AUTONOM ROBOTS PROJECT

Mehmet Ali Sevdinoğlu-2022555464
Arda Mülayim-201955056
Erdem Daud- 2018555012

CONTENTS

1.0 INTRODUCTION

This project aims to enable a mobile robot to perform random exploration by detecting and avoiding obstacles in its environment using laser scanner (Lidar) data. The robot analyzes its environment without adhering to a specific map and moves in directions where there are no obstacles.

1.1 UBUNTU

We developed our robot using the Ubuntu 20.04 version as the operating system, carefully integrating it with ROS (Robot Operating System) for seamless functionality. To simulate and visualize the robot's behavior, we utilized the powerful simulation platforms GAZEBO for dynamic environment interaction and RViz for real-time sensor and navigation visualization, creating a robust testing and development ecosystem. the obstacle avoidance and random exploration algorithm was developed using Python.

1.2 GAZEBO

In our project we usedGazebo to our simulation enviroment. Gazebo is a powerful simulation tool used to create realistic environments for testing and developing robots. It provides a physics engine for accurate motion dynamics, supports various sensors like Lidar and cameras, and allows developers to simulate real-world scenarios without requiring physical hardware.

2.0 RViz

We use the rviz to show the perspective of robot. (Robot Visualization) is a visualization tool in ROS used to display robot sensor data and state information in real-time. It allows developers to monitor and debug robot behaviors by visualizing elements like Lidar scans, robot models, navigation paths, and environment maps. RViz is essential for understanding how a robot perceives and interacts with its environment.

2.1 ROS

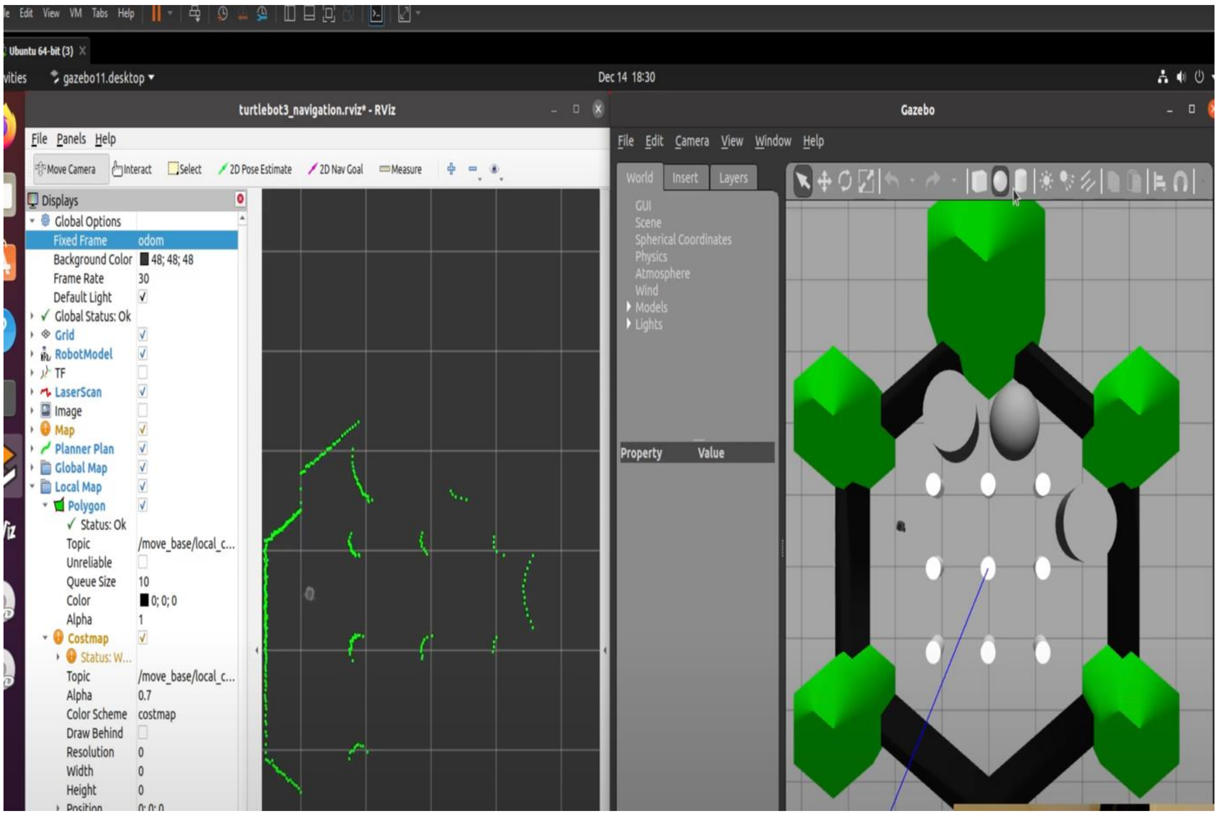
(Robot Operating System) is an open-source framework designed to develop and control robots. It provides us tools, libraries, and conventions for managing hardware abstraction, device drivers, communication between processes, and simulation environments. ROS simplifies complex robot tasks like navigation, perception, and motion planning, making it easier to develop and deploy robotic systems.

