

*AI 407 – Introduction to Robotics*

**Lab 4 Manual**

**Exploring Robotics Fundamentals with Turtlesim**

* **Lab objectives**
* **Introduction to Turtlesim**
* **Exploring ROS 2 commands with Turtlesim**
* **Controlling Turtlesim with Python**
* **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 Turtlesim**

Turtlesim is a lightweight 2D simulator included in ROS 2, designed for learning robotics and ROS fundamentals. It provides a simple turtle-like robot that moves in response to commands, making it ideal for beginners to explore ROS concepts like topics, services, and parameters.



**Key learning aspects of Turtlesim:**

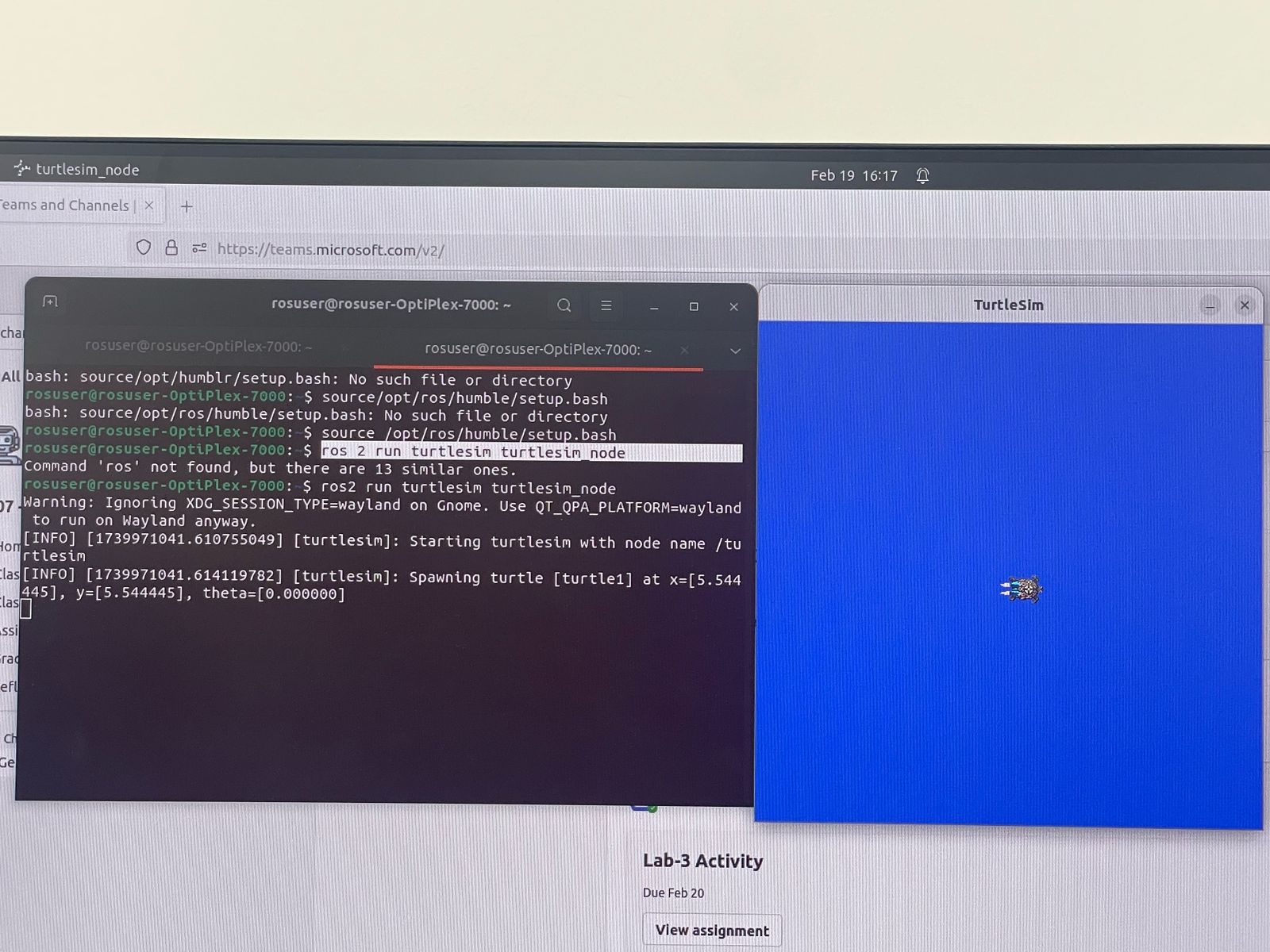
1. **ROS Topics & Messaging:** Users control the turtle by publishing velocity commands to the topic /turtle1/cmd\_vel and monitor its position via the topic /turtle1/pose, introducing the publish/subscribe model.
2. **Basic Robotics Principles:** Experimenting with linear and angular velocity helps understand motion and forward kinematics.
3. **Control Algorithms:** Users can implement ROS nodes to move the turtle in patterns like circles or squares, gaining hands-on experience with robot control logic.

