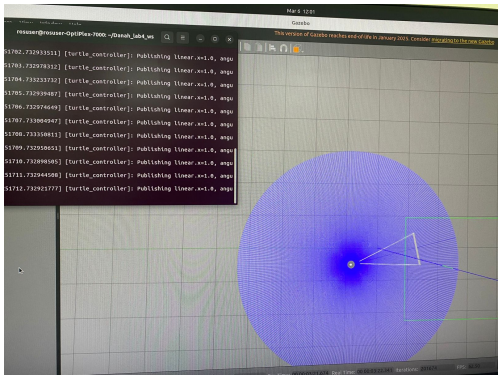
Save

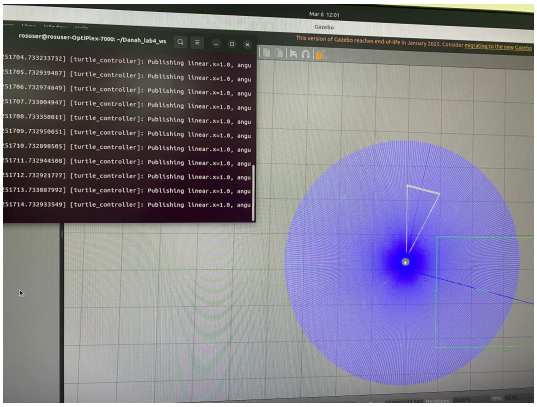
```python
import rclpy
from rclpy.node import Node
from geometry_msgs.msg import Twist

class TurtleController(Node):
        def __init__(self):
                super().__init__('turtle_controller')
                self.publisher_ = self.create_publisher(Twist, '/cmd_vel', 10)
                self.timer = self.create_timer(1.0, self.move_turtle)

        def move_turtle(self):
                msg = Twist()
                msg.linear.x = 1.0 # Move forward
                msg.angular.z = 1.0 # Rotate Left
                self.publisher_.publish(msg)
                self.get_logger().info(f'Publishing linear.x={msg.linear.x},
angular.z={msg.angular.z}')

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

if __name__ == '__main__':
        main()
```

Put screenshots of your work here (for ALL the steps):



University of Prince Mugrin
College of Computer and Cyber Sciences
Artificial Intelligence Department
Abubakar Waziri -4220056





❖ Exercise 3: Subscribing to the /scan Topic and Processing LIDAR Data

The objective is to learn how to **subscribe to the /scan topic**, read LIDAR data from TurtleBot3, and process distance measurements to detect nearby obstacles.

Step 1: Open a terminal and launch Gazebo with the TurtleBot3 world:
ros2 launch turtlebot3_gazebo turtlebot3_world.launch.py

Step 2: Explore the /scan Topic

a. Before writing code, inspect the LIDAR data, List all topics to confirm /scan exists:
   ros2 topic list
b. Echo LIDAR data to observe raw messages:
   ros2 topic echo /scan
   Look for ranges which contain distance values in different directions.

Step 3: Write a Python Script to Subscribe to LIDAR Data

Inside the same package (my_turtle_pkg), create a new Python script to subscribe to LiDAR data, name it (lidar_subscriber.py). Add the following code inside the script, then follow the same steps from the previous lab to execute it using ROS 2.

Open *lidar_subscriber.py ~/zainab_lab4_ws/src/my_turtle_pkg/my_turtle_pkg Save

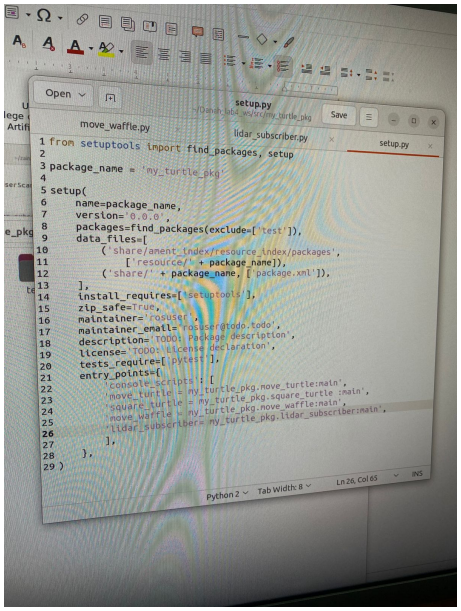
```python
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import LaserScan

class LidarSubscriber(Node):
    def __init__(self):
        super().__init__('lidar_subscriber')
        self.subscription = self.create_subscription(
            LaserScan,
            '/scan',
            self.lidar_callback,
            10)
        self.subscription

    def lidar_callback(self, msg):
        min_distance = min(msg.ranges)
        self.get_logger().info(f'Minimum Distance: {min_distance:.2f} meters')

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

if __name__ == '__main__':
    main()

