Documentation of Sensor-Based TurtleSim Movement Task

1. Introduction

The objective of this task was to utilize a sensor to control a robot's movements based on specific conditions gotten from sensor signals. It was aimed to test understanding and application of ROS2 concepts including nodes, messages, publishers, subscribers and interfacing with TurtleSim. I created a subscriber node that interprets sensor signals and commands the TurtleSim turtle to move or turn based on predefined rules.

2. Task Breakdown

According to the description, the sensor's purpose is to measure the environment and provide an encoded signal about its surroundings. Sensor provides three types of values:

- Negative values: Indicating the robot needs to make a **120-degree right turn**.
- Positive odd values: Indicating the robot needs to **move forward by 2 units**.
- Positive even values: Indicating the robot needs to **move forward by 1 unit**.

i) Subtasks and solution approach Installing Sensor Package:
I first unzipped the provided package and installed it in my ROS2 workspace. Then I ran it using this command:

```
ros2 run midterm_sensor_1 sensor
```

After this, I checked the topics which are being published using these commands:

```
ros2 topic list

ros2 topic info /sensor_signal
```

With these, I gathered details about the message type and the topics published by the sensor. I identified that the sensor sends integer values through the topic `/sensor1_signal` with messages of type `std_msgs/msg/Int32`.

Creating my Neptun Package:
I then created a ROS2 package named `mflgtk` to contain my solution nodes and logic. The package was structured using colcon to ensure correct dependency management.

Node Implementation:
I created a subscriber to listen to the sensor's topic `/sensor1_signal`. Based

on the value received, it calculates the necessary commands and publishes them to the TurtleSim node. The logic was:

- If the received value was negative, the robot executes a right turn using the angle `-2.0944` radians (equivalent to 120 degrees).
- If the received value was a positive odd number, the robot moves forward with a velocity of `4.0` units.
- If the received value was a positive even number, the robot moves forward with a velocity of `2.0` units.

Testing with TurtleSim:
I executed using the command `ros2 run mflgtk sensor_subscriber`. TurtleSim was used to visualize the movements, and the robot correctly responded to the sensor's outputs by creating a visual pattern of movements and turns in line with the provided instructions.

3. Results and Visualization

The final pattern drawn by the TurtleSim demonstrated triangular shapes formed by the turtle making right turns and moving forward by different units, confirming that the logic was implemented correctly.

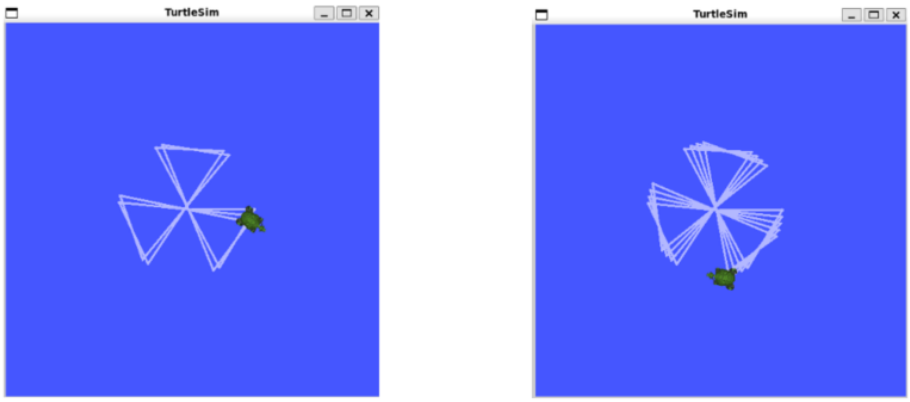

Figure 1: PrintScreen

4. Conclusion

This task successfully demonstrated the use of ROS2 features to interpret sensor signals and command a robot. The visual output in TurtleSim validated the correctness of our implementation and achieved the desired results.