* **Exercise 1: Exploring Turtlesim**

**Step 1: Source ROS 2 Environment**

Before running any ROS 2 commands, source the setup file:

*source /opt/ros/humble/setup.bash*

**

**Step 2: Run the Turtlesim node**

*ros2 run turtlesim turtlesim\_node*

*صورة تحتوي على نص, الإلكترونيات, لقطة شاشة, برمجيات

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.*

**Step 3: Understanding ROS 2 Topics for Turtlesim**

Key topics for controlling the turtle:

* **/turtle1/cmd\_vel** → Publishes velocity commands (geometry\_msgs/msg/Twist).
* **/turtle1/pose** → Provides the turtle’s current position and orientation (turtlesim/msg/Pose).
  1. To list all active topics:

*ros2 topic list*

* 1. Check the Message Type:

*ros2 topic info /turtle1/cmd\_vel*

This will show you the type of the message being used (which is geometry\_msgs/msg/Twist).

* 1. Check the Message Structure:

*ros2 interface show geometry\_msgs/msg/Twist*

This command displays the structure of the Twist message, which consists of linear and angular components (both of type Vector3).

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 4: Controlling Turtle Movement**

To move the turtle, publish a message to /turtle1/cmd\_vel. The message type is geometry\_msgs/msg/Twist, which contains linear and angular velocity components.

* **Controlling the Turtle Using the Keyboard:**

To control the turtle using the keyboard, run the following command in a new terminal:

*ros2 run turtlesim turtle\_teleop\_key*

* This command starts a teleoperation node that listens for keyboard inputs and publishes velocity commands to /turtle1/cmd\_vel.
* Use the arrow keys, or the keys written in the terminal to move the turtle:
* **Publishing Velocity Commands Manually**
  1. **Example: Move the Turtle Forward**

In a new terminal, publish a velocity command:

*ros2 topic pub /turtle1/cmd\_vel geometry\_msgs/msg/Twist ‘{linear: {x: 2.0, y: 0.0, z: 0.0}, angular: {x: 0.0, y: 0.0, z: 0.0}}’ --once*

This makes the turtle move forward with a speed of 2.0 m/s.

* 1. **Example: Rotate the Turtle**

To rotate the turtle to the left, publish a message with a positive angular velocity in the z-axis:

*ros2 topic pub /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: 1.5}}' --once*

This makes the turtle rotate counterclockwise at an angular velocity of 1.5 rad/s.

**Step 5: Monitor Turtle Pose**

To track the turtle’s position & orientation in real time, subscribe to /turtle1/pose topic using:

*ros2 topic echo /turtle1/pose*

This command displays the turtle’s x, y position, orientation, and velocity as it moves.

