

Section 1 Individual section

Personal Assignment 1

Question 1:

1 Could you use those (some of those) commands, to write and execute a 'yourfirstname_lastname.py' file, to calculate the sum of 0 to 99?
2



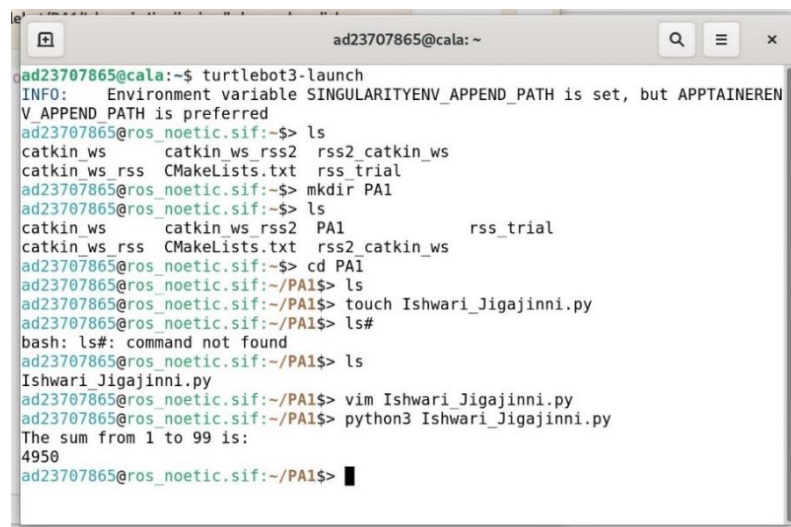
```
ad23707865@cala: ~
sm= sum(range(0,100))
print("The sum from 1 to 99 is:")
print(sm)
```

Answer:

The above python with the name 'Ishwari_Jigajinni.py' will calculate the sum of 0 to 99 and print it to the terminal. 'Python3 Ishwari_Jigajinni.py' is used to run the program. The result 4950 was printed in the terminal.

Question 2.

3 Detail what commands you used and for what purpose?



```
ad23707865@cala: ~
ad23707865@cala:~$ turtlebot3-launch
INFO: Environment variable SINGULARITYENV_APPEND_PATH is set, but APPTAINEREN
V_APPEND_PATH is preferred
ad23707865@ros_noetic.sif:~$ ls
catkin_ws      catkin_ws_rss2  rss2_catkin_ws
catkin_ws_rss  CMakeLists.txt  rss_trial
ad23707865@ros_noetic.sif:~$ mkdir PA1
ad23707865@ros_noetic.sif:~$ ls
catkin_ws      catkin_ws_rss2  PA1          rss_trial
catkin_ws_rss  CMakeLists.txt  rss2_catkin_ws
ad23707865@ros_noetic.sif:~$ cd PA1
ad23707865@ros_noetic.sif:~/PA1$ ls
ad23707865@ros_noetic.sif:~/PA1$ touch Ishwari_Jigajinni.py
ad23707865@ros_noetic.sif:~/PA1$ ls#
bash: ls#: command not found
ad23707865@ros_noetic.sif:~/PA1$ ls
Ishwari_Jigajinni.py
ad23707865@ros_noetic.sif:~/PA1$ vim Ishwari_Jigajinni.py
ad23707865@ros_noetic.sif:~/PA1$ python3 Ishwari_Jigajinni.py
The sum from 1 to 99 is:
4950
ad23707865@ros_noetic.sif:~/PA1$
```

Answer:

The commands used were.

1. 'turtlebot3-launch': is used to enter the ROS.
2. 'ls': is used to list the number of files in the current directory.
3. 'mkdir PA1': is used to create a directory (PA1).
4. 'cd PA1': is used to enter the directory PA1.
5. 'cd ..': is used to go back.
6. 'touch Ishwari_Jigajinni': is used to create the file.
7. 'vim Ishwari_Jigajinni': is used to edit the file.

8. 'chmod +x Ishwari_Jigajinni': is used to assign the permissions.
9. 'python3 Ishwari_Jigajinni.py': is used to run the python file (Ishwari_Jigajinni).

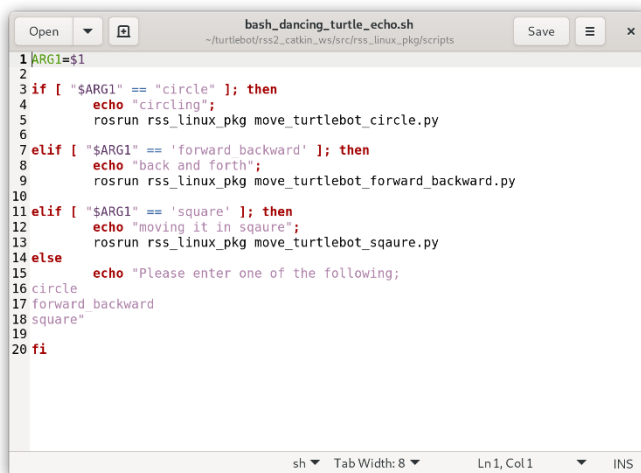
Personal Assignment 2

```

1 After fully understanding files in
2 1. '~/catkin_ws_rss/src/rss_linux_pkg/scripts'
3 2. '~/catkin_ws_rss/src/rss_linux_pkg/src'
4
5 Could you please write a '.py' file and then modify
6
7 3. bash bash_dancing_turtle_echo.sh circle or forward_backward
8
9 to make the robot move in a square?

```

Answer:



```

bash_dancing_turtle_echo.sh
~/turtlebot/rss2_catkin_ws/src/rss_linux_pkg/scripts
1 ARG1=$1
2
3 if [ "$ARG1" == "circle" ]; then
4     echo "circling";
5     rosrn rss_linux_pkg move_turtlebot_circle.py
6
7 elif [ "$ARG1" == 'forward backward' ]; then
8     echo "back and forth";
9     rosrn rss_linux_pkg move_turtlebot_forward_backward.py
10
11 elif [ "$ARG1" == 'square' ]; then
12     echo "moving it in square";
13     rosrn rss_linux_pkg move_turtlebot_sqaure.py
14 else
15     echo "Please enter one of the following;
16 circle
17 forward backward
18 square"
19
20 fi

```

The script 'bash_dancing_turtle_echo.sh' provided in week 2 is edited as shown on the left. If the input given is square it calls 'move_turtlebot_square.py' from rss_linux_pkg.

