SRI RAMAKRISHNA ENGINEERING COLLEGE

BE ROBOTICS AND AUTOMATION

ROS2 BASED BUTLER BOT USING CUSTOM ROBOT

-SANTHOSH K

Table of Contents

1. Introduction

This document details the development of a butler robot solution for the French Door Café as specified in the ROS Assessment Task. The café required an automated solution to handle food delivery from the kitchen to customer tables during busy periods, aiming to reduce operational costs while maintaining efficient service.

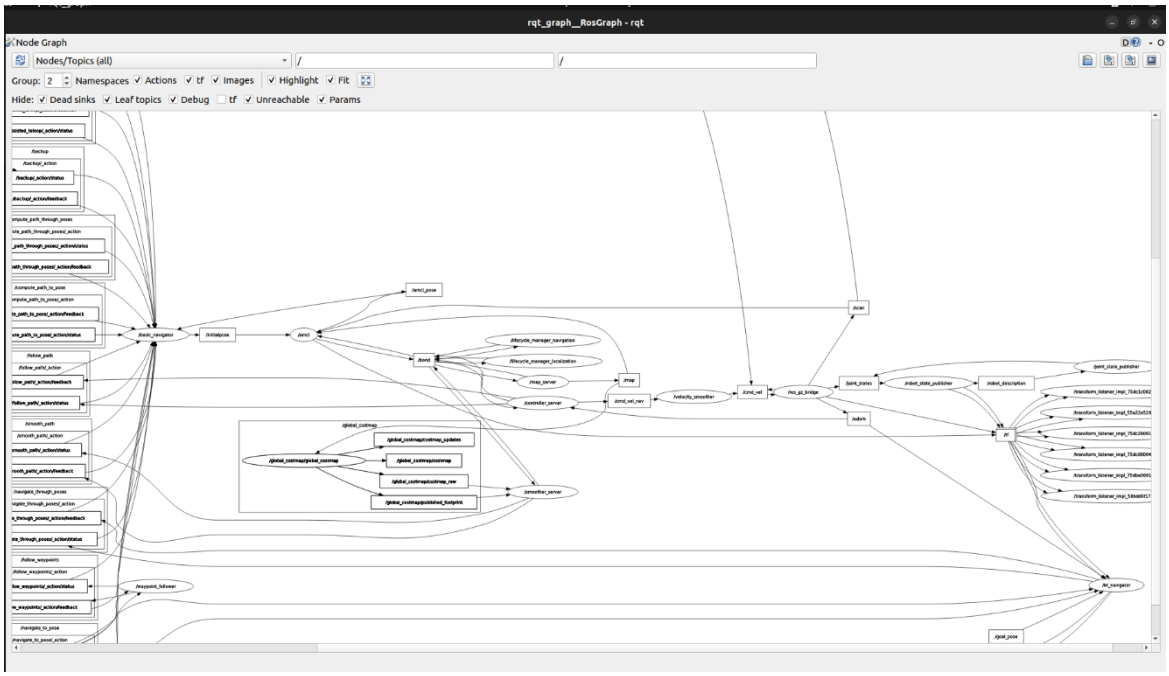
The butler robot is designed to:

- Navigate autonomously between a **home** position, **kitchen**, and three customer **tables**
- Collect orders from the kitchen and deliver them to respective tables
- Handle **confirmations** and timeouts at both kitchen and tables
- Process order **cancellations** mid-delivery
- Manage **multiple orders** in an optimized sequence
- Return to home position after completing delivery tasks

This solution leverages ROS 2 (Robot Operating System), Navigation Stack (Nav2), and Ignition Gazebo for simulation, providing a comprehensive implementation that meets all the specified requirements in a generic, scalable manner.

