



MANIPAL INSTITUTE OF TECHNOLOGY

MANIPAL

(A Constituent unit of MAHE, Manipal)

ASSIGNMENT 1

Submitted by

Adithya Golwalkar S R

220972012

2nd SEM - M.TECH

DEPARTMENT OF MECHATRONICS

DEPT. OF INDUSTRIAL AUTOMATION AND

ROBOTICS MANIPAL INSTITUTE OF

TECHNOLOGY, MANIPAL-576104

Exercise 1: Write a launch file `pub_sub.launch.py` to run the publisher and subscriber node.

CODE SNIPPET

Publisher

```
pub.py x
src > my_package > my_package > pub.py > ...
1  #!/usr/bin/
2  import rclpy
3  from rclpy.node import Node
4  from std_msgs.msg import String
5  class Publisher(Node):
6      def __init__(self):
7          super().__init__('Publisher')
8          self.get_logger().info("Publisher Node")
9          self.counter_=0
10         self.publisher_=self.create_publisher(String,"/message",10)
11         self.timer_=self.create_timer(1.0,self.publish_message)
12
13     def publish_message(self):
14         msg=String()
15         msg.data="Hello"+str(self.counter_)
16         self.publisher_.publish(msg)
17         self.counter_+=1
18
19
20 def main(args=None):
21     rclpy.init(args=args)
22     node=Publisher()
23     rclpy.spin(node)
24     rclpy.shutdown()
25
26 if __name__=="__main__":
27     main()
28
```

Subscriber

```
subscriber.py x
src > my_package > my_package > subscriber.py > ...
1  #!/usr/bin
2  import rclpy
3  from rclpy.node import Node
4  from std_msgs.msg import String
5  class subscriber (Node):
6      def __init__(self):
7          super().__init__("Subscriber")
8          self.get_logger().info("hello from subscr")
9          self.subscriber=self.create_subscription(String,"/message",self.msg_callback,10)
10
11      def msg_callback(self,msg):
12          self.get_logger().info(msg.data)
13
14
15  def main(args=None):
16      rclpy.init(args=args)
17      node=subscriber()
18      rclpy.spin(node)
19      rclpy.shutdown()
20
21
22  if __name__=="__main__":
23      main()
```

RESULT

The screenshot displays the ROS2 environment with a terminal window and the rqt_graph interface.

Terminal Output:

```
adithya@adithya-virtual-machine: ~/ros2_ws
adithya@adithya-virtual-machine:~/ros2_ws$ code
adithya@adithya-virtual-machine:~/ros2_ws$ colcon build --packages-select my_package
Starting >>> my_package
Finished <<< my_package [1.87s]

Summary: 1 package finished [2.11s]
adithya@adithya-virtual-machine:~/ros2_ws$ source install/setup.bash
adithya@adithya-virtual-machine:~/ros2_ws$ ros2 run pub
usage: ros2 run [-h] [--prefix PREFIX] package_name executable_name ...
ros2 run: error: the following arguments are required: executable_name, argv
adithya@adithya-virtual-machine:~/ros2_ws$ ros2 run my_package pub
[INFO] [1678381937.079271545] [Publisher]: Publisher Node
```

rqt_graph Output:

```
adithya@adithya-virtual-machine:~/ros2_ws$ ros2 run my_package sub
[INFO] [1678382420.000639565] [Subscriber]: hello from subscr
[INFO] [1678382421.107749847] [Subscriber]: Hello483
[INFO] [1678382422.107896769] [Subscriber]: Hello484
[INFO] [1678382423.110886263] [Subscriber]: Hello485
[INFO] [1678382424.109913282] [Subscriber]: Hello486
[INFO] [1678382425.106999212] [Subscriber]: Hello487
[INFO] [1678382426.108210022] [Subscriber]: Hello488
[INFO] [1678382427.109909712] [Subscriber]: Hello489
[INFO] [1678382428.107512704] [Subscriber]: Hello490
```

rqt_graph - rqt Node Graph:

The rqt_graph window shows a Node Graph with two nodes: /Publisher and /Subscriber. The /Subscriber node is connected to the /Publisher node via a message topic labeled /message.

```
graph LR
    Publisher([/Publisher]) -- /message --> Subscriber([/Subscriber])
```

Exercise 2: Create 2 nodes from scratch. First node has 1 publisher, the second has 1 publisher & 1 subscriber.