The small snippet of the 'move_turtlebot_square.py'

The robot is designed to travel in a square with 0.5 units for each side. It moves forward at a speed of 0.2 units per second and turns a

The workflow is:

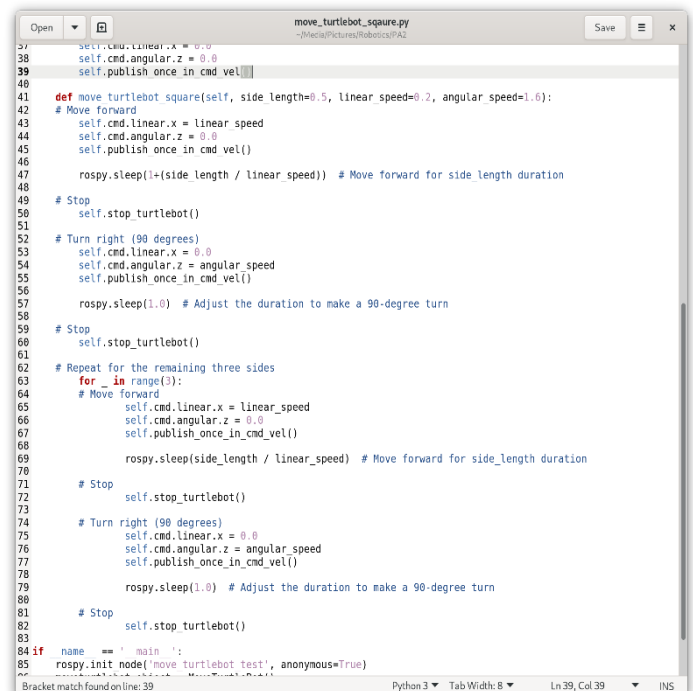
1. The robot begins by traveling 0.2 units per second forward in the x direction.

Using the formula,
 $\text{time} = \text{side_length} / \text{linear_velocity}$.

The robot moves in linear direction for 2.5 seconds

2. After moving forward, the robot stops and then rotates 90 degrees. Since the angular velocity is 1.6 radians per second, it takes 1 second to make the turn.

3. The robot repeats this process for the remaining three sides of the square.



```

move_turtlebot_square.py
~/Media/Pictures/Robotics/PA2
37 self.cmd.linear.x = 0.0
38 self.cmd.angular.z = 0.0
39 self.publish_once_in_cmd_vel()
40
41 def move_turtlebot_square(self, side_length=0.5, linear_speed=0.2, angular_speed=1.6):
42     # Move forward
43     self.cmd.linear.x = linear_speed
44     self.cmd.angular.z = 0.0
45     self.publish_once_in_cmd_vel()
46     rospy.sleep((side_length / linear_speed)) # Move forward for side length duration
47
48     # Stop
49     self.stop_turtlebot()
50
51     # Turn right (90 degrees)
52     self.cmd.linear.x = 0.0
53     self.cmd.angular.z = angular_speed
54     self.publish_once_in_cmd_vel()
55     rospy.sleep(1.0) # Adjust the duration to make a 90-degree turn
56
57     # Stop
58     self.stop_turtlebot()
59
60     # Repeat for the remaining three sides
61     for _ in range(3):
62         # Move forward
63         self.cmd.linear.x = linear_speed
64         self.cmd.angular.z = 0.0
65         self.publish_once_in_cmd_vel()
66         rospy.sleep(side_length / linear_speed) # Move forward for side_length duration
67
68         # Stop
69         self.stop_turtlebot()
70
71         # Turn right (90 degrees)
72         self.cmd.linear.x = 0.0
73         self.cmd.angular.z = angular_speed
74         self.publish_once_in_cmd_vel()
75         rospy.sleep(1.0) # Adjust the duration to make a 90-degree turn
76
77         # Stop
78         self.stop_turtlebot()
79
80     if __name__ == '__main__':
81         rospy.init_node('move_turtlebot_test', anonymous=True)
82
83
84
85

```

When observed in the Gazebo simulation environment, the robot moves in a square path. However, the shape isn't a perfect square due to inertia or environmental factors.

Personal Assignment 3

Question 1.

1. Rewrite the 'pengwang_publisher.py' such that the robot goes along a straight line (e.g. x direction or y direction, etc.), and name it as 'pengwang_publisher_line.py'.

Answer:

Publisher: A publisher in ROS is a node that sends out data or messages to a specific channel (topic). It shares information like sensor readings or commands with other nodes.

Subscriber: A subscriber in ROS is a node that receives data or messages from a specific channel (topic). It listens for updates and processes the received information.

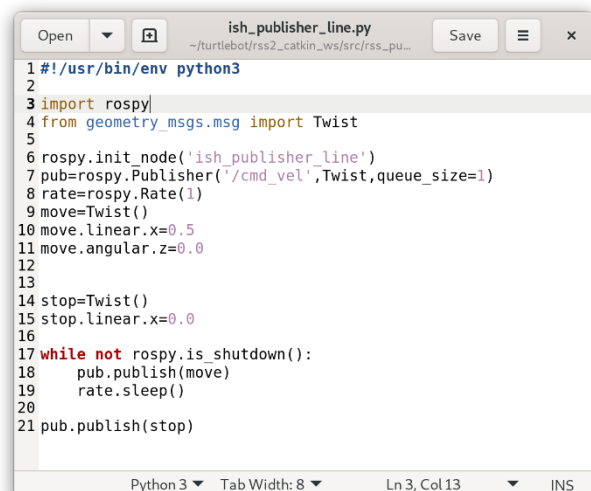
Topic: A topic in ROS is a named bus over which nodes exchange messages. It acts as a communication channel connecting publishers and subscribers.

The ish_publisher.py is edited to ish_publisher_line.py, so that it goes along a straight line. The program above creates a publisher that publishes messages of type 'Twist' to cmd_vel topic.

The Ish_publisher_line sets the linear velocity to 0.5 and angular velocity to 0.

So, the robot only has linear motion and no angular motion.

This node continuously publishes the velocity commands indicating robot is moving in a straight line.

