

CSE461 Lab Report 04

Fall 23

**Group: 01**

**Title**

Generation of Rectangle & Spiral Path Using Turtle Movement Controlled by Python Script & ROS Packages

***Prepared by,***

**Name:** Anika Islam

**ID:** 21101298

**Section:** 09

**Procedure:**

* **Task 1:**

1. ROS and Turtlebot packages are installed in a virtual machine.
2. ***roscore*** is run in a terminal to boot the master node.
3. ***rosrun turtlesim turtlesim\_node*** is run in a terminal to pop up a window with a turtle at rest.
4. In a different terminal, ***python3 rectangle.py*** command is run to start the simulation.
5. User input is asked in the terminal using the code in the **rectangle.py** file for the speed of the turtle, width, and height of the rectangle (distance moved by the turtle to generate the rectangle).
6. Turtle moves from down to up using the height, left to right using the width, from up to down using the height, and from right to left using the width. Using this process, the turtle generates a complete rectangle.

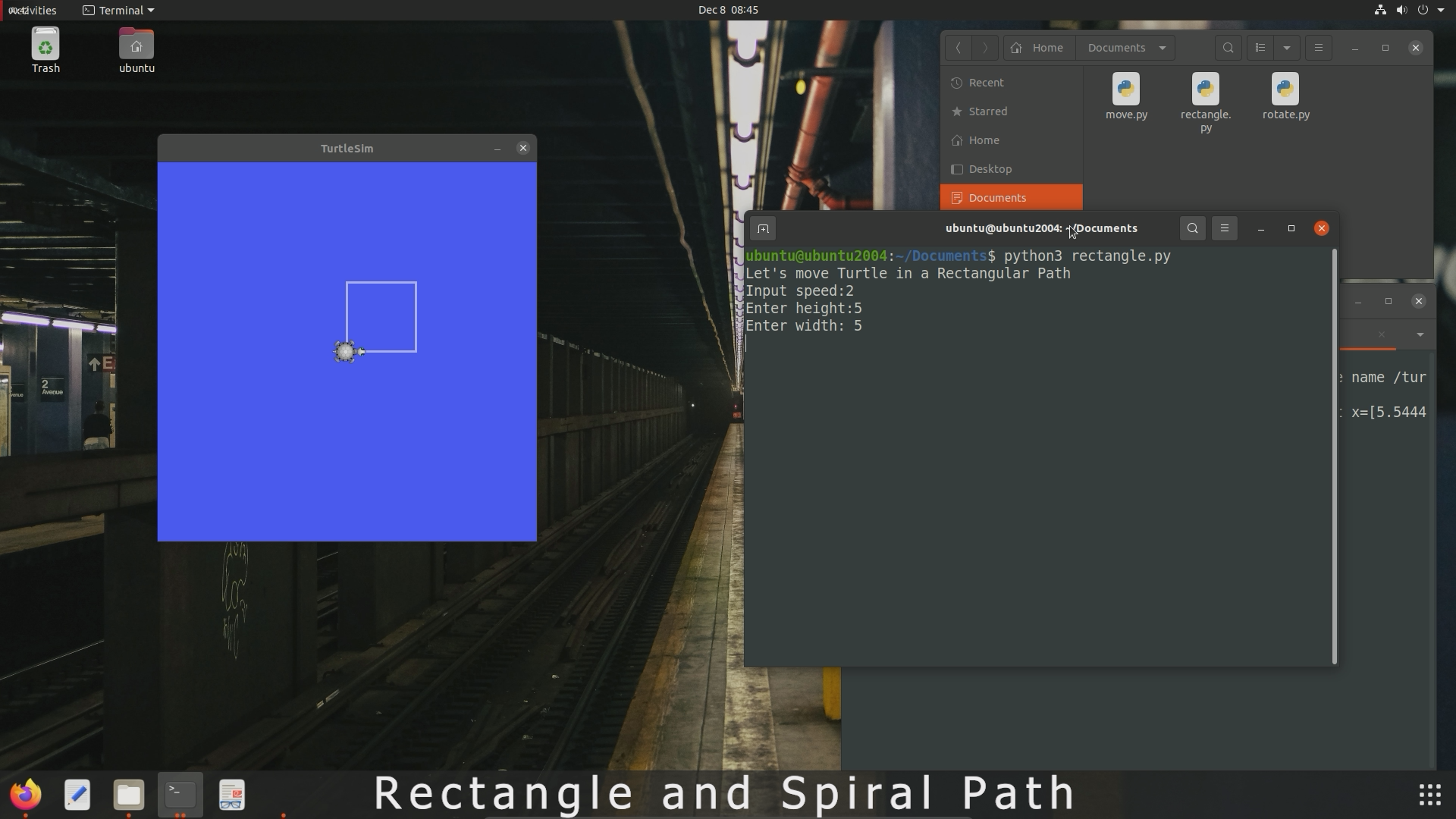
* **Task 2:**

1. The same process for task 1 (1 to 3) is repeated here.
2. In a different terminal, ***python3 rotate.py*** command is run to start the simulation.
3. User input is asked in the terminal using the code in the **rotate.py** file for the speed of the turtle and the number of rotations (the radius of the spiral).
4. A spiral is generated using the movement of the turtlebot.

**Simulation Image:**

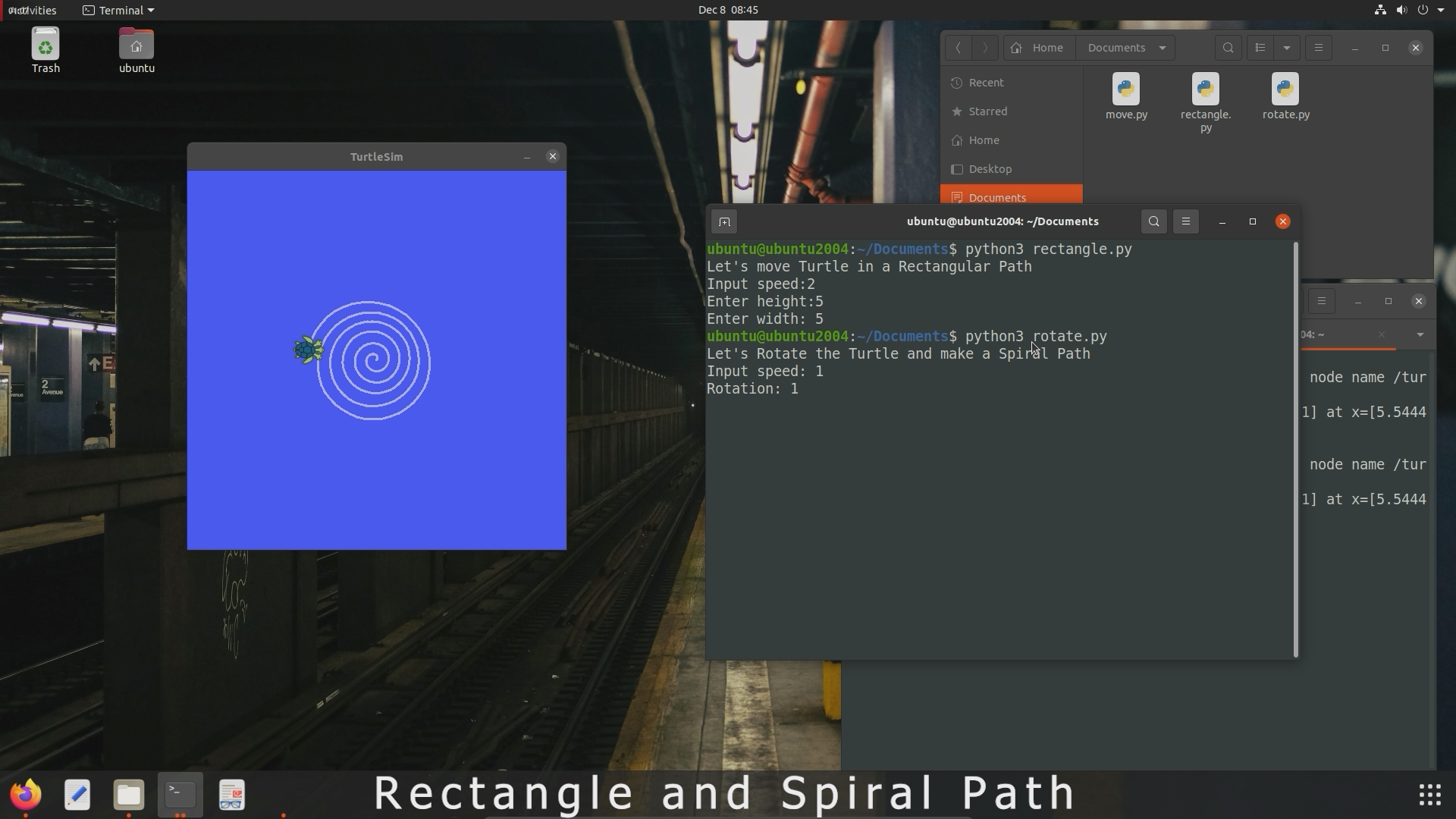
* **Task 1:**

Move the turtle to create a Rectangular Path.