2. System Architecture

The butler robot system follows a modular architecture with the following key components:



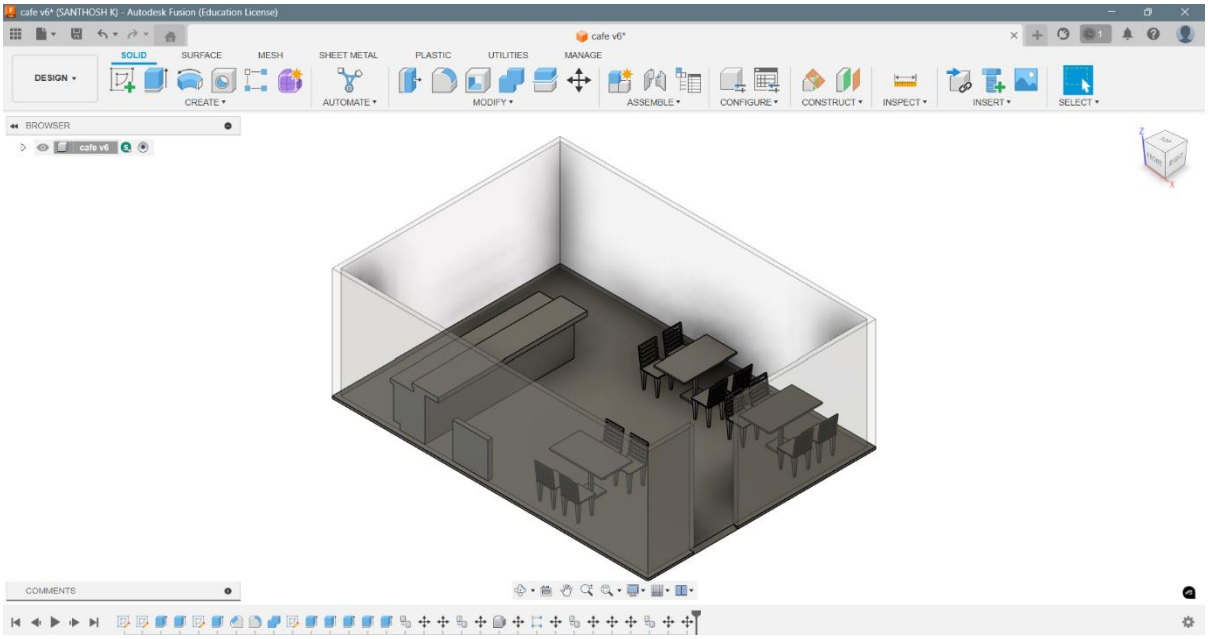
Communication between components is handled through ROS topics and services, ensuring loose coupling and maintainable code structure.

3. Custom Environment Development

3.1. Café World Modelling in Fusion 360

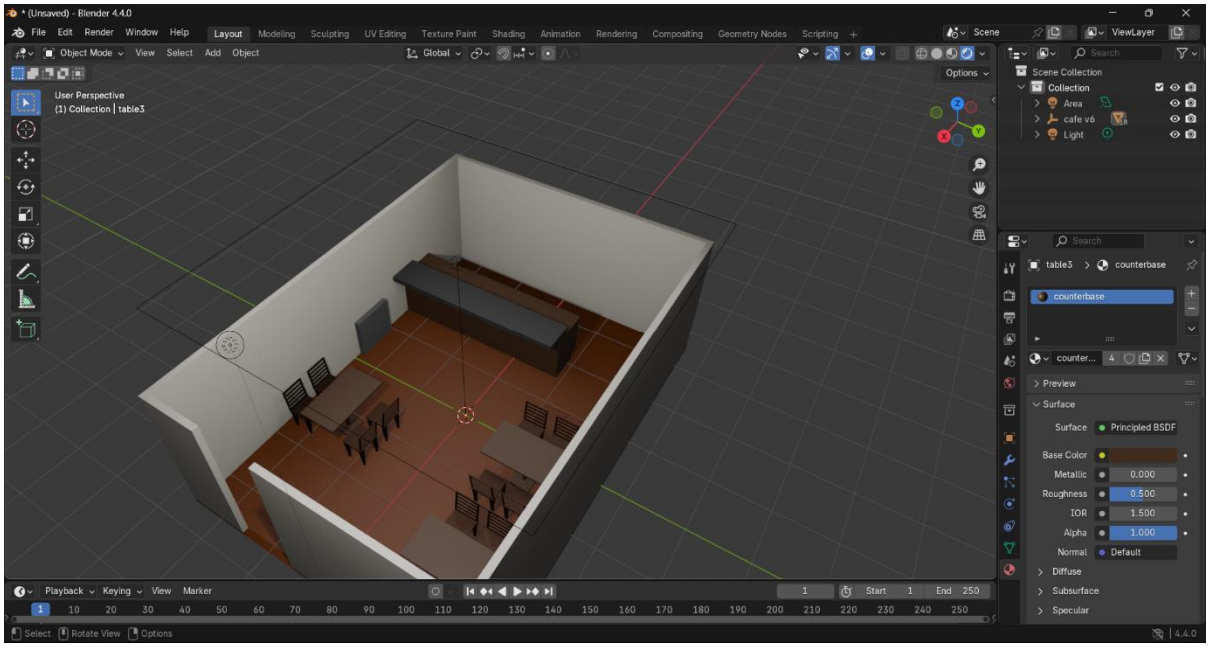
The café environment was modelled to include realistic representations of:

- Restaurant layout with tables, chairs, and obstacles
- Kitchen area with appropriate counters and workspace
- Robot home position



3.2. Texturing in Blender

To enhance visual fidelity, the Fusion 360 model was imported into Blender for detailed texturing:

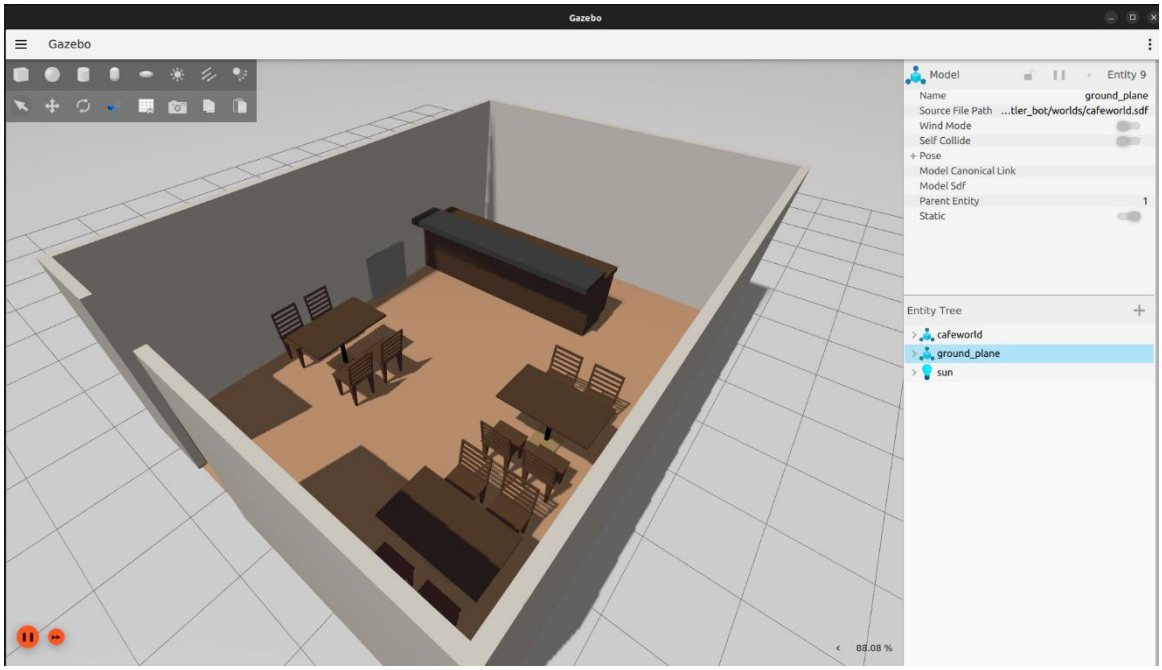


The texturing process included:

- Material assignment to different surfaces (wood, metal, fabric)

3.3. Launching in Ignition Gazebo

The textured model was exported in a format compatible with Ignition Gazebo and integrated with the simulation environment:

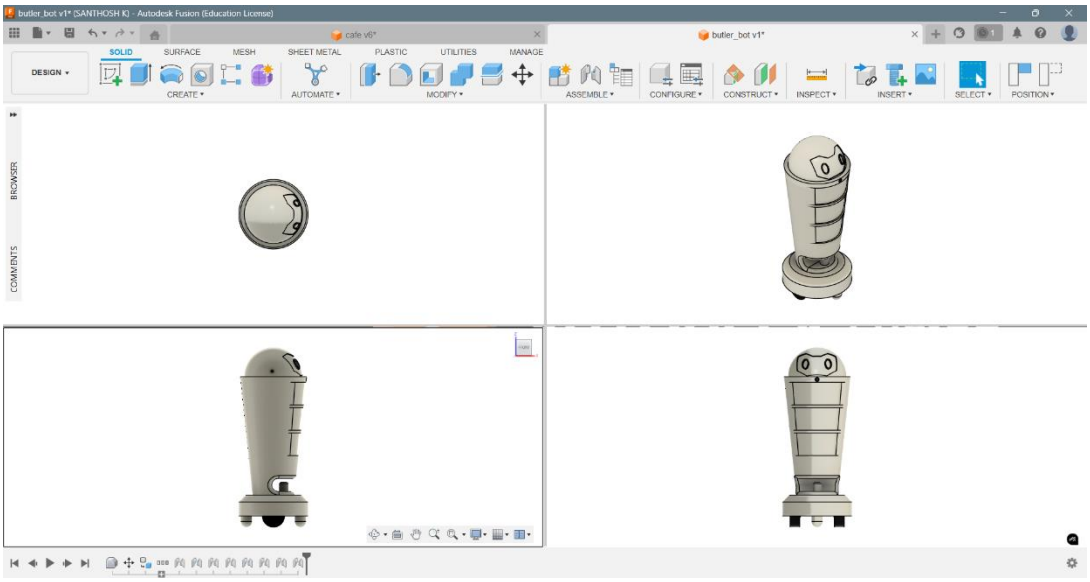


4. Robot Design and Development

4.1. Custom Robot Modelling

The butler robot was designed with the following considerations:

- Stable base for carrying food items
- Appropriate height for interacting with tables
- Sufficient space for sensors and computing hardware
- Aesthetic appeal suitable for a café environment

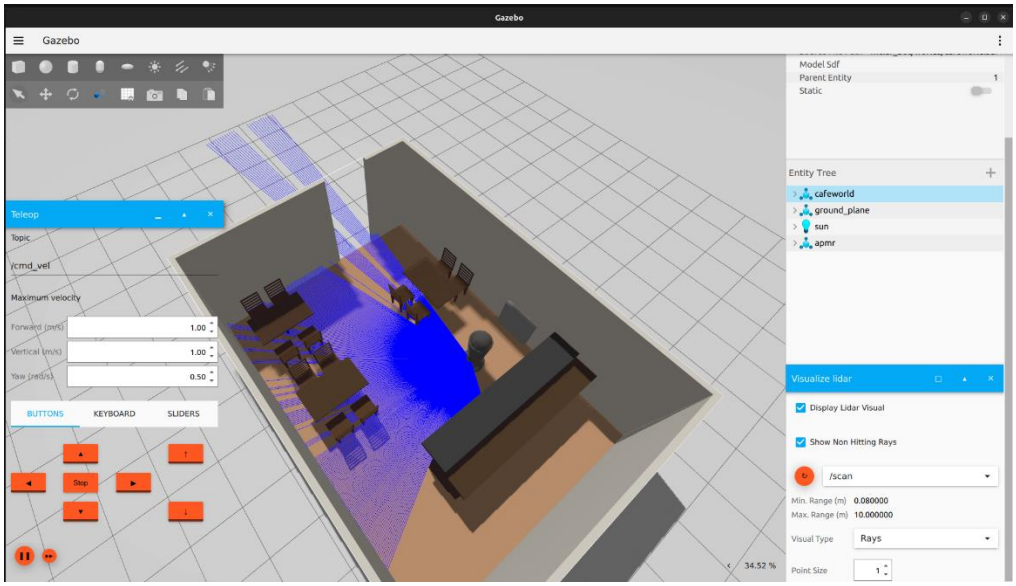


4.2. URDF Generation with URDF_Exporter_Ros2 Plugin

The Fusion 360 model was converted to URDF format using the URDF_Exporter_Ros2 plugin.