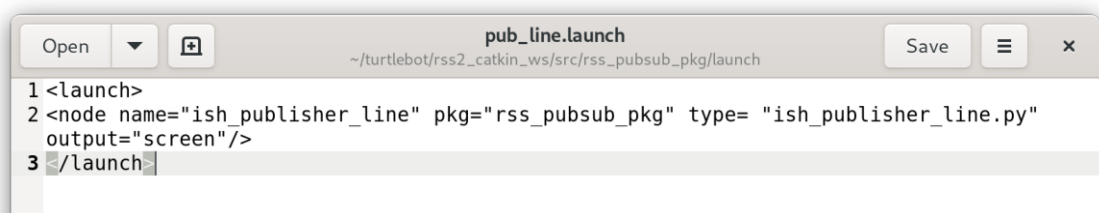


```
1#!/usr/bin/env python3
2
3import rospy
4from geometry_msgs.msg import Twist
5
6rospy.init_node('ish_publisher_line')
7pub=rospy.Publisher('/cmd_vel',Twist,queue_size=1)
8rate=rospy.Rate(1)
9move=Twist()
10move.linear.x=0.5
11move.angular.z=0.0
12
13
14stop=Twist()
15stop.linear.x=0.0
16
17while not rospy.is_shutdown():
18    pub.publish(move)
19    rate.sleep()
20
21pub.publish(stop)
```

Question 2

2. Write a launch file for 'pengwang_publisher_line.py'.

Answer: Launch files in ROS are used to start multiple nodes and set parameters efficiently, simplifying the initialization of complex systems. Here only one node "ish_publisher_line" is initialised from rss_pubsub_pkg.



```
1<launch>
2<node name="ish_publisher_line" pkg="rss_pubsub_pkg" type="ish_publisher_line.py"
  output="screen"/>
3</launch>
```

23707865

The launch file for `Ish_publisher_line` is written as above. We can launch it using the command
'`roslaunch rss_pubsub_pkg Ish_publisher_line.launch`'

Question 3

3. Use the '`pengwang_subscriber.py`' to listen to the topic published by '`pengwang_publisher_line.py`', to show the robot is indeed going along a straight line.

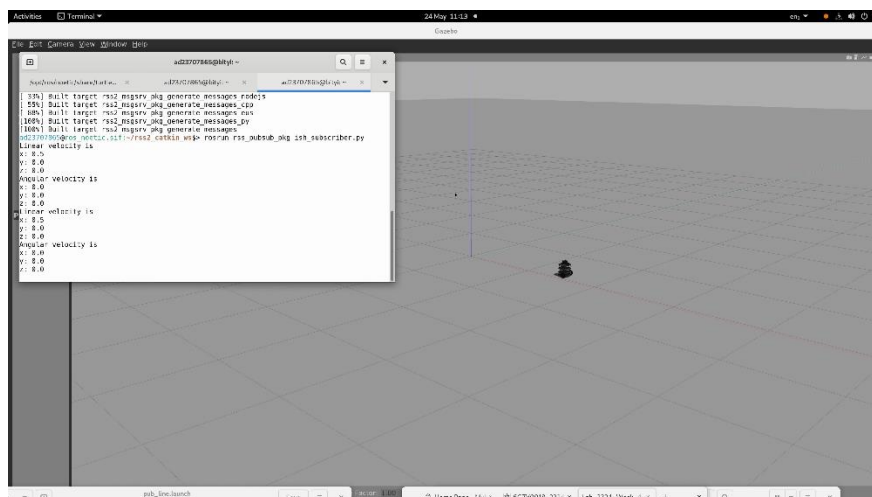
Answer:

The same subscriber can be used for both `Ish_publisher.py` and `Ish_publisher_line.py`. The subscriber script `ish_subscriber.py` listens to the `/cmd_vel` topic and prints the received commands as seen below. The linear velocity is consistently 0.5 and angular velocity is 0.0. This indicates that there is a straight-line motion.

```
[100%] Built target rss2_msgsrv_pkg_generate_messages_py
[100%] Built target rss2_msgsrv_pkg_generate_messages
ad23707865@ros_noetic.sif:~/rss2_catkin_ws$ rosrn rss_pubsub_pkg ish_subscriber.py
Linear velocity is
x: 0.5
y: 0.0
z: 0.0
Angular velocity is
x: 0.0
y: 0.0
z: 0.0
Linear velocity is
x: 0.5
y: 0.0
z: 0.0
Angular velocity is
x: 0.0
y: 0.0
z: 0.0
Linear velocity is
x: 0.5
y: 0.0
z: 0.0
Angular velocity is
x: 0.0
y: 0.0
z: 0.0
Linear velocity is
x: 0.5
y: 0.0
z: 0.0
Angular velocity is
x: 0.0
y: 0.0
z: 0.0
```

Question 4

4. Launch the `turtlebot3` empty environment.



After launching `Ish_publisher_line.py`, `Ish_subscriber.py`, and `turtlebot3_empty_world` is three separate terminals. We can see that the robot correctly moves in a straight line. In the subscriber terminal, we see the messages indicating linear velocity of 0.5 and angular velocity 0.0 confirming straight line indication.

In the Gazebo simulation we can visually confirm that it indeed goes in a straight line

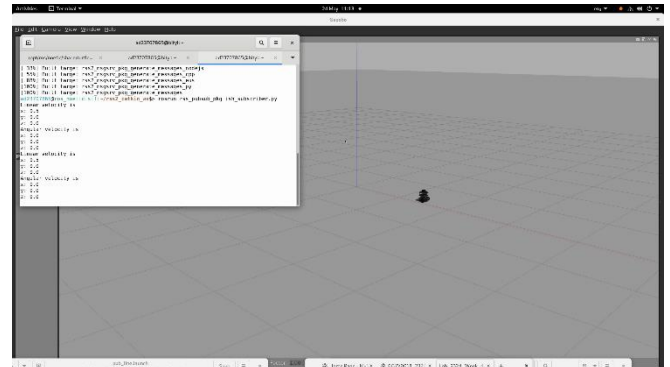
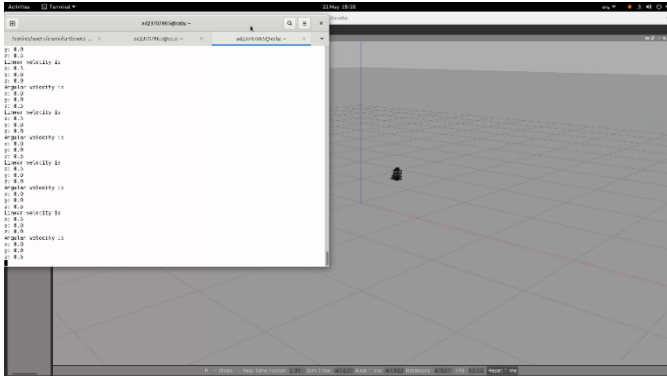
23707865

The turtlebot keeps moving in a straight line for infinite duration.

Question 5:

5. Observe and describe what happens when you launch 'pengwang_publisher.py' and 'pengwang_publisher_line.py', respectively.

Answer:



In the Ish_publisher.py, the turtlebot3_model in the gazebo moves in a circular motion, because it has both angular velocity and linear velocity of 0.5.

Whereas in Ish_publisher_line.py, the turtlebot3_model moves in a straight line for infinite time duration.

