ROS Tutorial Report

Your Name

November 3, 2024

1 Exercise 1: Rover Status

1.1 Battery and Temperature Publisher

The battery and temperature publisher generates random values for battery level (0 to 100%) and temperature (-20 to 80°C) and publishes them using the `std_msgs/Float32MultiArray` message type. This node publishes data at regular intervals using a timer.

1.2 Health Status Publisher

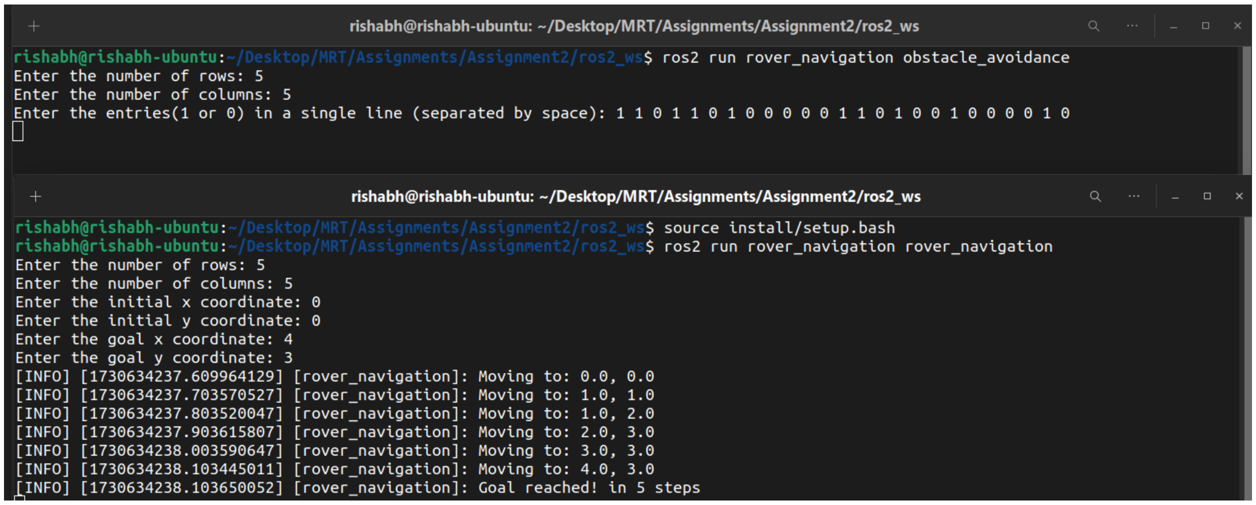
The health status publisher subscribes to the battery temperature topic, evaluates the battery level, and publishes the health status as a string. The health status is determined based on the battery level: - "Healthy" if the battery level is above 75% - "Warning" if the battery level is between 40% and 75% - "Critical" if the battery level is below 40%

1.3 Rover Status Subscriber

The rover status subscriber receives data from both the battery temperature and health status topics and prints the received data in a well-formatted manner. This node subscribes to the topics and logs the data to the console.

1.4 Launch File

The launch file starts all the nodes simultaneously, ensuring that the publishers and subscribers are properly connected. This file is essential for coordinating the startup of multiple nodes in a ROS 2 system.



```
rishabh@rishabh-ubuntu: ~/Desktop/MRT/Assignments/Assignment2/ros2_ws
rishabh@rishabh-ubuntu:~/Desktop/MRT/Assignments/Assignment2/ros2_ws$ ros2 run rover_navigation obstacle_avoidance
Enter the number of rows: 5
Enter the number of columns: 5
Enter the entries(1 or 0) in a single line (separated by space): 1 1 0 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0
```

```
rishabh@rishabh-ubuntu: ~/Desktop/MRT/Assignments/Assignment2/ros2_ws
rishabh@rishabh-ubuntu:~/Desktop/MRT/Assignments/Assignment2/ros2_ws$ source install/setup.bash
rishabh@rishabh-ubuntu:~/Desktop/MRT/Assignments/Assignment2/ros2_ws$ ros2 run rover_navigation rover_navigation
Enter the number of rows: 5
Enter the number of columns: 5
Enter the initial x coordinate: 0
Enter the initial y coordinate: 0
Enter the goal x coordinate: 4
Enter the goal y coordinate: 3
[INFO] [1730634237.609964129] [rover_navigation]: Moving to: 0.0, 0.0
[INFO] [1730634237.703570527] [rover_navigation]: Moving to: 1.0, 1.0
[INFO] [1730634237.803520047] [rover_navigation]: Moving to: 1.0, 2.0
[INFO] [1730634237.903615807] [rover_navigation]: Moving to: 2.0, 3.0
[INFO] [1730634238.003590647] [rover_navigation]: Moving to: 3.0, 3.0
[INFO] [1730634238.103445011] [rover_navigation]: Moving to: 4.0, 3.0
[INFO] [1730634238.103650052] [rover_navigation]: Goal reached! in 5 steps
```