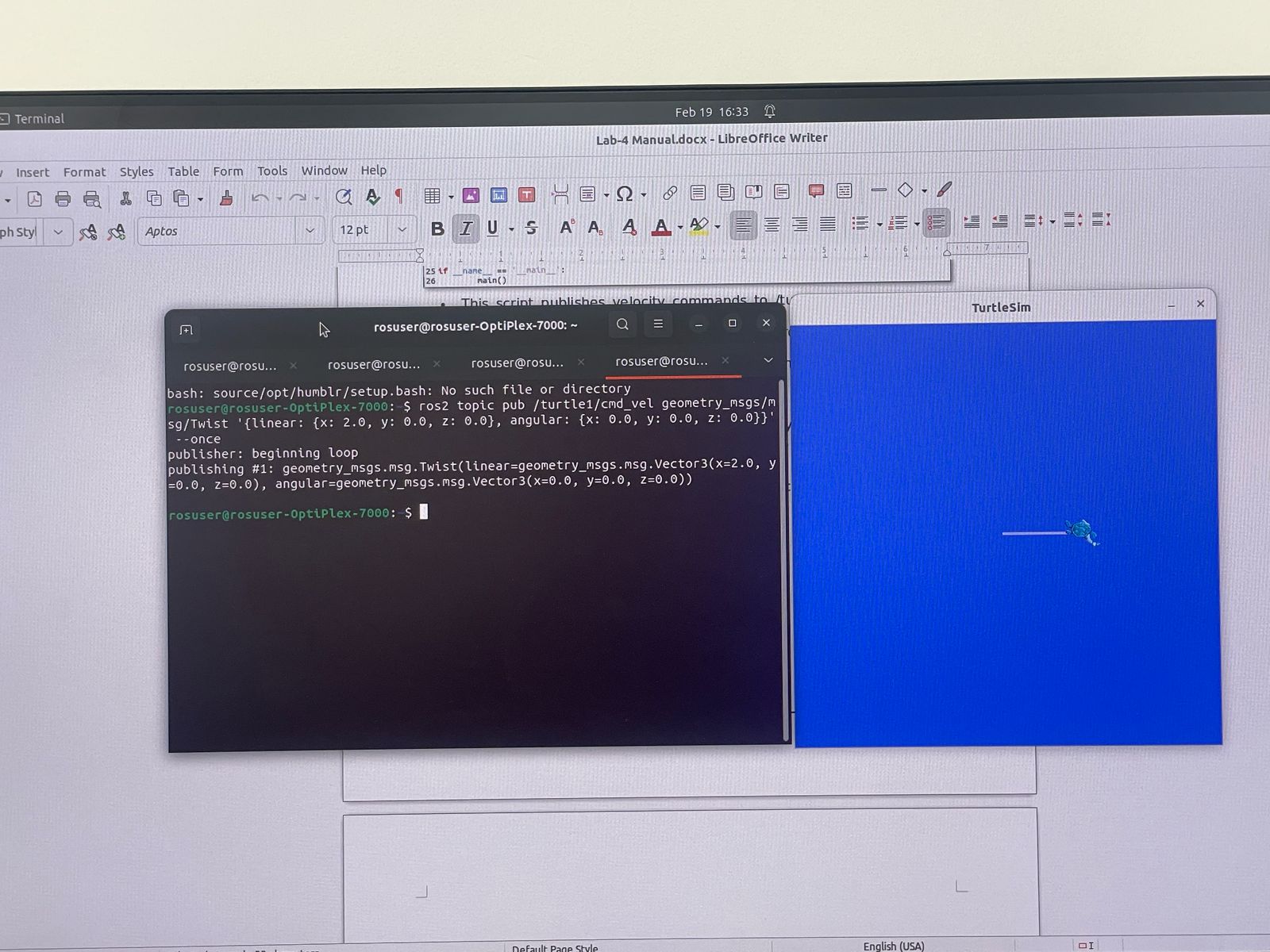
**Put screenshots of your work here:**

صورة تحتوي على نص, لقطة شاشة, برمجيات, وسائط متعددة

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.

صورة تحتوي على نص, الإلكترونيات, عرض, برمجيات

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.



* **Exercise 2: Controlling Turtlesim with Python**

In this exercise, you'll create a ROS 2 package to control the turtle using a Python script. You'll learn how to publish velocity commands programmatically instead of using terminal commands.