Question 6

6. Can you stop the robot from running? (Hint: There are two ways!)

Answer:

The robot runs for an unlimited time, this is because we don't call the stop function till the rospy is shutdown. So to stop the robot from running we have to shut down publisher_line file by pressing Ctrl+C or

Personal Assignment 4

Questions 1 and 2

1. Write a 'pengwang_pubsub.py' file, where you subscribe to '/odom' and publish to '/cmd_vel'.
2. In the file, define a distance the robot needs to run. After the robot reach the distance, stop.

Answer

This Python script employs ROS (Robot Operating System) for robot motion control. It subscribes to odometry data to monitor distance traveled, issues velocity commands to '/cmd_vel', and halts the robot once it travels a designated distance. The linear velocity is set to 0.5 in the x-direction, while velocities in other directions remain zero. The robot continues moving until it covers 5 units.

```

Open  [icon] ish_pubsub.py ~turtlebot/rss2_catkin_ws/src/rss_pubsub_... Save [icon] x
1 #!/usr/bin/env python3
2
3 import rospy
4 from geometry_msgs.msg import Twist
5 from nav_msgs.msg import Odometry
6 import math
7
8 class PengwangPubSub:
9     def __init__(self):
10         rospy.init_node('ish_pubsub')
11
12         # Publisher for the /cmd_vel topic
13         self.pub = rospy.Publisher('/cmd_vel', Twist,
14 queue_size=1)
15
16         # Subscriber for the /odom topic
17         self.sub = rospy.Subscriber('/odom', Odometry,
18 self.odom_callback)
19
20         # Define the distance the robot needs to run
21         self.distance_to_run = 5.0 # meters
22         self.current_distance = 0.0
23
24         # Variables to store previous position
25         self.prev_x = None
26         self.prev_y = None
27
28     def odom_callback(self, msg):
29         # Get the current position from the odometry message
30         x = msg.pose.pose.position.x
31         y = msg.pose.pose.position.y
32
33         # Calculate the distance traveled from the previous
34         position

```

Python 3 Tab Width: 8 Ln 59, Col 61 INS

```

Open  [icon] ish_pubsub.py ~turtlebot/rss2_catkin_ws/src/rss_pubsub_pkg/sc... Save [icon] x
31 # Calculate the distance traveled from the previous
32 position
33 if self.prev_x is not None and self.prev_y is not None:
34     distance_increment = math.sqrt((x - self.prev_x)**2 +
35 (y - self.prev_y)**2)
36     self.current_distance += distance_increment
37
38     # Update previous position
39     self.prev_x = x
40     self.prev_y = y
41
42     # Display the current distance traveled
43     rospy.loginfo(f"Current Distance: {self.current_distance:.
44 2f} meters")
45
46     def run(self):
47         rate = rospy.Rate(10) # Set the rate to 10 Hz
48         move = Twist()
49         move.linear.x = 0.5
50         move.linear.y = 0.0
51         move.linear.z = 0.0
52         # Set a constant linear velocity in x direction
53
54         while not rospy.is_shutdown() and self.current_distance <
55 self.distance_to_run:
56             self.pub.publish(move)
57             rate.sleep()
58
59             # Stop the robot
60             move.linear.x = 0.0
61             self.pub.publish(move)
62
63             # Display a message indicating the robot has stopped
64             rospy.loginfo("Target distance reached. Stopping the
65 robot.")
66
67 if __name__ == '__main__':
68     try:
69         pubsub = PengwangPubSub()
70         pubsub.run()
71     except rospy.ROSInterruptException:
72         pass

```

Python 3 Tab Width: 8 Ln 59, Col 61 INS

23707865

Question 3

3. Write a launch file to start it.

Answer

```
ish_pubsub.launch
~/turtlebot/rss2_catkin_ws/src/rss_pubsub_pkg/launch

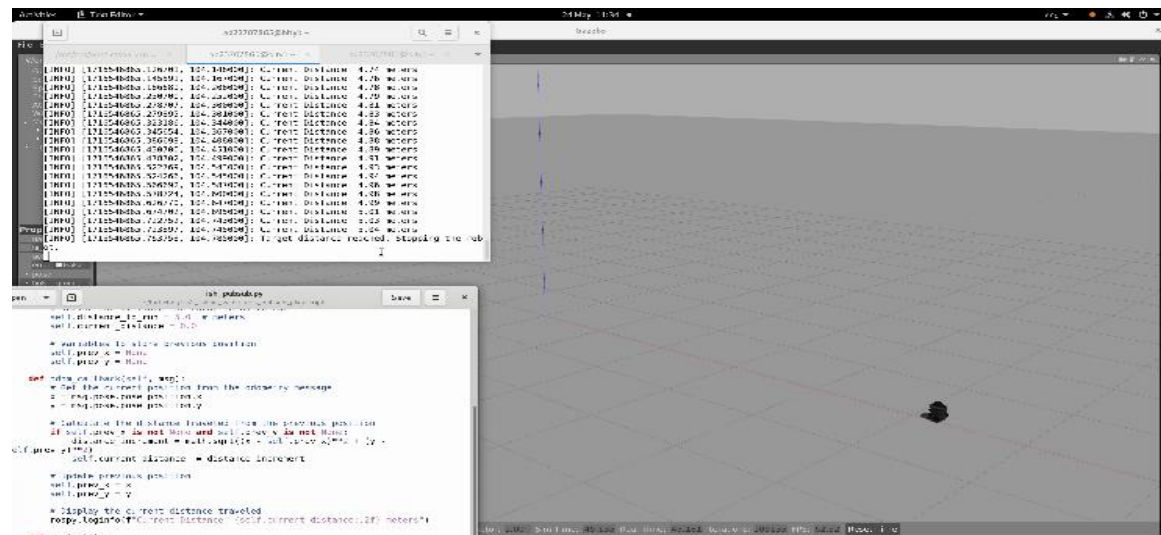
1 <launch>
2 <include file="$(find turtlebot3_gazebo)/launch/turtlebot3_empty_world.launch"/>
3 <node name="ish_pub" pkg="rss_pubsub_pkg" type="ish_pubsub.py" output="screen"/>
4 </launch>
```

The launch file is written as above. When you run this the launch file launches gazebo and then creates a node called ish_pub which runs the ish_pubsub.py python script.

