Differential Drive Robot – ROS 2 Mapping & Navigation

Project Report

Challa Santhosh

---

1. Environment Setup

The project was developed using the following environment:

- Operating System: Ubuntu 24.04 LTS
- ROS Version: ROS 2
- Simulation Tool: Gazebo
- Visualization Tool: RViz2
- Key Packages: SLAM Toolbox, Navigation2 (Nav2), robot_state_publisher, diff_drive_controller

ROS 2 was installed on Ubuntu 24.04 by following the official ROS documentation step-by-step. After successfully configuring the environment, the project repository was cloned:

Repository:
https://github.com/Challa200Santhosh/Diff_Drive_Robot

The workspace was built using:

```
colcon build
source install/setup.bash
```

Before launching the simulation, all required dependencies were verified and installed properly to ensure a stable setup.

2. Mapping and Navigation Process

2.1 Simulation

The robot was launched successfully in Gazebo simulation.
The following were verified:

- Proper spawning of the robot model
- Correct publishing of sensor topics
- Correct TF frame connections

2.2 Mapping (SLAM)

Mapping was performed using **SLAM Toolbox (online mode)**.

- The robot was manually controlled to explore the simulated environment.
- A 2D occupancy grid map was generated in real-time.
- The generated map was saved for future navigation use.

This step ensured accurate environmental representation for autonomous navigation.

2.3 Autonomous Navigation

After successfully generating the map:

- The saved map was loaded into the Navigation2 stack.
- Localization was configured using AMCL.
- Global and local planners were activated.
- Goal positions were set in RViz.

The robot successfully navigated autonomously to multiple target locations without manual intervention.

3. Problems Faced and Solutions

During implementation, several practical challenges were encountered and resolved.

3.1 Missing Dependency Packages

Issue:
colcon build frequently failed due to missing ROS 2 dependencies.

Resolution:

- Installed missing packages manually when required.
- Carefully analyzed terminal error logs.
- Rebuilt the workspace multiple times until all dependencies were correctly resolved.

3.2 Command-Line Errors

Issue:
Several errors were caused by:

- Incorrect command syntax
- From the beginning the process needs to be perfect otherwise it wont work.
- Wrong launch file arguments
- Forgetting to source the workspace

Resolution:

- Rechecked command syntax carefully.
- Ensured source install/setup.bash was executed before running commands.
- Verified file paths and launch arguments.
- Repeated execution multiple times until commands worked correctly.

Most command-line errors were resolved through careful verification and repeated trials.

3.3 Gazebo Launch Errors

Issue:
Robot did not spawn properly or simulation failed to start.

Resolution:

- Verified URDF/Xacro file paths.
- Checked Gazebo plugin configurations.
- Confirmed correct launch file structure.
- Revalidated environment setup.

3.4 TF and Frame Mismatch Errors

Issue:
Frame connection issues between map, odom, and base_link.

Resolution:

- Verified frame names in configuration files.
- Checked TF tree.
- Corrected frame IDs in SLAM and Navigation2 configuration files.
- Ensured consistency across all nodes.

3.5 Navigation Instability

Issue:
Robot rotating continuously or failing to reach goal positions.

Resolution:

- Tuned costmap and controller parameters.
- Verified localization accuracy.
- Restarted lifecycle nodes when necessary.
- Compared configurations with official documentation.

4. Approach to Problem Solving

Most challenges were resolved through:

- Careful analysis of terminal error messages
- Repeated execution and testing
- Referring back to official ROS 2 documentation
- Comparing configurations with reference repositories
- Patience and systematic debugging

Repeated testing and documentation reference played a key role in successfully completing the project.

5. Final Outcome

✓ ROS 2 successfully configured on Ubuntu 24.04
✓ Robot spawned correctly in Gazebo
✓ Map generated using SLAM Toolbox
✓ Map saved successfully
✓ Autonomous navigation achieved using Navigation2
✓ Launch and configuration files properly organized in the repository

This report presents a complete implementation of a Differential Drive Robot capable of mapping and autonomous navigation using ROS 2 in a simulated environment.