**Step 1: Create a New Workspace**

First, create a new workspace with your name (e.g., name\_lab4\_ws):

*mkdir -p ~/name\_lab4\_ws/src*

**Step 2: Create a New Package**

First, navigate to your ROS 2 workspace (e.g., name\_lab4\_ws) and create a package:

*cd ~/name\_lab4\_ws/src*

*ros2 pkg create --build-type ament\_python my\_turtle\_pkg --dependencies rclpy geometry\_msgs*

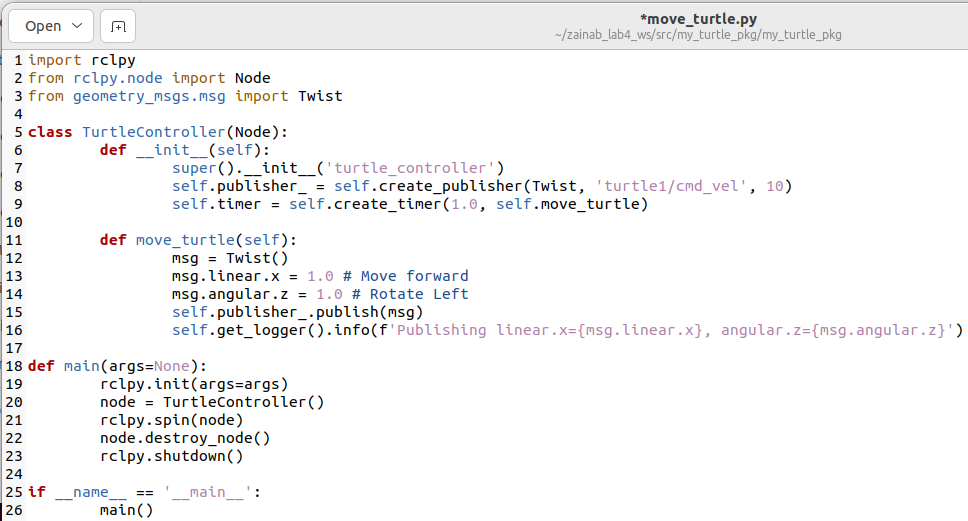
This creates a package named my\_turtle\_pkg inside the src folder.

**Step 3: Write the Python Script**

Navigate to the package directory and create a move\_turtle.py script:

*cd ~/name\_lab4\_ws/src/my\_turtle\_pkg/my\_turtle\_pkg*

*gedit move\_turtle.py*



* This script publishes velocity commands to /turtle1/cmd\_vel every 1 second.
* The turtle moves forward (linear.x = 1.0) and rotates left (angular.z = 1.0).
* The node logs the published values for debugging.

After you write the code, make it executable

*chmod +x move\_turtle.py*

**Step 4: Modify setup.py to Include the Script**

Open setup.py inside my\_turtle\_pkg and add the entry point:

'move\_turtle = my\_turtle\_pkg.move\_turtle:main',

**Step 5: Build the Package**

* Return to your workspace root and build the package:

*cd ~/name\_lab4\_ws*

*colcon build*

* Source the workspace in the. bashrc file:

source ~/name\_lab4\_ws/install/setup.bash

**Step 6: Run the Script**

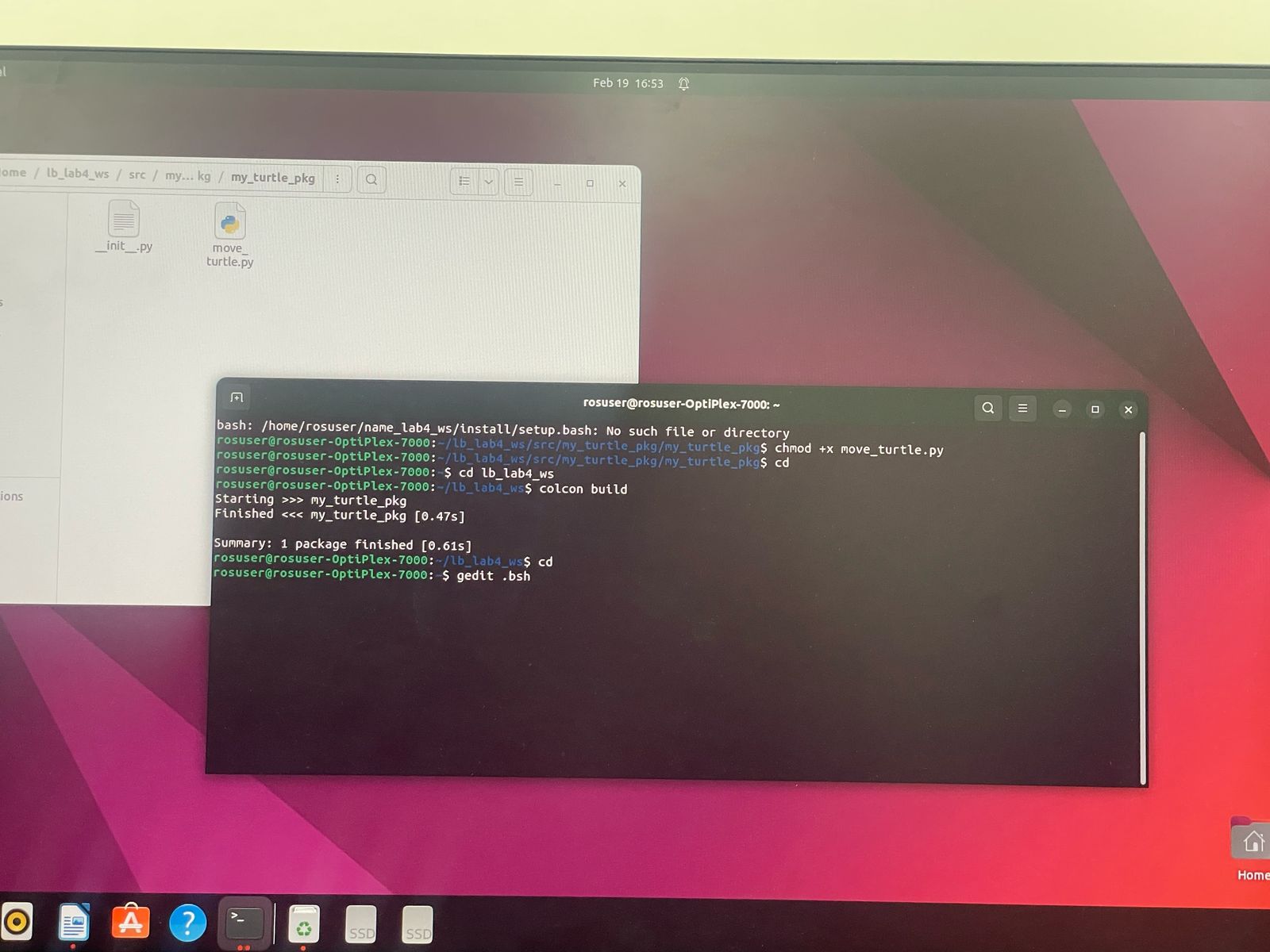
First Run the turtlesim node

*ros2 run turtlesim turtlesim\_node*

Then, in another terminal, run your Python script:

*ros2 run my\_turtle\_pkg move\_turtle*

**Put screenshots of your work here:**

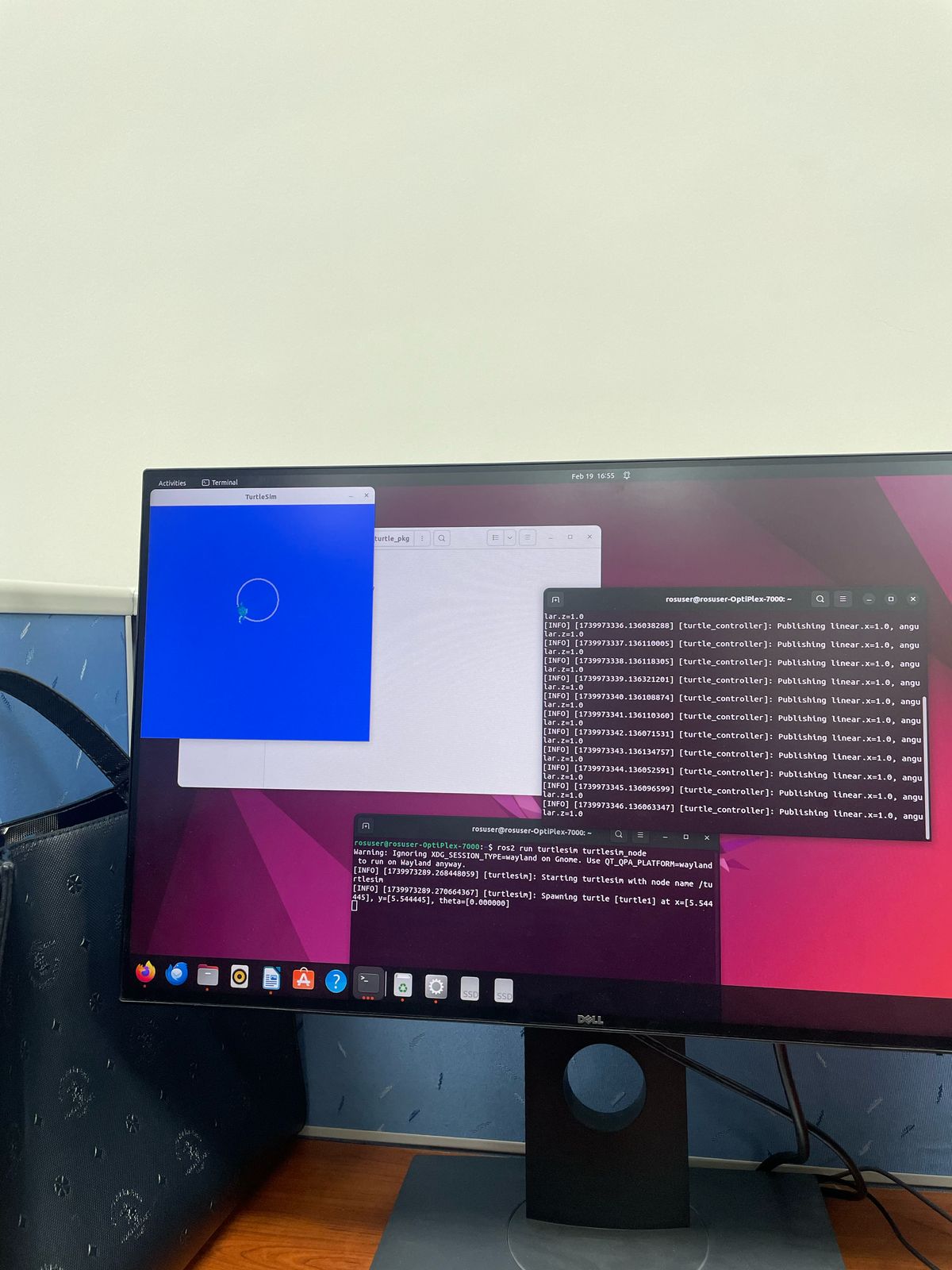
****

**صورة تحتوي على نص, قائمة طعام, لقطة شاشة

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.**

**صورة تحتوي على نص, الإلكترونيات, لقطة شاشة, برمجيات

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.**

****

* **Lab activity:**

**Task:** Write a Python script to move the turtle in a square path like this:

A screenshot of a computer

AI-generated content may be incorrect.

1. Provide the source code:

import rclpy

from rclpy.node import Node

from geometry\_msgs.msg import Twist

import time

class TurtleController(Node):

def \_\_init\_\_(self):

super().\_\_init\_\_('turtle\_controller')

self.publisher = self.create\_publisher(Twist, 'turtle1/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 = 0.0 # Rotate left

self.publisher.publish(msg)

self.get\_logger().info(f'Publishing linear x={msg.linear.x}, angular z={msg.angular.z}')

time.sleep(1)

msg.linear.x = 0.0

msg.angular.z = 1.57 # Rotate left

self.publisher.publish(msg)

time.sleep(1) # Correct indentation

def main(args=None):

rclpy.init(args=args)

node = TurtleController()