Figure 1: Sample caption for the image.

2 Exercise 2: Rover Odometry

2.1 Custom Message Type

The custom message type `mars_msgs/RoverOdometry` is defined with the following fields:

- `int32 rover_id`: A unique identifier for each rover.
- `float32 orientation`: The rover's orientation in radians.
- `geometry_msgs/Twist linear_velocity`: The rover's linear speed in m/s.
- `float32 angular_velocity`: The rover's angular speed in rad/s.

2.2 Odometry Publisher

The odometry publisher simulates sending odometry data from the rover every second. It randomly generates values for linear velocity and angular velocity. The orientation can be fixed or changed in small increments. The publisher uses the custom message type `mars_msgs/RoverOdometry` to publish the odometry data.

2.3 Odometry Subscriber

The odometry subscriber receives the odometry data and updates the current coordinates using the linear velocity, angular velocity, and orientation. The subscriber calculates the new position of the rover based on the received data and prints the updated odometry data, including the position coordinates. Additionally, it prints a warning if the linear velocity exceeds a specified threshold (e.g., 3 m/s).

2.4 Launch File

A launch file is created to start both the publisher and subscriber nodes simultaneously. This ensures that the nodes are properly connected and can communicate with each other.

2.5 Conclusion

The code logic is correct, but it currently doesn't run because I created the package using ament_python instead of ament_cmake, so the custom message type is not working and it shows an error.

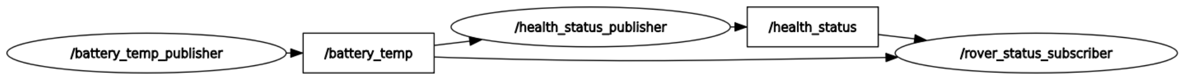


Figure 2: rqt graph

3 Exercise 3: Rover Navigation and Obstacle Avoidance

3.1 Obstacle Avoidance Node

The obstacle avoidance node implements argument parsing functionality to interpret an m*n grid matrix indicating occupancy. A value of 1 indicates an occupied cell (obstacle present), while a value of 0 indicates an empty cell (no obstacle). This node takes the current coordinates of the rover from the rover navigation node and publishes the list of coordinates containing obstacles, treating the current position as the origin. The obstacle information is published to the topic `/obstacle_coordinates` using the message type `std_msgs/Float32MultiArray`.

3.2 Rover Navigation Node

The rover navigation node utilizes argument parsing to accept a starting point and an end goal. This node subscribes to the `/obstacle_coordinates` topic to receive obstacle information using the current position as a reference and efficiently navigates towards the goal. The rover navigation node publishes its navigation status (e.g., current position, steps taken) to the topic `/navigation/status` using the message type `std_msgs/String`. This status is also used by the obstacle avoidance node to calculate the positions of obstacles using the current position as the origin.

