Reflection: TurtleBot3 Automation Project

Introduction

This project focused on building an automated TurtleBot3 system using ROS2. The main objective was to integrate different components such as setup automation, maintenance monitoring, navigation, object detection, and an additional custom feature. At first, I thought the project would mainly involve writing ROS2 code, but during the process I realized that understanding how different ROS2 components work together was just as important as coding.

Challenges Encountered

One of the first difficulties I faced was managing the ROS2 environment. I often forgot to source the required setup files, which caused commands or nodes to not work as expected. At that time, it was confusing because there were no clear error messages, and I initially thought there were problems in my code. Later, I understood that ROS2 heavily depends on correct environment configuration.

Another major challenge was object detection. My initial plan was to use camera data with OpenCV. However, during testing I found that the TurtleBot3 simulation did not actually publish camera image data, even though the topic existed. This caused the object detection node to run without producing any output. I spent time checking topics and node connections before realizing that the problem was not the code itself, but the availability of sensor data.

Navigation also required attention. Several components such as Gazebo, SLAM, Nav2, and custom nodes needed to be launched in the correct order. If one of them was missing, the robot would not move or respond correctly. Understanding this dependency took time and multiple trials.

Solutions and Learning Process

To solve the detection issue, I changed the design from camera-based detection to LiDAR-based detection using the `/scan` topic. By analyzing LaserScan distance values and grouping nearby points, I was able to detect obstacle clusters reliably. This solution worked better with the available simulation setup and simplified the overall system.

During debugging, I learned how important ROS2 command-line tools are. Commands such as `ros2 node list`, `ros2 topic list`, `ros2 topic info`, and `ros2 topic echo` helped me understand what was actually running in the system. These tools made it easier to identify whether problems were caused by missing publishers, inactive nodes, or incorrect assumptions.

The project also helped me understand the importance of modular design. Each feature was implemented as a separate node, which made debugging and testing easier. The safety stop feature showed how detection results can directly affect robot behavior by stopping the robot when obstacles are detected.

Lessons Learned

This project showed me that robotics development involves a lot of trial and error. Many issues were related to system configuration rather than programming mistakes. I learned that it is important to verify sensor data, topic publishers, and node connections before assuming there is a problem in the code.

I also learned to adapt my approach when initial plans did not work. Switching from camera-based detection to LiDAR-based detection was an important lesson in being flexible and practical. Overall, this project improved my understanding of ROS2 and gave me more confidence in working with robotic systems that involve multiple interacting components.