3.0 PYTHON

Our python script implements autonomous behavior for a robot, focusing on exploration and obstacle avoidance. The script begins by initializing a ROS node named wander_bot, enabling communication with other ROS nodes. It subscribes to the /scan topic to receive Lidar sensor data, which provides crucial environmental information. Simultaneously, it publishes velocity commands to the /cmd_vel topic, allowing the robot to move and adjust its trajectory.

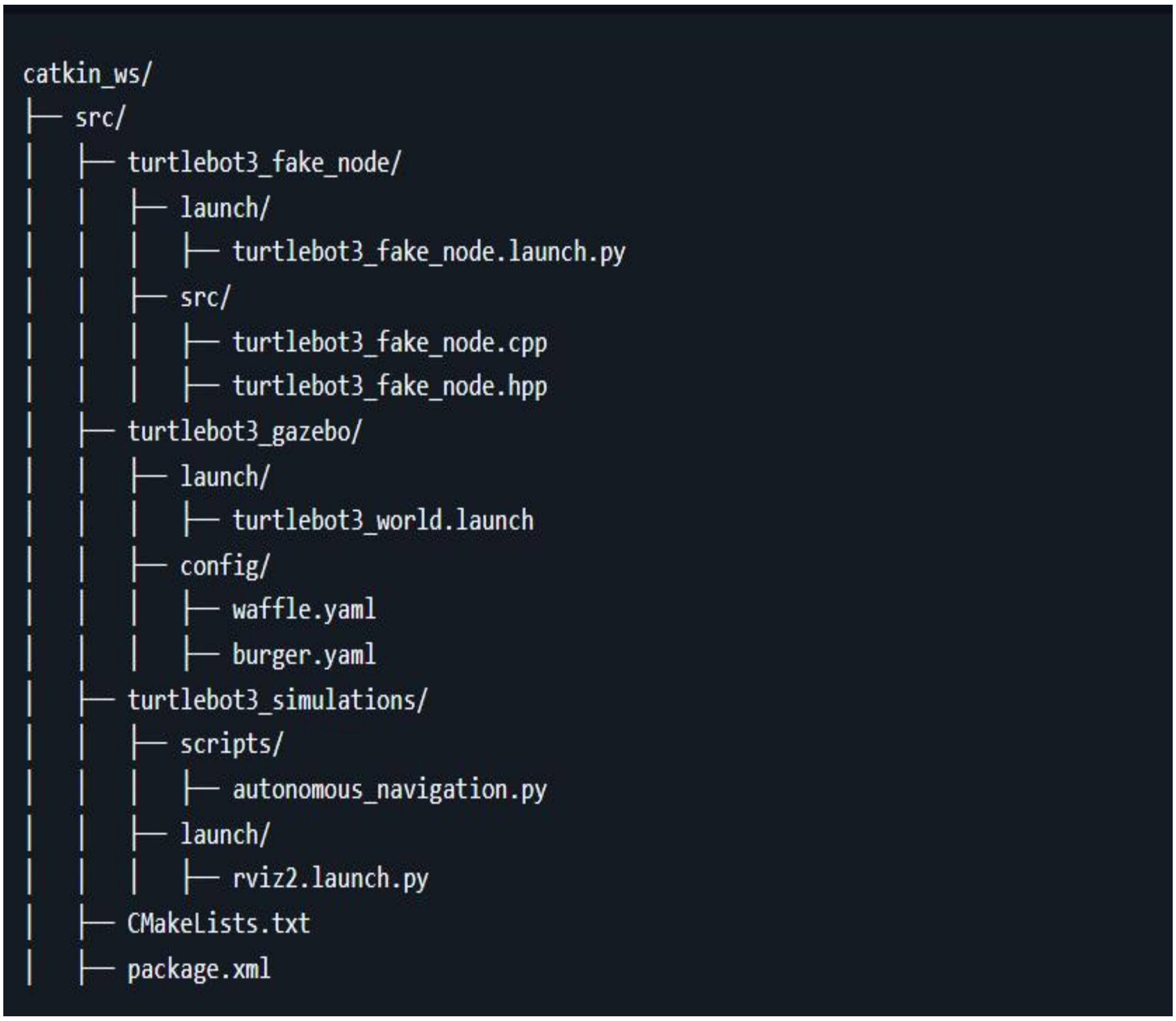
The robot operates in two primary states: "explore" and "turn." In the "explore" state, it moves forward while introducing slight random angular adjustments to simulate curiosity. If an obstacle is detected within 0.5 meters ahead, the robot transitions to the "turn" state, where it decides to turn left or right based on the proximity of surrounding obstacles, ensuring a safe path forward. To maintain diversity in its movements, the robot periodically adjusts its exploration direction, preventing repetitive behavior.