4.3. Sensor Integration

LiDAR Configuration:



4.4. Differential Drive Configuration

<plugin

filename="libignition-gazebo-diff-drive-system.so"

name="ignition::gazebo::systems::DiffDrive">

<!--wheel info-->

<left_joint>left_wheel_joint</left_joint>

<right_joint>right_wheel_joint</right_joint>

<wheel_separation>0.32</wheel_separation>

<wheel_radius>0.05</wheel_radius>

<topic>cmd_vel</topic>

<odom_topic>/odom</odom_topic>

<odom_publish_frequency>30</odom_publish_frequency>

<frame_id>odom</frame_id>

<child_frame_id>base_footprint</child_frame_id>

<tf_topic>tf</tf_topic>

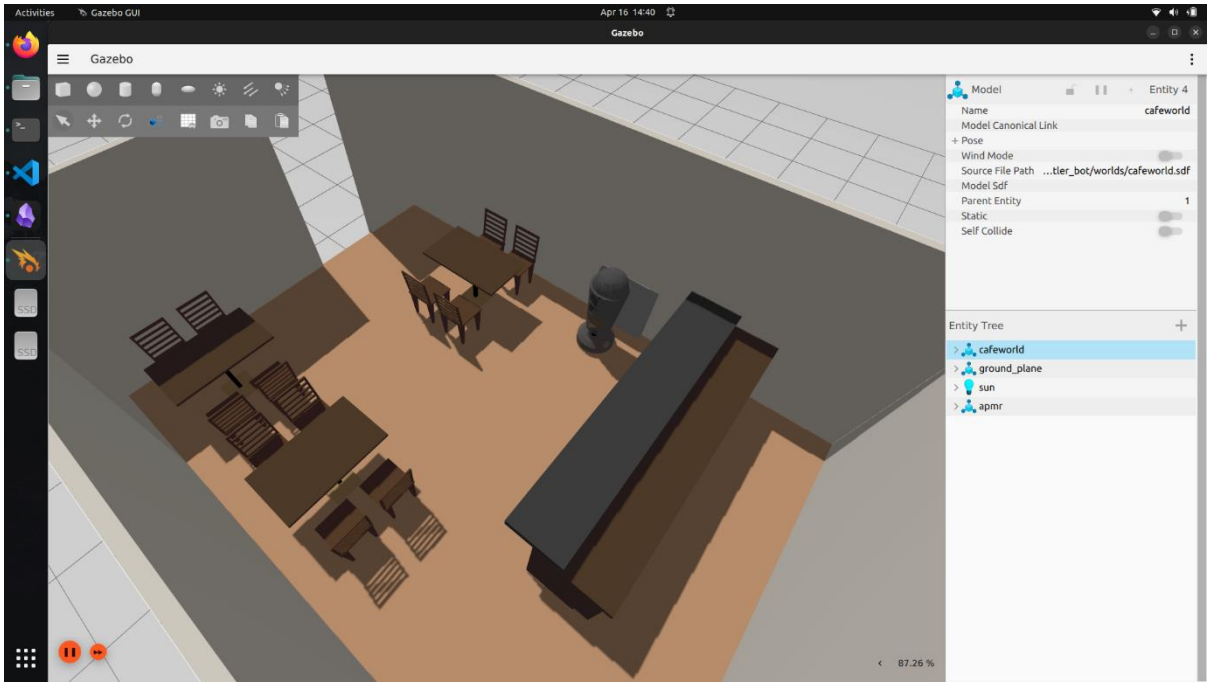
</plugin>

5. System Setup

5.1. Robot Spawning

Run the following in a terminal:

$ ros2 launch butler_bot butler_ign_spawn.launch.py

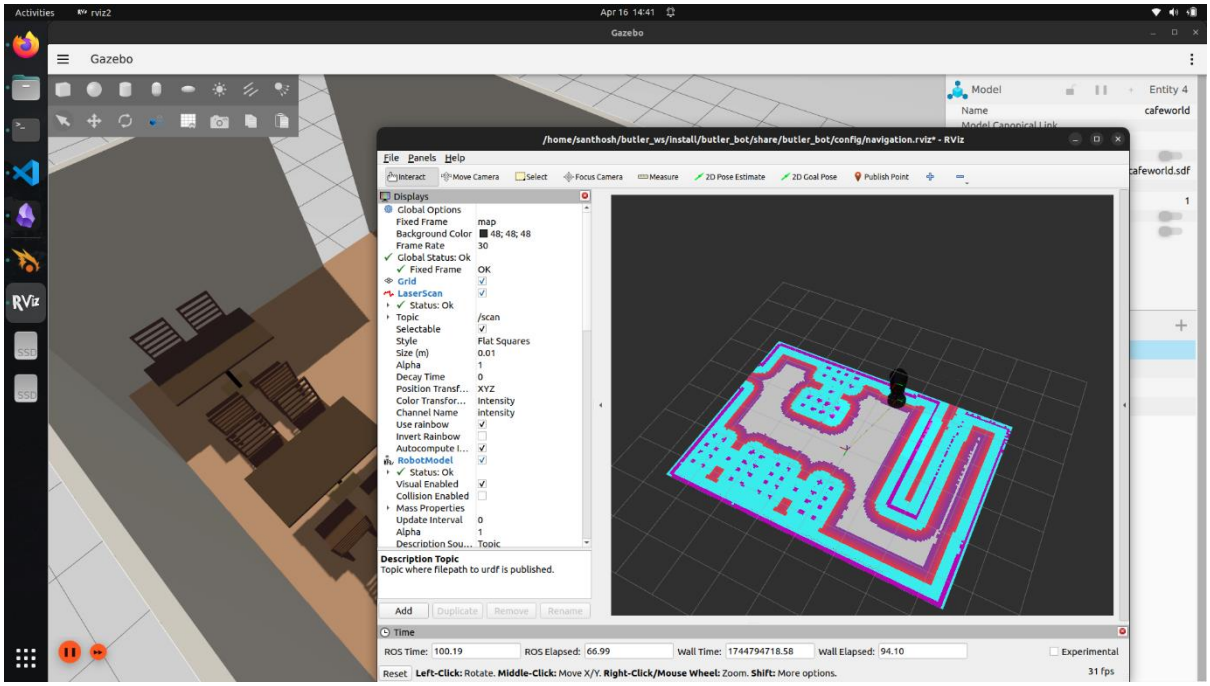


5.2. Nav2 Configuration

Run the following in a new terminal:

$ ros2 launch butler_bot navigation2.launch.py

open params/butler.yaml to change nav2 params



5.3. Launch Files Setup

$ ros2 launch butler_bot display.launch.py :

Launches RViz2 for visualizing the robot model and sensor data.

$ ros2 launch butler_bot ign.launch.py :

Launches the simulated café environment and spawns the butler robot in Ignition Gazebo.

$ ros2 launch butler_bot navigation2.launch.py :

Starts the Navigation2 stack for autonomous path planning and obstacle avoidance.

6. Implementation of Requirements

This section outlines the full implementation process to simulate the butler robot in the custom café environment using ROS 2 and Ignition Gazebo, following the task requirements.

6.1 Workspace Setup

Create a new ROS 2 workspace and clone the project:

```
mkdir -p ~/butler_ws/src
cd ~/butler_ws/src
git clone https://github.com/SANTH0SH-K/butler_bot.git
cd ..
rosdep install --from-paths src --ignore-src -r -y
colcon build
```

source install/setup.bash

6.2 Launching Ignition Gazebo with Café Environment

To launch the custom environment and spawn the robot:
$ ros2 launch butler_bot ign.launch.py

6.3 Launching RViz and Nav2

In a new terminal:

source ~/butler_ws/install/setup.bash

ros2 launch butler_bot navigation2.launch.py

NOTE : use nav2.rviz param file instead of navigation.rviz to enable global and local path

6.4 Issuing Orders via CLI

- **Place Order :** ros2 topic pub /order std_msgs/String "data: 'table1'"
- **Confirm at Kitchen or Table :** ros2 topic pub /confirmation std_msgs/String "data: 'confirmed'"
- **Cancel Order :** ros2 topic pub /cancel_order std_msgs/String "data: 'cancel'"