CODE SNIPPET

Publisher

```
1pubs.py x
src > my_package > my_package > 1pubs.py > ...
1  #!/usr/bin/env python3
2
3  import rclpy
4  from rclpy.node import Node
5  from std_msgs.msg import Int32
6
7  class publisher(Node):
8      def __init__(self):
9          super().__init__("Publsiher")
10         self.get_logger().info("Hello from ROS2, Node is Started")
11         # self.counter = 0
12         self.publisher_ = self.create_publisher(Int32, "/message", 10)
13         self.timer_ = self.create_timer(1.0, self.timer_callback)
14     def timer_callback(self):
15         msg = Int32()
16         msg.data = 8      #"Hello " + str(self.counter)
17         #self.get_logger().info(msg.data)
18         self.publisher_.publish(msg)
19         # self.counter+=1
20
21     def main(args=None):
22         rclpy.init(args=args)
23         node = publisher()
24         rclpy.spin(node)
25         rclpy.shutdown()
26
27     if __name__ == "__main__":
28         main()
29
```

Subscriber Publisher

```
2subspubs.py X
src > my_package > my_package > 2subspubs.py > ...
1  #!/usr/bin/env python3
2
3  import rclpy
4  from rclpy.node import Node
5  from std_msgs.msg import Int32
6  from std_msgs.msg import String
7
8  class subscriber(Node):
9      def __init__(self):
10         super().__init__("Subscriber")
11         self.get_logger().info("Hello from subscriber...please subscribe to my channel")
12         self.counter=0
13         self.publisher_ = self.create_publisher(Int32,"/msg_count",10)
14         self.subscribe_=self.create_subscription(Int32, "/message",self.subscriber_callback,10)
15
16         def subscriber_callback(self,msg):
17             msg_a = Int32()
18             msg_int = int(msg.data)
19             self.get_logger().info(str(msg.data))
20             #msg_a.data = str(msg)+"Hello_Sumukha " + str(self.counter)
21             msg_a.data = msg_int + self.counter
22             self.publisher_.publish(msg_a)
23             self.counter+=1
24
25     def main(args=None):
26         rclpy.init(args=args)
27         node = subscriber()
28         rclpy.spin(node)
29         rclpy.shutdown()
30
31
32 if __name__ == "__main__":
33     main()
34
```

Subscriber

```
3subs.py
src > my_package > my_package > 3subs.py > three_node > subscriber_callback
1  #!/usr/bin/env python3
2
3  import rclpy
4  from rclpy.node import Node
5  from std_msgs.msg import Int32
6  import math
7  import time
8
9
10 class three_node(Node):
11     def __init__(self):
12         super().__init__("Three_node")
13         self.get_logger().info("Node has been started...")
14         self.counter = 0
15         # self.timer_ = self.create_timer(1.0,self.timer_callback)
16         self.subscribe_=self.create_subscription(Int32, "/msg_count",self.subscriber_callback,10)
17         ## called /message topic
18
19     def subscriber_callback(self,msg_a):## Subscribed callback function
20         self.get_logger().info(str(msg_a.data))
21
22
23     def main(args=None):
24         rclpy.init(args=args)
25         node = three_node()
26         rclpy.spin(node)
27         rclpy.shutdown()
28
29
30 if __name__ == "__main__":
31     main()
32
```

RESULT

The image displays a ROS2 environment with three terminal windows and a Node Graph visualization.

Terminal 1 (Top Left): Shows the output of `ros2 run my_package pubs`. It prints: `[INFO] [1678384243.807321372] [Publisher]: Hello from ROS2, Node is Started`.

Terminal 2 (Top Right): Shows the output of `ros2 run my_package subs`. It prints: `[INFO] [1678384246.545388357] [Three_node]: Node has been started....` followed by a list of nodes from 8 to 19.

Terminal 3 (Bottom Left): Shows the output of `ros2 run my_package subspubs`. It prints: `[INFO] [1678384257.158036717] [Subscriber]: Hello from subscriber...please subscribe to my channel` followed by a list of subscribers from 8 to 18.