Lidar data processing is a critical aspect of the script. The data is filtered to exclude invalid values such as inf or 0.0, ensuring reliable obstacle detection. Any processing errors are logged for debugging purposes. By combining dynamic exploration with robust obstacle avoidance, this script enables the robot to autonomously navigate its environment, showcasing adaptability and effective decision-making in real-time.



Displays	
Global Options	
Fixed Frame	odom
Background Color	48; 48; 48
Frame Rate	30
Default Light	✓
✓ Global Status: Ok	
Grid	✓
RobotModel	✓
TF	☐
LaserScan	✓
Image	☐
Map	✓
Planner Plan	✓
Global Map	✓
Local Map	✓
Polygon	✓
✓ Status: Ok	
Topic	/move_base/local_c...
Unreliable	☐
Queue Size	10
Color	0; 0; 0
Alpha	1
Costmap	✓
Status: W...	
Topic	/move_base/local_c...
Alpha	0.7
Color Scheme	costmap
Draw Behind	☐
Resolution	0
Width	0
Height	0
Position	0; 0; 0

4.0 PROJECT ARHİTECTURE



```
catkin_ws/
├── src/
│   ├── turtlebot3_fake_node/
│   │   ├── launch/
│   │   │   ├── turtlebot3_fake_node.launch.py
│   │   ├── src/
│   │   │   ├── turtlebot3_fake_node.cpp
│   │   │   ├── turtlebot3_fake_node.hpp
│   ├── turtlebot3_gazebo/
│   │   ├── launch/
│   │   │   ├── turtlebot3_world.launch
│   │   ├── config/
│   │   │   ├── waffle.yaml
│   │   │   ├── burger.yaml
│   ├── turtlebot3_simulations/
│   │   ├── scripts/
│   │   │   ├── autonomous_navigation.py
│   │   ├── launch/
│   │   │   ├── rviz2.launch.py
│   ├── CMakeLists.txt
│   ├── package.xml
```

5.0 Challenge

1-ROS configurations and creating workspace

According the some discordant version between ubuntu and ros - make us hard times. For the solution, we try to setup all the systems and platforms with different combinations until make a coorination.

2-Rviz Visualiations

Because of we didn’t give the map informations to the robot making a visuliaitons from the robot perspective had been hard.for the solution we virsualiationed the lidar data to create view.

3-Filtering The Wrong Vlue That Came From Lidar

To address the problem of filtering erroneous values from Lidar data, such as inf, 0.0, or NaN, the solution involves preprocessing the sensor readings to remove invalid entries before using the data for decision-making. This can be achieved programmatically by applying a filter that excludes values outside a specified range, typically defined by the Lidar’s minimum and maximum sensing distances. For instance, you can write a function to discard readings less than 0.1 meters or greater than 10 meters, as well as values marked as infinite or undefined (NaN). In a ROS-based setup, this filtering is applied directly within the callback function handling Lidar messages, ensuring only valid data is used for tasks like obstacle avoidance or environment mapping. Additionally, advanced filtering can be implemented using ROS packages like laser_filters, which allow configurable thresholds for data cleaning. By incorporating these methods, you can enhance the reliability of your Lidar-based applications.

SOURCES:
Ros Course
https://www.youtube.com/watch?v=VheHmWV7Iu0&list=PLlqdnFs9xNwql5KET7v7zyl393y10qxtw&index=10

Ros
http://wiki.ros.org/Courses

OUR YOUTUBE LINK :
https://www.youtube.com/watch?v=AC5vtsC7Uyw