7. Testing and Results

Task 1 :

- To Run task1, execute below command.
  - ros2 run butler_bot task_1.py
- It subscribes to the /order topic, so you can send commands like:
  - ros2 topic pub /order std_msgs/String "data: 'table1'" --once

Task 2 :

- To Run task2, execute below command.
  - ros2 run butler_bot task_2.py
- To send an order to a table, publish to the order topic:
  - ros2 topic pub /order std_msgs/String "data: 'table1'" --once
- Send confirmations (required at each location):
  - ros2 topic pub /confirmation std_msgs/String "data: 'confirmed'" --once

Task 3 :

- To Run task3, execute below command.
  - ros2 run butler_bot task_3.py
- To send an order to a table, publish to the order topic:

    - ros2 topic pub /order std_msgs/String "data: 'table1'" --once
- Send confirmations (required at each location):
    - ros2 topic pub /confirmation std_msgs/String "data: 'confirmed'" --once

Task 4:

- To Run task4, execute below command.
    - ros2 run butler_bot task_4.py
- To send an order to a table, publish to the order topic:
    - ros2 topic pub /order std_msgs/String "data: 'table1'" --once
- To cancel an active order:
    - ros2 topic pub /cancel_order std_msgs/String "data: 'cancel'" --once

Task 5:

- To Run task5, execute below command.
    - ros2 run butler_bot task_5.py
- To order deliveries to tables, publish to the order topic with table numbers:
    - ros2 topic pub /order std_msgs/String "data: '1 2 3'" --once

Task 6 :

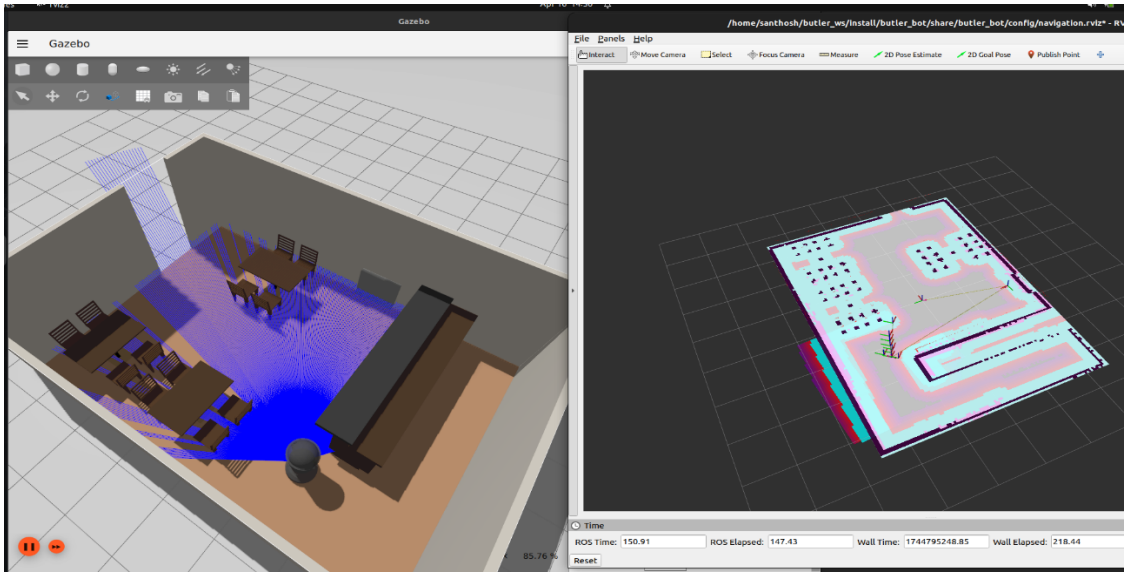
- To Run task6, execute below command.
    - ros2 run butler_bot task_6.py
- Send orders to multiple tables:
    - ros2 topic pub /order std_msgs/String "data: '1 2 3'" --once
- Send confirmations (required at each location):
    - ros2 topic pub /confirmation std_msgs/String "data: 'confirmed'" --once