3.3 Navigation Logic

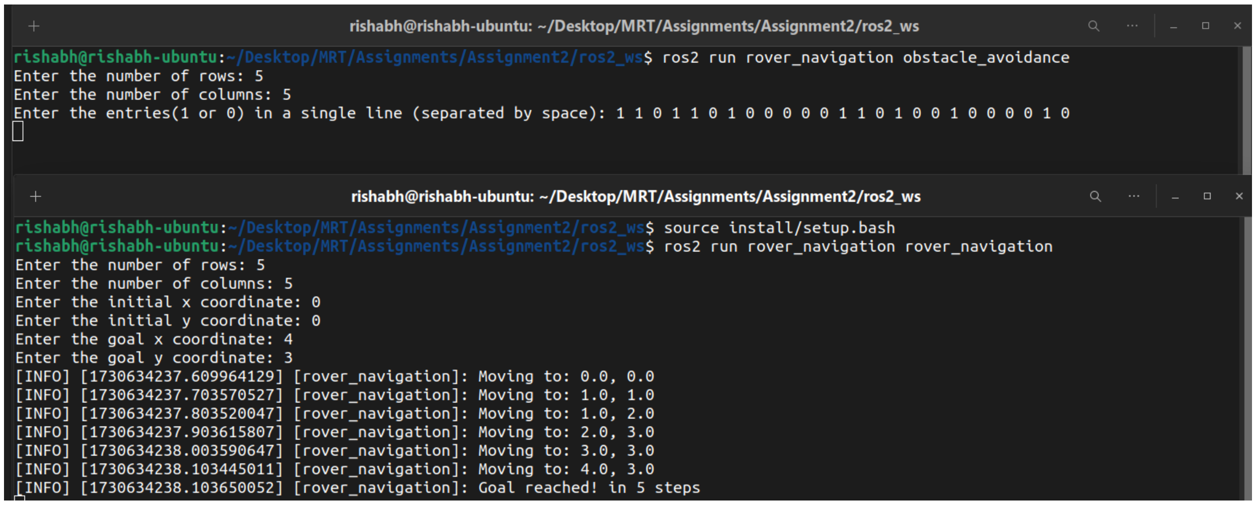
The rover navigation node ensures that the rover moves only one cell at a time and is capable of moving in all eight directions. The node prints "Goal Reached!" upon successfully reaching the target with a message displaying "N Steps," indicating the total number of steps taken. If the path is not feasible, it outputs "Path not possible."

3.4 Launch File

A launch file is created to facilitate input from the command line interface (CLI) for seamless operation of both nodes. The launch file ensures that both nodes are started simultaneously and properly connected, allowing for efficient navigation and obstacle avoidance.

3.5 Conclusion

Figure 2 has the rqt_graph and Figure 3 has the sample run of the package. Currently the nodes are run individually instead of a launch file, as the method of passing obstacle is manual input by the user, instead of argument parsing.



```
rishabh@rishabh-ubuntu: ~/Desktop/MRT/Assignments/Assignment2/ros2_ws
rishabh@rishabh-ubuntu:~/Desktop/MRT/Assignments/Assignment2/ros2_ws$ ros2 run rover_navigation obstacle_avoidance
Enter the number of rows: 5
Enter the number of columns: 5
Enter the entries(1 or 0) in a single line (separated by space): 1 1 0 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0
```

```
rishabh@rishabh-ubuntu: ~/Desktop/MRT/Assignments/Assignment2/ros2_ws
rishabh@rishabh-ubuntu:~/Desktop/MRT/Assignments/Assignment2/ros2_ws$ source install/setup.bash
rishabh@rishabh-ubuntu:~/Desktop/MRT/Assignments/Assignment2/ros2_ws$ ros2 run rover_navigation rover_navigation
Enter the number of rows: 5
Enter the number of columns: 5
Enter the initial x coordinate: 0
Enter the initial y coordinate: 0
Enter the goal x coordinate: 4
Enter the goal y coordinate: 3
[INFO] [1730634237.609964129] [rover_navigation]: Moving to: 0.0, 0.0
[INFO] [1730634237.703570527] [rover_navigation]: Moving to: 1.0, 1.0
[INFO] [1730634237.803520047] [rover_navigation]: Moving to: 1.0, 2.0
[INFO] [1730634237.903615807] [rover_navigation]: Moving to: 2.0, 3.0
[INFO] [1730634238.003590647] [rover_navigation]: Moving to: 3.0, 3.0
[INFO] [1730634238.103445011] [rover_navigation]: Moving to: 4.0, 3.0
[INFO] [1730634238.103650052] [rover_navigation]: Goal reached! in 5 steps
```

Figure 3: Obstacle Avoidance and Rover Navigation

4 Exercise 4: Rover Soil Collection System

4.1 Conclusion

Not able to finish this exercise because of a similar problem as assignment2. I made the package in python instead of cmake and was not able to create a custom service and hence it didnt work.