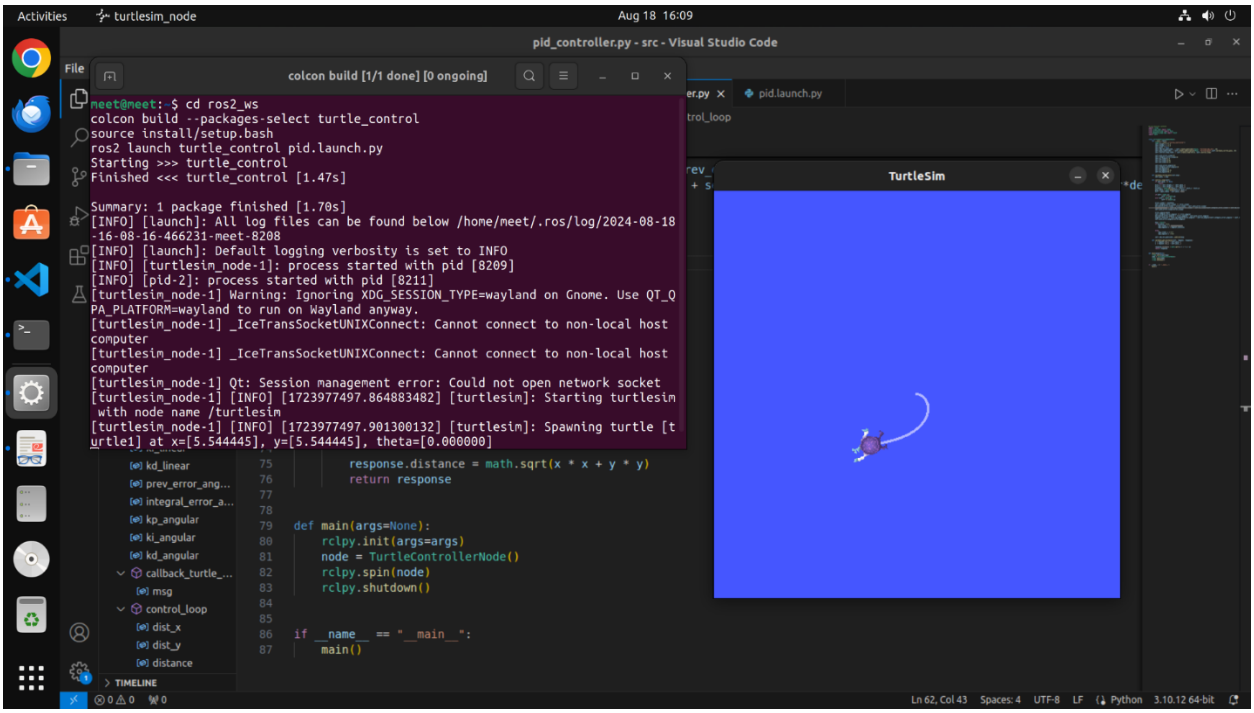
Lab 3 – Exercise

Meet Kansara – 220929270 Roll no. 54

Exercise: Building PID controller and launching with Turtlesim.

Code execution and analysis:

```python
#!/usr/bin/env python3
import rclpy
from rclpy.node import Node
from turtlesim.msg import Pose
from geometry_msgs.msg import Twist
import math


class TurtleControllerNode(Node):
    def __init__(self):
        super().__init__("turtle_controller")
        self.target_x = 4.0
        self.target_y = 4.0
        self.pose_ = None
        self.cmd_vel_publisher_ = self.create_publisher(Twist, "turtle1/cmd_vel", 10)
        self.pose_subscriber_ = self.create_subscription(Pose, "turtle1/pose",self.callback_turtle_pose, 10)
        self.control_loop_timer_ = self.create_timer(0.01, self.control_loop)

        self.prev_error_linear=0
        self.integral_error_linear=0
        self.kp_linear=1.5
        self.ki_linear=0.02
        self.kd_linear=0.15

        self.prev_error_angular=0
        self.integral_error_angular=0
        self.kp_angular=5
        self.ki_angular=0.02
        self.kd_angular=0.2

    def callback_turtle_pose(self,msg):
        self.pose_ = msg

    def control_loop(self):
        if self.pose_ == None:
            return
        dist_x = self.target_x - self.pose_.x
        dist_y = self.target_y - self.pose_.y
        distance = math.sqrt(dist_x * dist_x + dist_y * dist_y)
        goal_theta = math.atan2(dist_y, dist_x)
        diff = goal_theta - self.pose_.theta

        if diff > math.pi:
            diff -= 2*math.pi
        elif diff < -math.pi:
            diff += 2*math.pi

        error_linear = distance
        self.integral_error_linear += error_linear
        derivative_error_linear = error_linear - self.prev_error_linear
        linear_velocity = (self.kp_linear*error_linear + self.ki_linear*self.integral_error_linear + self.kd_linear*derivative_error_linear)
        self.prev_error_linear=error_linear

        error_angular=diff
        self.integral_error_angular += error_angular
        derivative_error_angular = error_angular - self.prev_error_angular
        angular_velocity = (self.kp_angular*error_angular + self.ki_angular*self.integral_error_angular + self.kd_angular*derivative_error_angular)
        self.prev_error_angular=error_angular

        msg = Twist()
        if distance > 0.5:
            msg.linear.x = linear_velocity
            msg.angular.z = angular_velocity

        else:
            msg.linear.x = 0.0
            msg.angular.z = 0.0

        self.cmd_vel_publisher_.publish(msg)

    def callback_get_distance(self, request, response):
        x = request.loc_x - self.pose_.x
        y = request.loc_y - self.pose_.y

        response.distance = math.sqrt(x * x + y * y)
        return response


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


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



Conclusion: Successful execution and integration of Turtlesim with PID controller has been achieved.