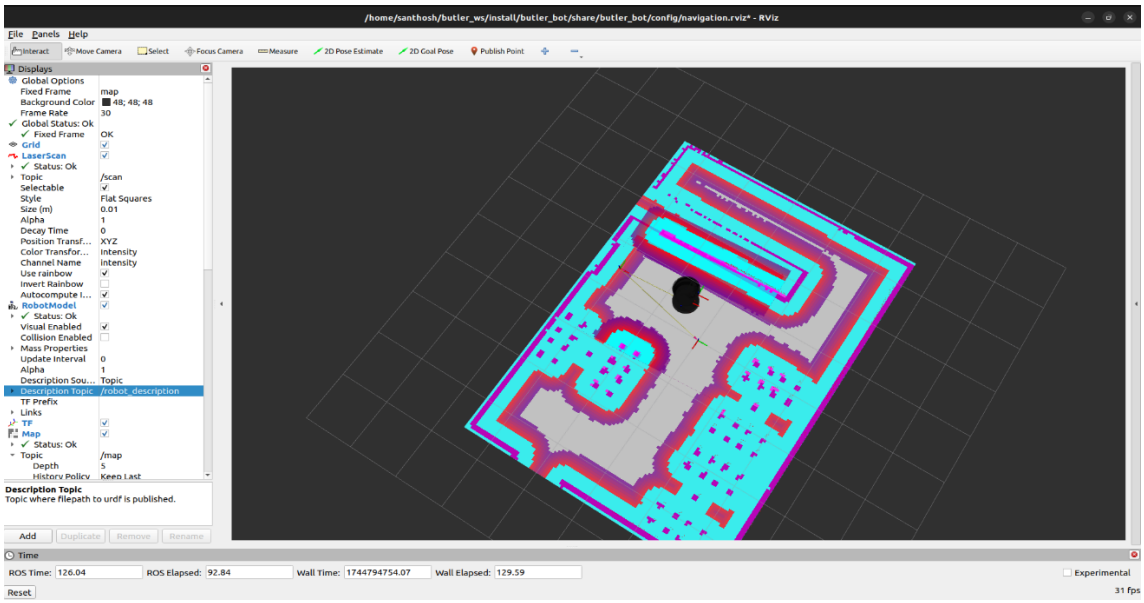
Task 7 :

- To Run task7, execute below command.
    - ros2 run butler_bot task_7.py
- Send orders to multiple tables:
    - ros2 topic pub /order std_msgs/String "data: '1 2 3'" --once
- Send confirmations at kitchen and tables:
    - ros2 topic pub /confirmation std_msgs/String "data: 'confirmed'" --once
- Cancel specific table orders:

- ros2 topic pub /cancel std_msgs/String "data: 'table2'" --once

- Cancel the current navigation:
  - ros2 topic pub /cancel std_msgs/String "data: 'cancel'" --once





Github link : [SANTH0SH-K/Butler_bot_ros2]

Requirement	**Test Case**	**Result**
Single Order	Basic delivery path	✓Passed
Timeout Handling	Kitchen timeout	✓Passed

Requirement	Test Case	Result
Timeout Handling	Table timeout	✓Passed
Confirmation Management	Kitchen confirmation, table timeout	✓Passed
Order Cancellation	Cancel while going to kitchen	✓Passed
Order Cancellation	Cancel while going to table	✓Passed
Multiple Orders	Delivery to all tables	✓Passed
Multiple with Timeout	Skip table1, deliver to others	✓Passed
Multiple with Cancellation	Skip table2, deliver to others	✓Passed

8. Conclusions and Future Work

This project successfully implemented a ROS-based butler robot system capable of autonomously delivering food orders in a café environment. The implementation meets all seven requirements specified in the assessment task, demonstrating robust navigation, confirmation handling, timeout management, and multi-order processing.

Key accomplishments:

- Custom environment and robot modeling
- Effective integration with Nav2 for autonomous navigation
- Robust state machine for managing different delivery scenarios
- Efficient handling of multiple orders with cancellations and timeouts

9. References

1. ROS 2 Documentation: ROS 2 Documentation — ROS 2 Documentation: Humble documentation
2. Navigation 2 Documentation:Nav2 — Nav2 1.0.0 documentation
3. Ignition Gazebo: Ignition Tutorials — Gazebo fortress documentation