Question 4

4. Launch the turtlebot3 empty environment to observe if it works.

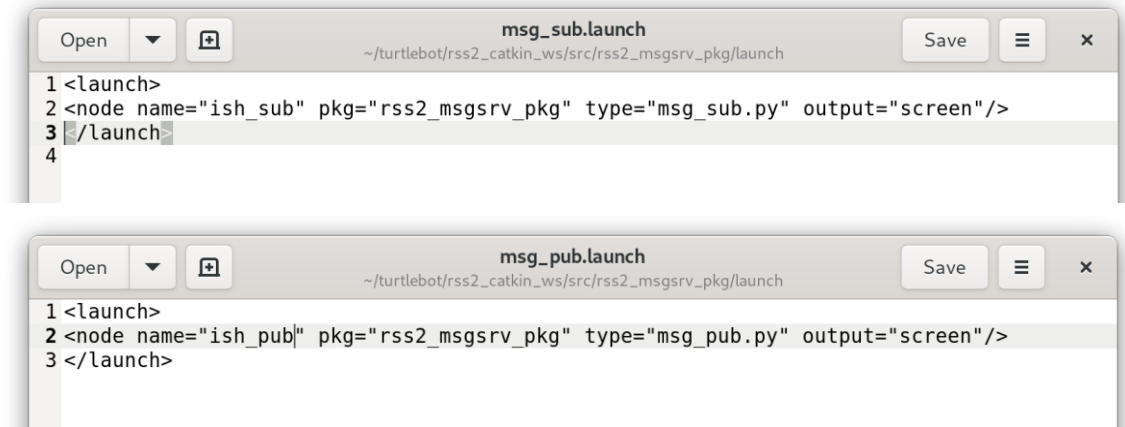
Answer



When turtlebot3_empty_world is launched in Gazebo, the robot is supposed to move in a straight line for 5 units with a linear velocity of 0.5. then it stops. But in the simulation, we see that the robot moves more than 5 units and then stops. This is due to a robot scale error.

PERSONAL ASSIGNMENT 5**Question 1**

- Create launch files to start the publisher and subscriber in section 5.2 in Part I.

Answer

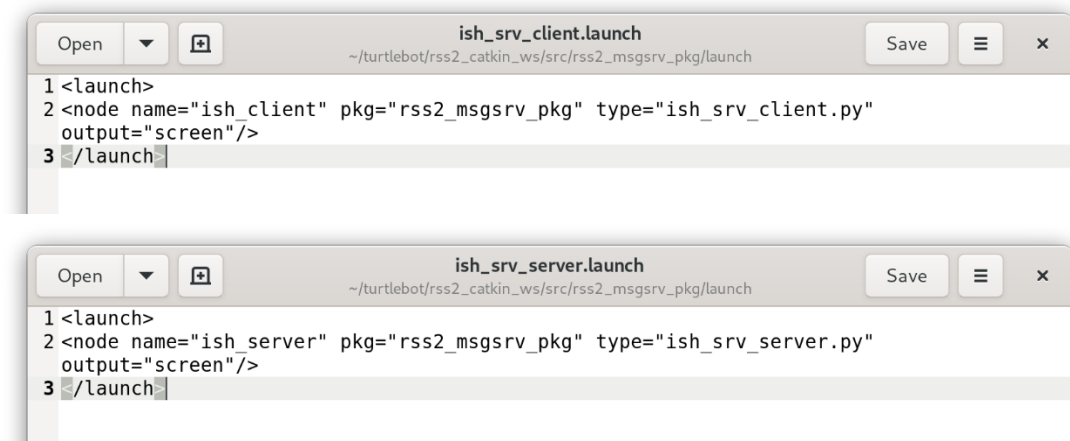
```
msg_sub.launch
~/turtlebot/rss2_catkin_ws/src/rss2_msgsrv_pkg/launch
1 <launch>
2 <node name="ish_sub" pkg="rss2_msgsrv_pkg" type="msg_sub.py" output="screen"/>
3 </launch>
4

msg_pub.launch
~/turtlebot/rss2_catkin_ws/src/rss2_msgsrv_pkg/launch
1 <launch>
2 <node name="ish_pub" pkg="rss2_msgsrv_pkg" type="msg_pub.py" output="screen"/>
3 </launch>
```

The launch files for msg_pub.py and msg_sub.py are created as shown above.

Question 2

- Create launch files to start the server and client in sections 4.7 and 4.8 in Part II.

Answer

```
ish_srv_client.launch
~/turtlebot/rss2_catkin_ws/src/rss2_msgsrv_pkg/launch
1 <launch>
2 <node name="ish_client" pkg="rss2_msgsrv_pkg" type="ish_srv_client.py"
  output="screen"/>
3 </launch>

ish_srv_server.launch
~/turtlebot/rss2_catkin_ws/src/rss2_msgsrv_pkg/launch
1 <launch>
2 <node name="ish_server" pkg="rss2_msgsrv_pkg" type="ish_srv_server.py"
  output="screen"/>
3 </launch>
```

The launch files for server and client are created as above.

Question 3, 4, 5, 6

- Modify (or you can do it from scratch) the codes in sections 4.7 and 4.8 in Part II, such that the robot moves for 30 seconds.

1. The first 20 seconds the robot moves in a circle
2. Then stops for 5 seconds.
3. Then moves along x-axis for 5 seconds
4. Stop

Answer

The server file in 4.7 is changed as below.

```

1 #!/usr/bin/env python3
2
3 import rospy
4 from rsl_msgs.srv import srv_turtlebot_move, srv_turtlebot_moveResponse
5 from geometry_msgs.msg import Twist
6
7 def my_callback(request):
8     rospy.loginfo('Turtlebot move service has been called')
9
10    vel.linear.x = 0.4
11    vel.angular.z = 0.8
12    total_time = 0
13    while total_time <= 20:
14        ish_pub.publish(vel)
15        rospy.loginfo('Moving in a circle. Time = %d seconds', total_time)
16        rate.sleep() # Sleep for 1 second
17        total_time += 1
18
19    # Stop for 5 seconds
20    vel.linear.x = 0.0
21    vel.angular.z = 0.0
22    ish_pub.publish(vel)
23    stop_time = 0
24    while stop_time < 5:
25        rospy.loginfo('Stopping. Time = %d seconds', stop_time)
26        rate.sleep() # Sleep for 1 second
27        stop_time += 1
28
29    # Move along the x-axis for 5 seconds
30    vel.linear.x = 0.2 # Forward linear velocity
31    vel.angular.z = 0.0 # No angular velocity
32    total_time = 0
33    while total_time <= 5:
34        ish_pub.publish(vel)
35        rospy.loginfo('Moving along x-axis. Time = %d seconds', total_time)
36        rate.sleep() # Sleep for 1 second
37        total_time += 1
38
39    # Stop
40    vel.linear.x = 0.0
41    vel.angular.z = 0.0
42    ish_pub.publish(vel)
43

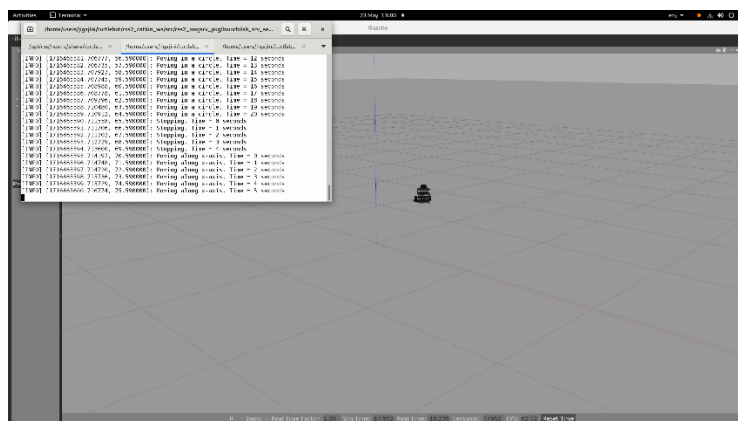
```

```

39 # Stop
40 vel.linear.x = 0.0
41 vel.angular.z = 0.0
42 ish_pub.publish(vel)
43
44 return srv_turtlebot_moveResponse(True)
45
46 rospy.init_node('turtlebot_move_server')
47
48 # This is the service called '/turtlebot_move_service'
49 ish_service = rospy.Service('/turtlebot_move_service', srv_turtlebot_move,
50 my_callback)
51
52 ish_pub = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
53 vel = Twist()
54
55 # Make sure counting second by second
56 rate = rospy.Rate(1) # 1 Hz rate for 1 second intervals
57
58 rospy.loginfo('Service /turtlebot_move_service is ready!')
59
60 # Maintain the service open
61 rospy.spin()
62