Node Graph (Bottom Right): A window titled `rqt_graph_RosGraph - rqt` showing a graph with three nodes: `/Publisher`, `/Subscriber`, and `/Three_node`. The connections are: `/Publisher` to `/Subscriber` via `/message`, and `/Subscriber` to `/Three_node` via `/msg_count`.

Exercise 3: Write a ROS2 program to move the turtle in different shapes.

(i) Square (ii) Circle (iii) Triangle (iv) Spiral

Square

CODE SNIPPET

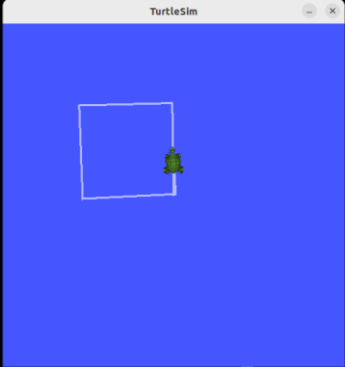
```
turtsqr.py X
src > my_package > my_package > turtsqr.py > ...
1  #!/usr/bin/env python3
2  import rclpy
3  from rclpy.node import Node
4  from turtlesim.msg import Pose
5  from geometry_msgs.msg import Twist
6  import math
7  import time
8  # from turtle_control import TurtleControllerNode
9  class traingle(Node):
10     def __init__(self):
11         super().__init__("turtle_controller")
12         self.length_y = 2.0
13         self.length_x = 3.0
14         self.theta = math.pi
15         self.pose_ = None
16         self.cmd_vel_publisher_ = self.create_publisher(Twist, "turtle1/cmd_vel", 10)
17         self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)
18
19     def control_loop(self):
20         msg = Twist()
21         msg.linear.x = 0.0
22         msg.angular.z = (math.pi/2)
23         if msg.angular.z == math.pi/2:
24             self.cmd_vel_publisher_.publish(msg)
25             time.sleep(2)
26             msg.angular.z = 0.0
27         msg.linear.x = self.length_x
28         # msg.linear.y = self.length_y
29         self.cmd_vel_publisher_.publish(msg)
30         time.sleep(1)
31         msg.angular.z = math.pi/2
32         msg.linear.x = 0.0
33         self.cmd_vel_publisher_.publish(msg)
34         time.sleep(2)
35         if msg.angular.z == math.pi/2:
36             msg.angular.z = 0.0
37         msg.linear.x = self.length_x
38         self.cmd_vel_publisher_.publish(msg)
39         time.sleep(1)
40         msg.angular.z = math.pi/2
41         msg.linear.x = 0.0
42         self.cmd_vel_publisher_.publish(msg)
43         time.sleep(2)
44
45         if msg.angular.z == math.pi/2:
46             msg.angular.z = 0.0
47             msg.linear.x = self.length_x
48             self.cmd_vel_publisher_.publish(msg)
49             time.sleep(2)
50
51     def main(args=None):
52         rclpy.init(args=args)
53         tri = traingle()
54         rclpy.spin(tri)
55         rclpy.shutdown()
56
57 if __name__ == "__main__":
58     main()
```

RESULT

```
adithya@adithya-virtual-machine: ~
adithya@adithya-virtual-machine:~$ colcon build --packages-select my_package
Starting >>> my_package
Finished <<< my_package [1.15s]

Summary: 1 package finished [1.41s]
adithya@adithya-virtual-machine:~$ source install/setup.bash
adithya@adithya-virtual-machine:~$ ros2 run my_package sqr
[]

adithya@adithya-virtual-machine:~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland a
nyway.
[INFO] [1678386491.614932623] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1678386491.619707456] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445],
theta=[0.000000]
```