```

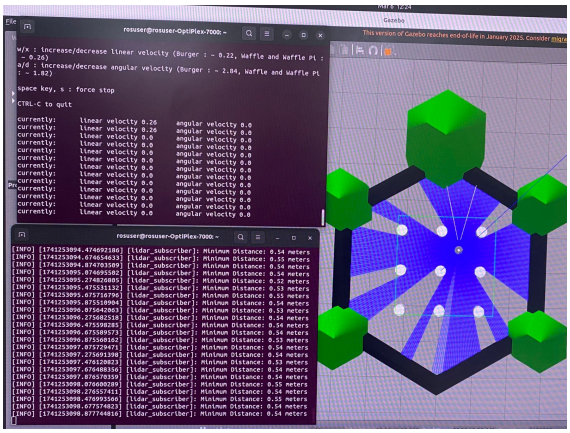
Step 4: Run the LiDAR subscriber:

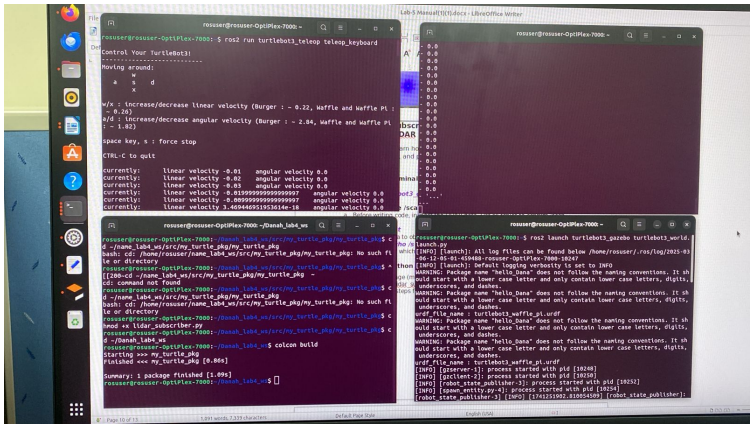
ros2 run my_turtle_pkg lidar_subscriber

Put screenshots of your work here (for ALL the steps):

University of Prince Mugrin
College of Computer and Cyber Sciences
Artificial Intelligence Department
Abubakar Waziri -4220056







❖ Lab activity:

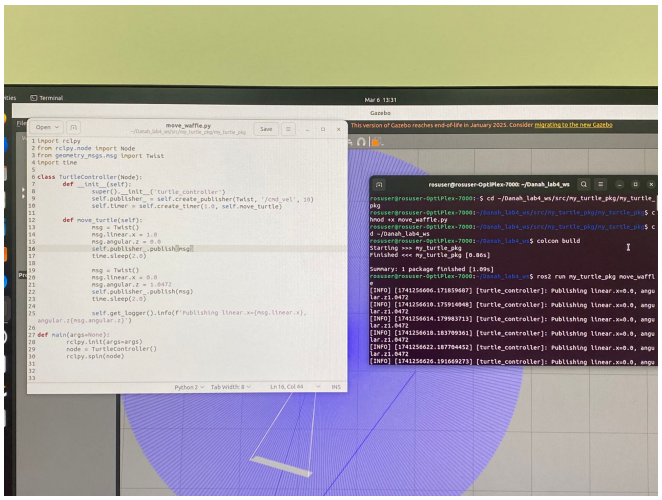
Task1: Write a Python script to move the turtlebot3 in a triangle path inside the Gazebo:

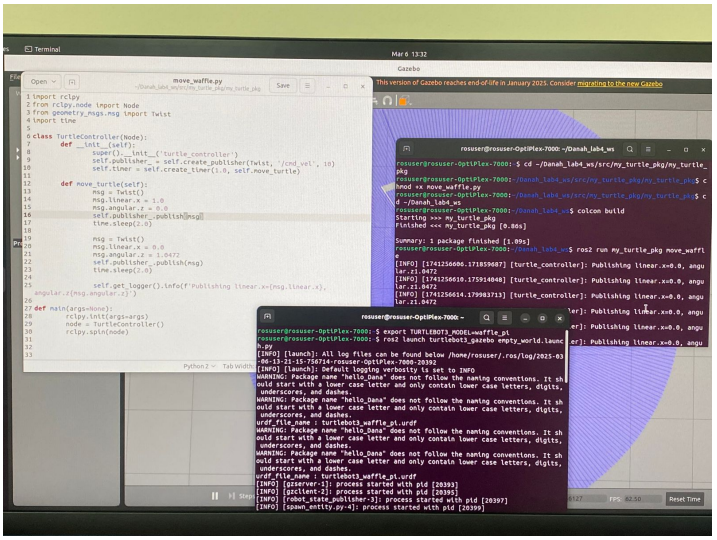
1. Provide the source code:

2. Explain what you did to make it compatible with TurtleBot3:

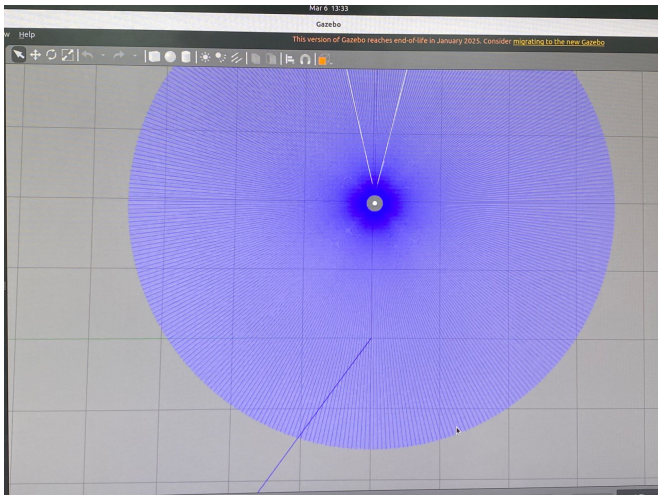
add time sleep and change the values for x and z

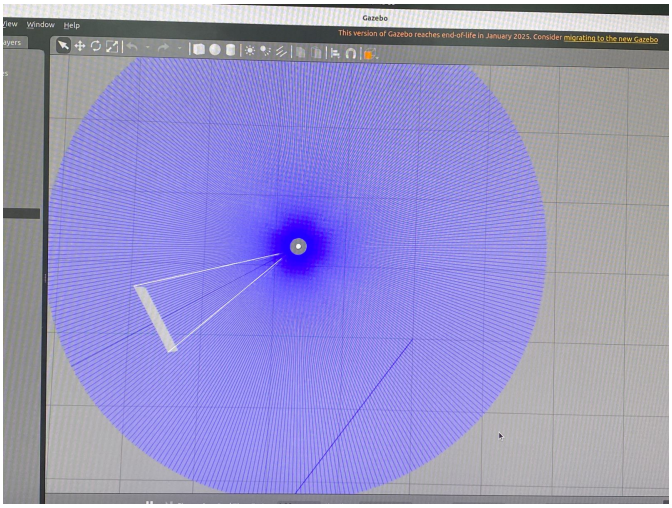
3. Provide screenshot of the terminal that's running the code and the turtlebot3 gazebo window:

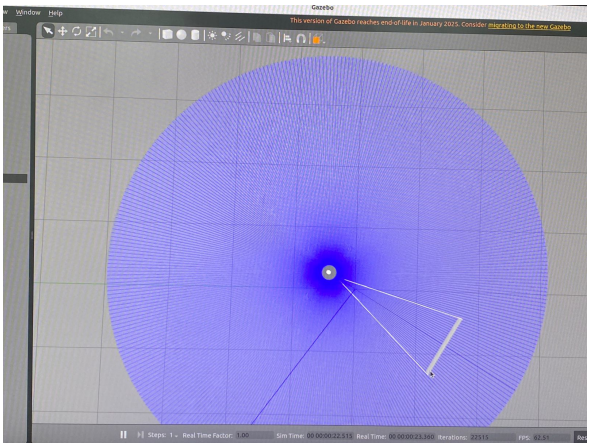




University of Prince Mugrin
College of Computer and Cyber Sciences
Artificial Intelligence Department
Abubakar Waziri -4220056







References

https://emanual.robotis.com/docs/en/platform/turtlebot3/simulation/#gazebo-simulation