```

To manage service requests, the script establishes a callback function called "my_callback" and initialises a ROS service called "turtlebot_move_service". It adjusts the TurtleBot's velocity to move in a circle for 20 seconds after receiving a request. The robot is then stopped for five seconds and then moves for five seconds in the x-axis direction. Using a 'Twist' message, velocity commands are published to '/cmd_vel'. A rate of one hertz (Hz) maintains time synchronization, so that operations take place at one-second intervals.



When the gazebo is launched we see the robot circles for time 20 seconds and then it stops for 5 seconds and then it starts moving in straight line for 5 seconds then stops completely.

Even if the time synchronisation is set at 1 Hz, the actual time, and the time according to the robots does not tally with each other. For robots, the time moves little faster.

Personal Assignment 6

Question 1

Given the following service message type

```
1 float64 sideLength
2 int32 repetitions
3 ---
4 bool success
```

- Create a 'turtlebot_move_square.srv' message, and put it in the right place.

Answer

```
1 float64 sideLength
2 int32 repetitions
3 ---
4 bool success
```

The service file turtlebot_move_square.srv is created as above and put in the srv folder of rrs2_msgsrv_pkg folder.

Question 2

- Modify 'CMakeLists.txt' and 'package.xml'.

Answer

```
50
51 ## Generate messages in the 'msg' folder
52 add_message_files(
53   FILES
54   date_cmd_vel.msg
55   Message2.msg
56 )
57
58 ## Generate services in the 'srv' folder
59 add_service_files(
60   FILES
61   srv_turtlebot_move.srv
62   turtlebot_move_square.srv
63   Service2.srv
64 )
65
```

The CMakeLists.txt is modified as shown to the left. The service file turtlebot_move_square.srv is added to it.

```

50 <!-- <doc_depend>doxygen</doc_depend> -->
51 <buildtool_depend>catkin</buildtool_depend>
52 <build_depend>geometry_msgs</build_depend>
53 <build_depend>nav_msgs</build_depend>
54 <build_depend>rospy</build_depend>
55 <build_depend>std_msgs</build_depend>
56 <build_depend>message_generation</build_depend>
57
58 <build_export_depend>geometry_msgs</build_export_depend>
59 <build_export_depend>nav_msgs</build_export_depend>
60 <build_export_depend>rospy</build_export_depend>
61 <build_export_depend>std_msgs</build_export_depend>
62 <build_export_depend>message_runtime</build_export_depend>
63
64 <exec_depend>geometry_msgs</exec_depend>
65 <exec_depend>nav_msgs</exec_depend>
66 <exec_depend>rospy</exec_depend>
67 <exec_depend>std_msgs</exec_depend>
68 <exec_depend>message_runtime</exec_depend>
69

```

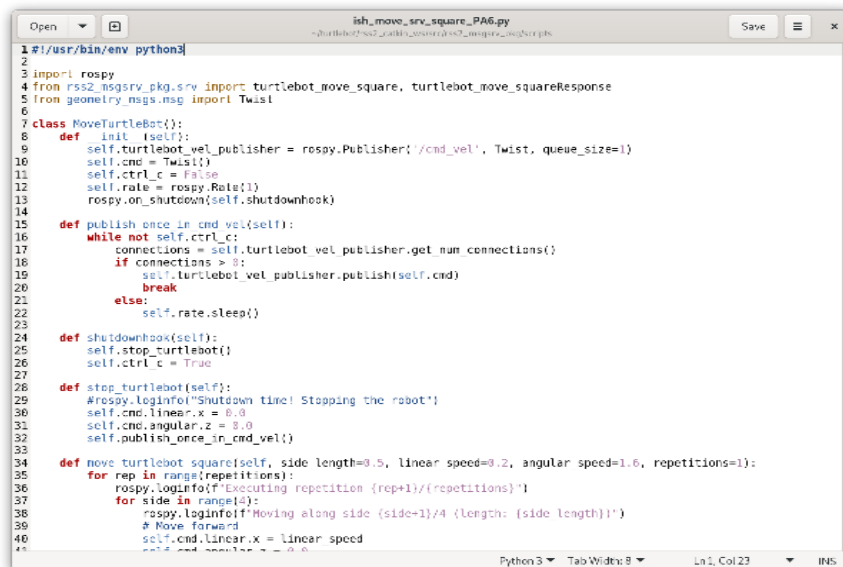
The Package.xml file is edited as shown on the right.

Questions 3,4, and 5

- Create a client and a server that use the 'turtlebot_move_square.srv' message to do
 1. When the server being called, it should move along a square with side length defined by 'sideLength'.
 2. The robot must repeat moving along the square defined by 'repetitions', e.g., if 'repetitions=4', then the robot must move along the square 4 times.
 3. When the robot finishes the movement, it should return 'True'. Otherwise, 'False'.