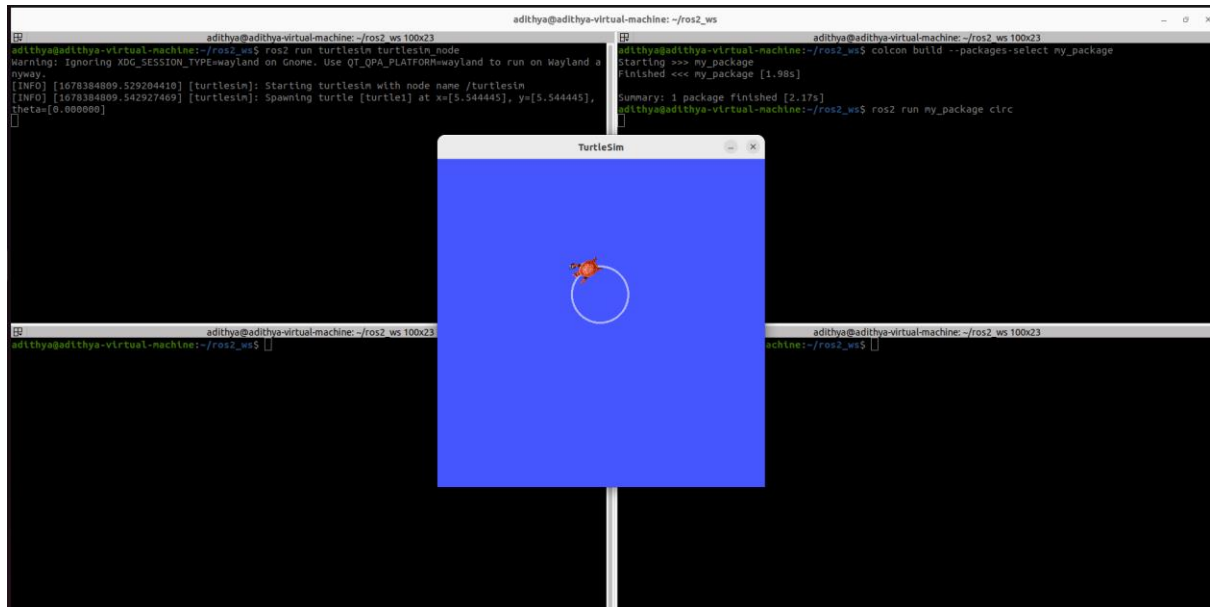
The image shows a terminal window with two panes. The left pane shows the output of a colcon build command, indicating that the package 'my_package' was successfully built. The right pane shows the output of the 'ros2 run turtlesim turtlesim_node' command, which starts the turtlesim simulation. Below the terminal panes, a 'Turtlesim' window is displayed. It features a blue background with a white square outline. A small green turtle icon is positioned at the bottom-right corner of the square.

Circle

CODE SNIPPET

```
turtcircle.py x
src > my_package > my_package > turtcircle.py > ...
1  #!/usr/bin/env python3
2
3  import rclpy
4  from rclpy.node import Node
5  from turtlesim.msg import Pose
6  from geometry_msgs.msg import Twist
7  import math
8  # from turtle_control import TurtleControllerNode
9
10 class traingle(Node):
11     def __init__(self):
12         super().__init__("turtle_controller")
13         self.length_y = 3.0
14         self.theta = math.pi
15         self.pose_ = None
16         self.cmd_vel_publisher_ = self.create_publisher(Twist, "turtle1/cmd_vel", 10)
17         self.pose_subscriber_ = self.create_subscription(Pose, "turtle1/pose", self.callback_turtle_pose, 10)
18         self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)
19     def callback_turtle_pose(self, msg):
20         self.pose_ = msg
21
22     def control_loop(self):
23         if self.pose_ == None:
24             return
25
26         msg = Twist()
27         msg.linear.x = self.length_y
28         msg.angular.z = self.theta
29         self.cmd_vel_publisher_.publish(msg)
30
31
32
33
34 def main(args=None):
35     rclpy.init(args=args)
36     tri = traingle()
37     rclpy.spin(tri)
38     rclpy.shutdown()
39
40 if __name__ == "__main__":
41     main()
```

RESULT



The screenshot displays a ROS2 environment with multiple terminal windows and a TurtleSim window. The top-left terminal window shows the command `ros2 run turtlesim turtlesim_node` being executed, with output indicating the start of the turtlesim node. The top-right terminal window shows the command `colcon build --packages-select my_package` being executed, with output indicating the build process. The bottom-left terminal window shows the command `ros2 run my_package circ` being executed, with output indicating the start of the `circ` node. The bottom-right terminal window shows the command `ros2 run my_package circ` being executed, with output indicating the start of the `circ` node. The central TurtleSim window shows a blue square environment with a small orange turtle icon and a white circle, representing the turtle's path.