****

* **Task 2:**

Move the turtle in an outward spiral path.

****

**Code:**

* **Task 1: (rectangle.py)**

| *#!/usr/bin/python3* import rospy from geometry\_msgs.msg import Twist     def **rectangle**():  *# Starts a new node*  rospy.init\_node('robot\_cleaner', anonymous=True)  velocity\_publisher = rospy.Publisher('/turtle1/cmd\_vel', Twist, queue\_size=10)  vel\_msg = Twist()      *# user input*  print("Let's move Turtle in a Rectangular Path")  speed = float(input("Input speed:"))  height = float(input("Enter height:"))  width = float(input("Enter width: "))      *# Now the movement of the Turtle(U,R,D,L)*  *# up movement*  vel\_msg.linear.y = abs(speed)  velocity\_publisher.publish(vel\_msg)  rospy.sleep(height / speed)  *# stop*  vel\_msg.linear.y = 0  velocity\_publisher.publish(vel\_msg)     *# right movement*  vel\_msg.linear.x = abs(speed)  velocity\_publisher.publish(vel\_msg)  rospy.sleep(width/speed)  *# stop*  vel\_msg.linear.x = 0  velocity\_publisher.publish(vel\_msg)    *# down movement*  vel\_msg.linear.y = -abs(speed)  velocity\_publisher.publish(vel\_msg)  rospy.sleep(height/speed)  *# stop*  vel\_msg.linear.y = 0  velocity\_publisher.publish(vel\_msg)    *# left movement*  vel\_msg.linear.x = -abs(speed)  velocity\_publisher.publish(vel\_msg)  rospy.sleep(width/speed)  *# stop*  vel\_msg.linear.x = 0  velocity\_publisher.publish(vel\_msg)   *# Driver Class Code*  if \_\_name\_\_ == '\_\_main\_\_':  try:  *#Testing our function*  rectangle()  except rospy.ROSInterruptException: pass |
| --- |

* **Task 2: (rotate.py)**

| *#!/usr/bin/python3* import rospy from geometry\_msgs.msg import Twist import math   def **rotate**():  *# Starts a new node*  rospy.init\_node('robot\_cleaner', anonymous=True)  velocity\_publisher = rospy.Publisher('/turtle1/cmd\_vel', Twist, queue\_size=10)  vel\_msg = Twist()    *# user input*  print("Let's Rotate the Turtle and make a Spiral Path")  speed = float(input("Input speed: "))  rotation = float(input("Rotation: "))  rotate = int(rotation)   *# Now the main rotation*  init = 0.5  for i in range(rotate\*360):  angle = i\*0.1  x = init\*angle\*math.cos(angle)  y = init\*angle\*math.sin(angle)    vel\_msg.linear.x = x  vel\_msg.linear.y = y  velocity\_publisher.publish(vel\_msg)  rospy.sleep(0.01)    *# after finish*  vel\_msg.linear.x = 0  vel\_msg.linear.y = 0  velocity\_publisher.publish(vel\_msg)  *# Driver Class Code*  if \_\_name\_\_ == '\_\_main\_\_':  try:  *#Testing our function*  rotate()  except rospy.ROSInterruptException: pass |
| --- |

**Discussion:**

In this Lab, we have acquired a fundamental understanding of controlling the movement of a turtlebot to generate a rectangle and spiral using ROS and Python. The lab provided us with basic knowledge of ROS packages and nodes by controlling the turtle bot using Python scripting.

**Question Answer:**

**1) How does the communication between the controller and the turtle bot happen?**

* The Python script is the controller and acts as the sender of messages to the *turtlesim\_node* which is the receiver of commands, through specific communication channels (ROS topics). The *turtlesim\_node* interprets the commands and allows the turtle to move accordingly. This forms a communication loop and allows the turtle to move in its simulated environment.

**2) What are the challenges faced in this lab?**

* However, no challenges were faced during rectangle generation using Python scripting and ROS. However, difficulties such as determining the correct radius of rotation and distance moved by the turtle during the rotation were faced while generating the spiral. To determine the suitable radius and distance moved, a try-and-error approach was taken in Python coding, and hard code was generated, which was extremely time-consuming.

**Video Link:**

[Rectangle and Spiral Path Simulation Video](https://youtu.be/vZL0h56GWIs).