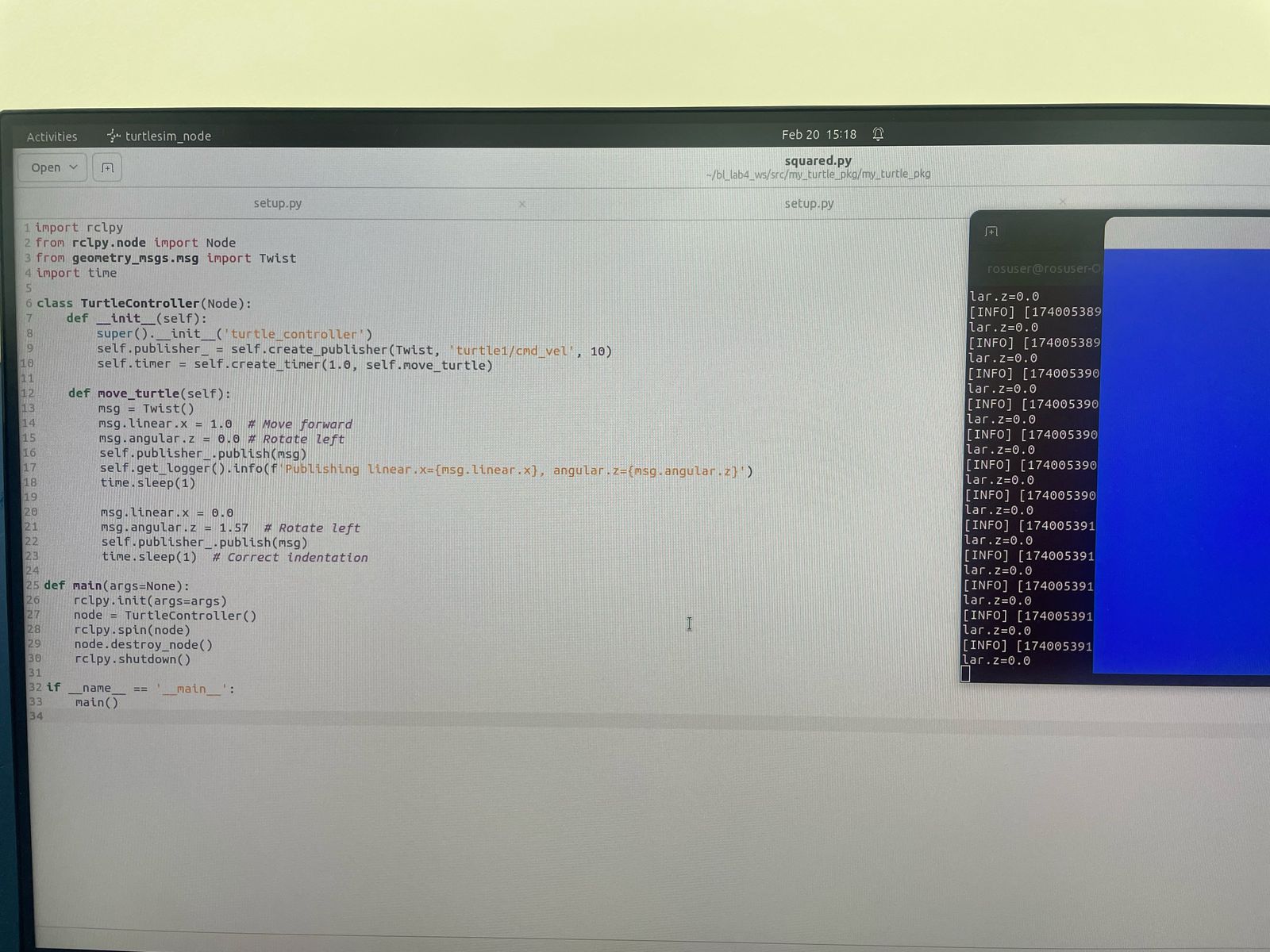
rclpy.spin(node)

node.destroy\_node()

rclpy.shutdown()

if \_\_name\_\_ == '\_\_main\_\_':

main()

1. 
2. Explain what you did to make it move in square shape:

Just move forward and set the rotation to 0 and sleep so it will make a line

Then set the line to be 0 and only rotate by 90 degrees

3. Provide screenshot of the terminal that’s running the code and the Turtlesim window:

**صورة تحتوي على نص, جهاز إخراج, الحاسوب, جهاز إلكتروني

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.**

**References**

[Creating a workspace — ROS 2 Documentation: Humble documentation](https://docs.ros.org/en/humble/Tutorials/Beginner-Client-Libraries/Creating-A-Workspace/Creating-A-Workspace.html)