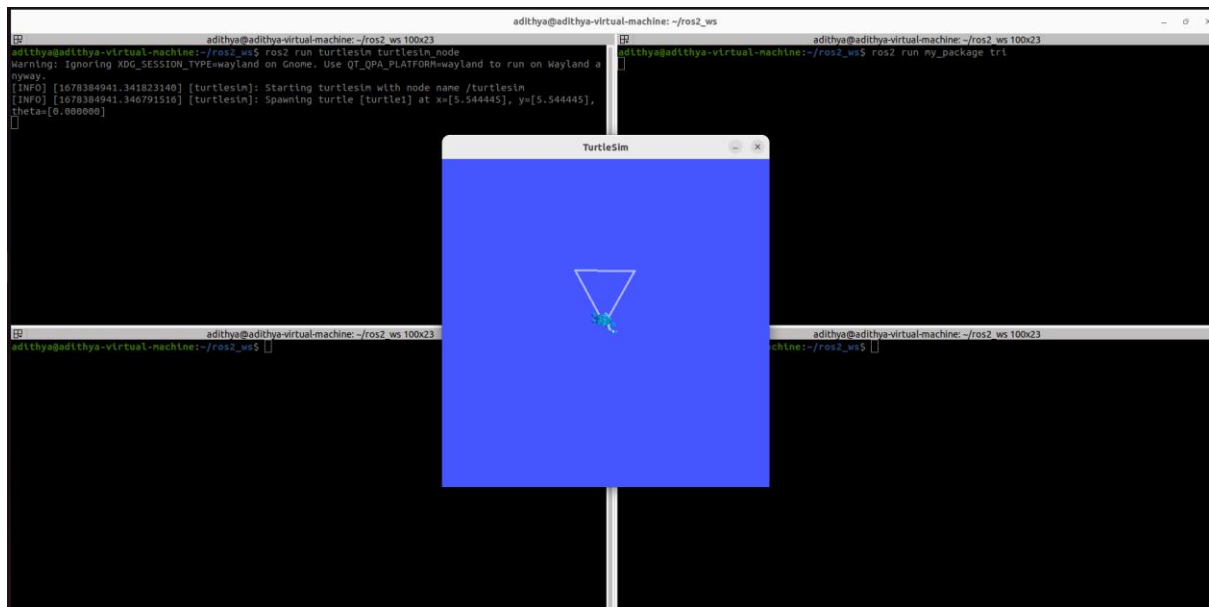
```
adithya@adithya-virtual-machine: ~/ros2_ws
adithya@adithya-virtual-machine:~/ros2_ws$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland
[INFO] [1678384809.529204410] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1678384809.542927469] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445],
theta=[0.888000]
adithya@adithya-virtual-machine:~/ros2_ws$ colcon build --packages-select my_package
Starting >>> my_package
Finished <<< my_package [1.98s]
Summary: 1 package finished [2.17s]
adithya@adithya-virtual-machine:~/ros2_ws$ ros2 run my_package circ
adithya@adithya-virtual-machine:~/ros2_ws$
```

Triangle

CODE SNIPPET

```
turttria.py x
src > my_package > my_package > turttria.py > ...
1  #!/usr/bin/env python3
2  import rclpy
3  from rclpy.node import Node
4  from turtlesim.msg import Pose
5  from geometry_msgs.msg import Twist
6  import math
7  import time
8  # from turtle_control import TurtleControllerNode
9  class traingle(Node):
10     def __init__(self):
11         super().__init__("turtle_controller")
12         self.length_x = 2.0
13         self.length_y = 2.0
14         self.theta = math.pi
15         self.pose = None
16         self.cmd_vel_publisher_ = self.create_publisher(Twist, "turtle1/cmd_vel", 10)
17         self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)
18
19     def control_loop(self):
20         msg = Twist()
21         msg.linear.x = 0.0
22         msg.angular.z = (math.pi/3)
23         if msg.angular.z == math.pi/3:
24             self.cmd_vel_publisher_.publish(msg)
25             time.sleep(2)
26             msg.angular.z = 0.0
27         msg.linear.x = self.length_x
28         # msg.linear.y = self.length_y
29         self.cmd_vel_publisher_.publish(msg)
30         time.sleep(1)
31         msg.angular.z = math.pi/1.525
32         msg.linear.x = 0.0
33         self.cmd_vel_publisher_.publish(msg)
34         time.sleep(2)
35         if msg.angular.z == math.pi/1.525:
36             msg.angular.z = 0.0
37         msg.linear.x = self.length_x
38         self.cmd_vel_publisher_.publish(msg)
39         time.sleep(1)
40         msg.angular.z = math.pi/1.525
41         msg.linear.x = 0.0
42         self.cmd_vel_publisher_.publish(msg)
43         time.sleep(2)
44
45
46     msg.linear.x = self.length_x
47     self.cmd_vel_publisher_.publish(msg)
48     time.sleep(2)
49 def main(args=None):
50     rclpy.init(args=args)
51     tri = traingle()
52     rclpy.spin(tri)
53     rclpy.shutdown()
54
55 if __name__ == "__main__":
56     main()
57
```