Answer:

SERVER :



```

1 #!/usr/bin/env python3
2
3 import rospy
4 from ross2_msgs.srv import turtlebot_move_square, turtlebot_move_squareResponse
5 from geometry_msgs.msg import Twist
6
7 class MoveTurtleBot():
8     def __init__(self):
9         self.turtlebot_vel_publisher = rospy.Publisher('/cmd_vel', Twist, queue_size=1)
10        self.cmd = Twist()
11        self.ctrl_c = False
12        self.rate = rospy.Rate(1)
13        rospy.on_shutdown(self.shutdownhook)
14
15        def publish_once_in_cmd_vel(self):
16            while not self.ctrl_c:
17                connections = self.turtlebot_vel_publisher.get_num_connections()
18                if connections > 0:
19                    self.turtlebot_vel_publisher.publish(self.cmd)
20                    break
21            else:
22                self.rate.sleep()
23
24        def shutdownhook(self):
25            self.stop_turtlebot()
26            self.ctrl_c = True
27
28        def stop_turtlebot(self):
29            #rospy.loginfo("Shutdown time! Stopping the robot")
30            self.cmd.linear.x = 0.0
31            self.cmd.angular.z = 0.0
32            self.publish_once_in_cmd_vel()
33
34        def move_turtlebot_square(self, side_length=0.5, linear_speed=0.2, angular_speed=1.6, repetitions=1):
35            for rep in range(repetitions):
36                rospy.loginfo("Executing repetition {rep+1}/{repetitions}")
37                for side in range(4):
38                    rospy.loginfo("Moving along side {side+1}/4 (length: {side_length})")
39                    # Move forward
40                    self.cmd.linear.x = linear_speed
41                    self.cmd.angular.z = 0.0

```

23707865

```

1  ihs_move_srv_srvc_PAA6.py
2  ~turtlebot2_ws2_catkin_ws/src/move2_move2/scripts
3  Open ▾
4  Save
5  [Menu Icon]
6
7  37 rospy.loginfo(f"Executing repetition {req.repetitions} /
8  38 for side in range(4):
9  39     rospy.loginfo(f"Moving along side {side+1}/4 (length: {side_length})")
10  40     # Move forward
11  41     self.cmd.linear.x = linear_speed
12  42     self.cmd.angular.z = 0.0
13  43     self.publish_once_in_cmd_vel()
14  44     rospy.sleep(1 + (side_length / linear_speed)) # Move forward for side_length duration
15  45
16  46     # Stop
17  47     self.stop_turtlebot()
18  48
19  49     # Turn right (90 degrees)
20  50     self.cmd.linear.x = 0.0
21  51     self.cmd.angular.z = angular_speed
22  52     self.publish_once_in_cmd_vel()
23  53     rospy.sleep(1.0) # Adjust the duration to make a 90-degree turn
24  54
25  55     # Stop
26  56     self.stop_turtlebot()
27  57
28  58 def handle_move_square(req):
29  59     rospy.loginfo(f"Moving in a square with side length {req.sideLength} and {req.repetitions} repetitions")
30  60     mover = MoveTurtleBot()
31  61     try:
32  62         mover.move_turtlebot_square(side_length=req.sideLength, repetitions=req.repetitions)
33  63         return turtlebot_move_squareResponse(True)
34  64     except Exception as e:
35  65         rospy.logerr(f"Error while moving: {e}")
36  66         return turtlebot_move_squareResponse(False)
37  67
38  68 def move_square_server():
39  69     rospy.init_node('move_square_server')
40  70     s = rospy.Service('turtlebot_move_square', turtlebot_move_square, handle_move_square)
41  71     rospy.loginfo("Ready to move in a square.")
42  72     rospy.spin()
43  73
44  74 if __name__ == '__main__':
45  75     move_square_server()
46  76

```

This script moves turtleBot in a square shape. It uses ROS, a system for controlling robots. The TurtleBot moves forward for a certain distance, then turns right, and repeats this to form a square.

This Python script defines a ROS service `turtlebot_move_square` to make a TurtleBot move in a square pattern. It initializes a class `MoveTurtleBot` to control the robot's movement, with methods for publishing velocity commands, stopping the robot, and moving it in a square. The `move_square_server` function sets up the sROS node and service.

When the service is called, it moves the TurtleBot in a square path with configurable side length, linear and angular speeds, and repetitions. Time intervals ensure precise movement. Any errors during execution are logged.

CLIENT

```
Open ▾ ▶ ◀ ◂ ◃ ◅ ▹ ▸ ◂ ◃ ◅ ▹ ▸
```

```
ish_move_client_squarePA6.py
~\turtlebotreal_cake_wiki\client_rosery_pkg\scripts
```

```
Save ◻ ◼ ◽ ◾ ◿ ◰ ◱ ◲ ◳ ◴ ◵ ◶ ◷ ◸ ◹ ◺ ◻ ◼ ◽ ◾ ◿ ◰ ◱ ◲ ◳ ◴ ◵ ◶ ◷ ◸ ◹ ◺
```

```
ish_move_srv_square_PA6.py
x
```

```
ish_move_client_squarePA6.py
x
```

```
1 #!/usr/bin/env python3
2
3 import rospy
4 from rsl2_msgsrv.pkg.srv import turtlebot_move_square
5
6 def move_square_client(side_length, repetitions):
7     rospy.wait_for_service('turtlebot_move_square')
8     try:
9         move_square = rospy.ServiceProxy('turtlebot_move_square', turtlebot_move_square)
10        resp = move_square(side_length, repetitions)
11        return resp.success
12    except rospy.ServiceException as e:
13        rospy.logerr(f'Service call failed: {e}')
14        return False
15
16 if __name__ == '__main__':
17     rospy.init_node('move_square_client')
18     # Default values for side length and repetitions
19     side_length = rospy.get_param('~side_length', 1.0)
20     repetitions = rospy.get_param('~repetitions', 4)
21     success = move_square_client(side_length, repetitions)
22     if success:
23         rospy.loginfo("TurtleBot moved in square successfully")
24     else:
25         rospy.logerr("Failed to move TurtleBot in square")
26
```

```
Python 3 ▾ Tab Width: 8 ▾ Ln19, Col 52 ▾ IN
```

This Python script creates a client to call the `'turtlebot_move_square'` service which we created in question 1 of this assignment, allowing users to command a TurtleBot to move in a square. It initializes a ROS node and defines a function `'move_square_client'` to interact with the service. We

23707865

specify the side length as 1 and the number of repetitions as 4. The client logs success or failure messages based on the service response.