RESULT



Spiral

CODE SNIPPET

```
turtspiral.py X
src > my_package > my_package > turtspiral.py > ...
1  #!/usr/bin/env python3
2
3  import rclpy
4  from rclpy.node import Node
5  from turtlesim.msg import Pose
6  from geometry_msgs.msg import Twist
7  from turtlesim.srv import Spawn
8  from functools import partial
9  import math
10 import random
11 import time
12
13 class spiral(Node):
14     def __init__(self):
15         super().__init__("Spiral_Motion")
16         self.velocity_publisher_ = self.create_publisher(Twist, "/turtle1/cmd_vel", 10)
17         self.pose_subscriber_ = self.create_subscription(Pose, "/turtle/pose", self.pose_callback, 10)
18         self.pose_x = 0
19         self.pose_y = 0
20         self.timer_ = self.create_timer(1.0, self.publish_velocity)
21         self.get_logger().info("turtle halloocination.....into the spiral")
22         self.radius = 2.0
23
24     def pose_callback(self, pose_msg=Pose()):
25         self.pose_x = pose_msg.x
26         self.pose_y = pose_msg.y
27
28         if(round(self.pose_y)==8.0):
29             self.get_logger().warn("About to hit wall!")
30
31     def publish_velocity(self):
32         vel_msg = Twist()
33         vel_msg.linear.x = self.radius
34
35         if self.pose_x < 10.0 and self.pose_y < 10.0:
36             vel_msg.angular.z = 22/7
37             self.radius += 0.2
38         else:
39             vel_msg.angular.z = 0.0
40             vel_msg.linear.x = 0.0
41
42             self.get_logger().info("done rotation...")
43             self.timer_.cancel()
```

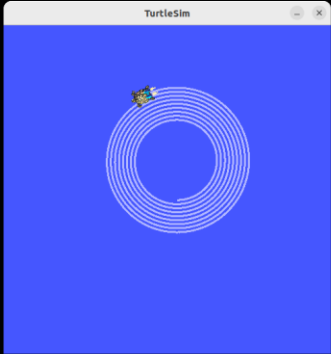
```
turtspiral.py X
src > my_package > my_package > turtspiral.py > ...
45         self.velocity_publisher_.publish(vel_msg)
46
47     def main(args=None):
48         rclpy.init(args=args)
49         node = spiral()
50         rclpy.spin(node)
51         rclpy.shutdown()
52
53     if __name__ == "__main__":
54
55         main()
```

RESULT

```
adithya@adithya-virtual-machine: ~
adithya@adithya-virtual-machine:~$ colcon build --packages-select my_package
Starting >>> my_package
Finished <<< my_package [1.14s]

Summary: 1 package finished [1.40s]
adithya@adithya-virtual-machine:~$ source install/setup.bash
adithya@adithya-virtual-machine:~$ ros2 run my_package spir
[INFO] [1678387656.171614668] [spiral_motion]: turtle hallucination.....into the spiral

adithya@adithya-virtual-machine:~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland a
nyway.
[INFO] [1678386491.614932623] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1678386491.619707456] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445],
theta=[0.000000]
adithya@adithya-virtual-machine:~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland a
nyway.
[INFO] [1678387562.203924194] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1678387562.208623771] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445],
theta=[0.000000]
adithya@adithya-virtual-machine:~$ ros2 run turtlesim turtlesim_node
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland to run on Wayland a
nyway.
[INFO] [1678387666.490176144] [turtlesim]: Starting turtlesim with node name /turtlesim
[INFO] [1678387666.495202108] [turtlesim]: Spawning turtle [turtle1] at x=[5.544445], y=[5.544445],
theta=[0.000000]
```



Exercise 4: Write a ROS2 code to spawn a new turtle (default: turtle2) in a random location. Modify the code to move the turtle1 to turtle2 location and kill it. Use PID algorithm to navigate the turtle.

CODE SNIPPET

Spawning

```
spawn.py x killit.py sample.py ... spawn.py x
src > my_package > my_package > spawn.py > spawn_kill
1  #!/usr/bin/env python3
2  import rclpy
3  from rclpy.node import Node
4  from turtlesim.msg import Pose
5  from geometry_msgs.msg import Twist
6  from turtlesim.srv import Spawn
7  from functools import partial
8  import math
9  import random
10 import time
11 class spawn_kill(Node):
12     def __init__(self):
13         super().__init__("Turtle Kill")
14         self.cmd_vel_publisher_ = self.create_publisher(Twist, "turtle1/cmd_vel", 10)
15         self.pose_1 = self.create_subscription(Pose, '/turtle1/pose', self.pose1_callback, 10)
16         self.pose_2 = self.create_subscription(Pose, '/turtle2/pose', self.pose2_callback, 10)
17         self.pose1 = None
18         self.pose2 = None
19         self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)
20     def pose1_callback(self, msg):
21         self.pose1 = msg
22     def pose2_callback(self, msg02):
23         self.pose2 = msg02
24     def control_loop(self):
25         if self.pose1 == None and self.pose2 == None:
26             return
27         dist_x = self.pose2.x - self.pose1.x
28         dist_y = self.pose2.y - self.pose1.y
29         distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)
30         goal_theta = math.atan2(dist_y, dist_x)
31         diff = goal_theta - self.pose2.theta
32         msg = Twist()
33         if distance > 0.2:
34             msg.angular.z = diff
35             self.cmd_vel_publisher_.publish(msg)
36             time.sleep(1)
37             msg.angular.z = 0.0
38
39         msg.linear.x = distance
40         self.cmd_vel_publisher_.publish(msg)
41         time.sleep(1)
42     else:
43         msg.angular.z = 0.0
44         msg.linear.x = 0.0
45         self.cmd_vel_publisher_.publish(msg)
46         time.sleep(2)
47         node = rclpy.create_node('node01')
48         node.destroy_node()
49         rclpy.shutdown()
50         time.sleep(2)
51 def main(args=None):
52     rclpy.init(args=args)
53     node_a = spawn_kill()
54     rclpy.spin(node_a)
55     rclpy.shutdown()
56 if __name__ == "__main__":
57     main()
```

Killing

```
spawn.py  killit.py  sample.py
src > my_package > my_package > killit.py > spawn
1  #!/usr/bin/env python3
2  import rclpy
3  from rclpy.node import Node
4  from turtlesim.msg import Pose
5  from geometry_msgs.msg import Twist
6  from turtlesim.srv import Spawn
7  from functools import partial
8  import math
9  import random
10 class spawn(Node):
11     def __init__(self):
12         super().__init__("node01")
13         self.get_logger().info("Hello Node started for spawn program...")
14     # self.pose_subscriber =
15     self.create_subscription(Pose, "/turtle1/pose", self.pose_callback, 1)
16     self.client_for_spawn()
17     def client_for_spawn(self, x = random.uniform(0.5,10), y = random.uni
18         client = self.create_client(Spawn, "/spawn")
19         while not client.wait_for_service(1.0):
20             self.get_logger().warn("Waiting for service ....." )
21             request = Spawn.Request()
22             request.x = x
23             request.y = y
24             request.theta = theta
25             future = client.call_async(request)
26     def main(args=None):
27         rclpy.init(args=args)
28     # global node01
29     node01 = spawn()
30     rclpy.spin(node01)
31     rclpy.shutdown()
32     if __name__ == "__main__":
33         main()
```

RESULT