SERVER OUTPUT

The server file displays which side of the square robot is moving in, it also says which repetition is going on.

```
ad23707865@bitity: ~  
roscore | ssh jopt/ros/noetic/share... | ad23707865@bitity: ~ | ad23707865@bitity: ~  
[INFO] [1716552037.633789, 100, 277000]: Moving along side 2/4 (length: 1.0)  
[INFO] [1716552043.836288, 106, 277000]: Shutdown time! Stopping the robot  
[INFO] [1716552044.836706, 107, 277000]: Shutdown time! Stopping the robot  
[INFO] [1716552044.836706, 107, 277000]: Moving along side 3/4 (length: 1.0)  
[INFO] [1716552059.039986, 113, 278000]: Shutdown time! Stopping the robot  
[INFO] [1716552059.040687, 114, 278000]: Shutdown time! Stopping the robot  
[INFO] [1716552059.041560, 114, 278000]: Moving along side 4/4 (length: 1.0)  
[INFO] [1716552057.437175, 120, 279000]: Shutdown time! Stopping the robot  
[INFO] [1716552058.436922, 121, 279000]: Shutdown time! Stopping the robot  
^C[INFO] [1716552106.155718, 163, 374000]: Shutdown time! Stopping the robot  
^C[INFO] [1716552106.155718, 163, 374000]: Ready to move square pkg ish move srv square PA6.py.  
[INFO] [1716552103.868856, 0.000000]: Ready to move square pkg ish move srv square PA6.py.  
[INFO] [1716552108.500176, 171, 715000]: Moving in a square with side length 1.0 and 4 repetitions  
[INFO] [1716552108.501712, 171, 716000]: Executing repetition 1/4  
[INFO] [1716552108.582386, 171, 717000]: Moving along side 1/4 (length: 1.0)  
[INFO] [1716552116.503777, 179, 718000]: Moving along side 2/4 (length: 1.0)  
[INFO] [1716552122.506731, 186, 719000]: Moving along side 3/4 (length: 1.0)  
[INFO] [1716552130.508752, 193, 716000]: Moving along side 4/4 (length: 1.0)  
[INFO] [1716552137.511736, 200, 716000]: Executing repetition 2/4  
[INFO] [1716552137.512571, 209, 716000]: Moving along side 1/4 (length: 1.0)  
[INFO] [1716552144.515768, 207, 717000]: Moving along side 2/4 (length: 1.0)  
[INFO] [1716552151.518732, 214, 717000]: Moving along side 3/4 (length: 1.0)  
[INFO] [1716552158.520459, 221, 717000]: Moving along side 4/4 (length: 1.0)  
[INFO] [1716552165.526755, 228, 717000]: Executing repetition 3/4  
[INFO] [1716552165.525356, 238, 718000]: Moving along side 1/4 (length: 1.0)  
[INFO] [1716552172.527752, 235, 718000]: Moving along side 2/4 (length: 1.0)  
[INFO] [1716552178.530734, 242, 718000]: Moving along side 3/4 (length: 1.0)  
[INFO] [1716552186.532746, 249, 718000]: Moving along side 4/4 (length: 1.0)  
[INFO] [1716552193.535735, 256, 718000]: Executing repetition 4/4  
[INFO] [1716552193.536566, 256, 718000]: Moving along side 1/4 (length: 1.0)  
[INFO] [1716552208.539750, 263, 719000]: Moving along side 2/4 (length: 1.0)  
[INFO] [1716552207.541762, 270, 719000]: Moving along side 3/4 (length: 1.0)  
[INFO] [1716552214.544739, 277, 719000]: Moving along side 4/4 (length: 1.0)
```

CLIENT OUTPUT

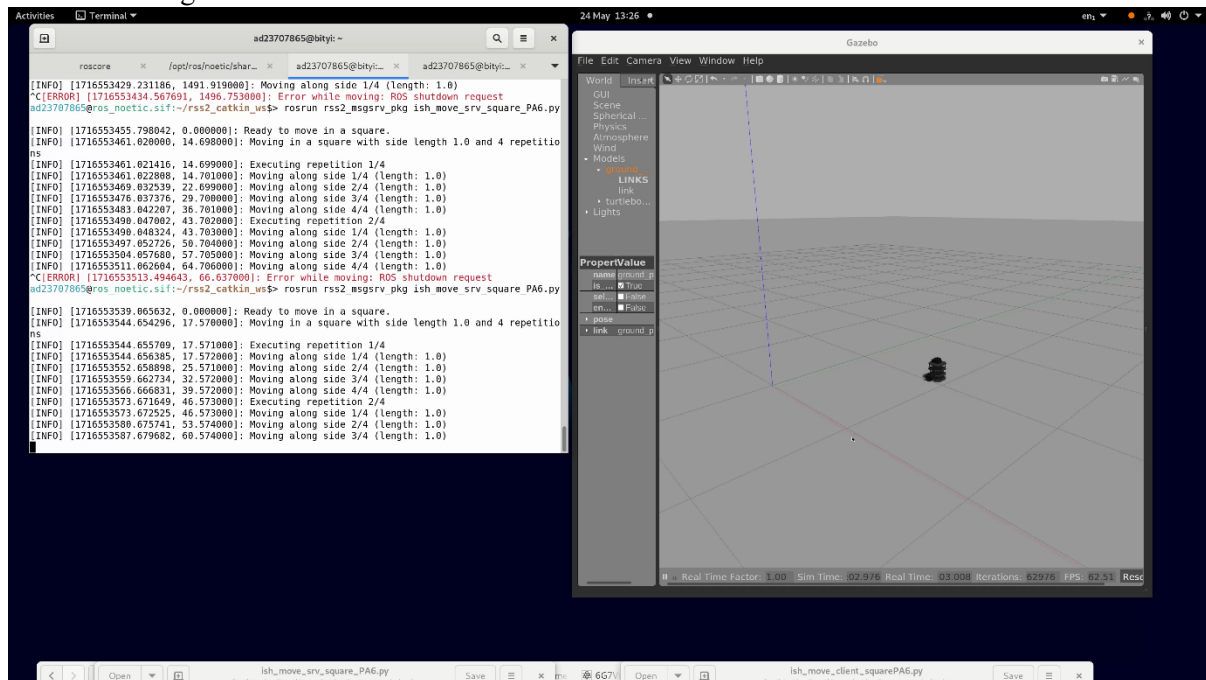
The client returns if the robot has successfully moved in the square or not. That will be printed on the terminal.

```
ad23707865@ros_noetic.sif:~/rss2_catkin_ws$> rosrn rss2_msgsrv_pkg ish_move_client_squarePA6.py
^C[ERROR] [1716551934.563981, 569.495000]: Service call failed: service [/turtlebot_move_square] r
eturned no response
[ERROR] [1716551934.564850, 569.495000]: Failed to move TurtleBot in square
ad23707865@ros_noetic.sif:~/rss2_catkin_ws$> rosrn rss2_msgsrv_pkg ish_move_client_squarePA6.py
[INFO] [1716552058.044669, 121.279000]: TurtleBot moved in square successfully
```

GAZEBO OUTPUT

23707865

In Gazebo, we can visualize that the robot is making 4 repetitions with side lengths a little higher than 1 unit. This might be due to some wheel calibration error or motor error.



Section 2. Group Assignment

Trello board

