BB-bot in cafeteria environment to handling the orders

Date: *15.04.2025*
Author: Hariveerabadran V S

OBJECTIVE:

To develop an autonomous butler robot for the French Door Café that efficiently delivers food from the kitchen to customer tables, reducing employee workload and improving service speed, especially during busy hours. It handles tables with certain scenario.

APPROACH:

The ROS 2 **service-client** communication model is used to allow users (acting as clients) to send requests to the robot server, such as placing an order, cancelling it, or confirming delivery. For handling **multiple users**, a **first-come, first-served queue system** is implemented to manage requests efficiently.

The **Nav2 stack** is used for **autonomous navigation** based on predefined coordinates, enabling the robot to move between home, kitchen, and customer tables. A simple interface acts as the **user interface**, allowing users to interact with the system for order handling within the café.

Behavior Logic

The robot handles various scenarios:

- **Normal flow**: Home → Kitchen → Table → Kitchen → Home
- **Order Cancelled (before pickup)**: Home → Kitchen (skip table) → Home
- **Order Cancelled (on the way)**: Home → Kitchen → Table (cancelled) → Kitchen → Home
- **No response at table**: Robot skips to next delivery
- **Multiple orders**: Robot chains delivery to multiple tables

ROS2 pkg:

In bb-amr_ws workspace contain gazebo simulation package, service, and Pyqt5 gui package.

cafe_interfaces: Ament_cmake_pkg for the .srv to host service-client communication.

.srv file contain:
```
string tables // request
bool order
bool cancel
string mode
---
string msg  //respond
```

- **mode used for accept the order from user and kitche.**

gazebo_sim: Contains gazebo simulation and rviz launch file with URDF, meshes, world. Param file for nav2, slamtoolbox , twist mux.

- **To Launch gazebo :** ros2 launch gazebo_sim gazebo.launch.py
- **To Lauch rviz for nav2:** ros2 launch gazebo_sim nav2.launch.py
- **Teleop cmd:** ros2 run teleop_twist_keyboard teleop_twist_keyboard --ros-args --remap cmd_vel:=/cmd_vel_key

NOTE: It contains other launch file to mapping and displaying rviz.
The topic /cmd_vel for bb_bot remap to / bb_botamr/cmd_ve
Lidar /scan remap to bb_botamr/scan
Twist mux used for priority control on bot while nav2 stack running
Under config/ folder nav2, ekf, slam param can be modified.

bb_ui: Contain ui-client file and service to accept the request and perform motion via nav2 **simple commander API**.

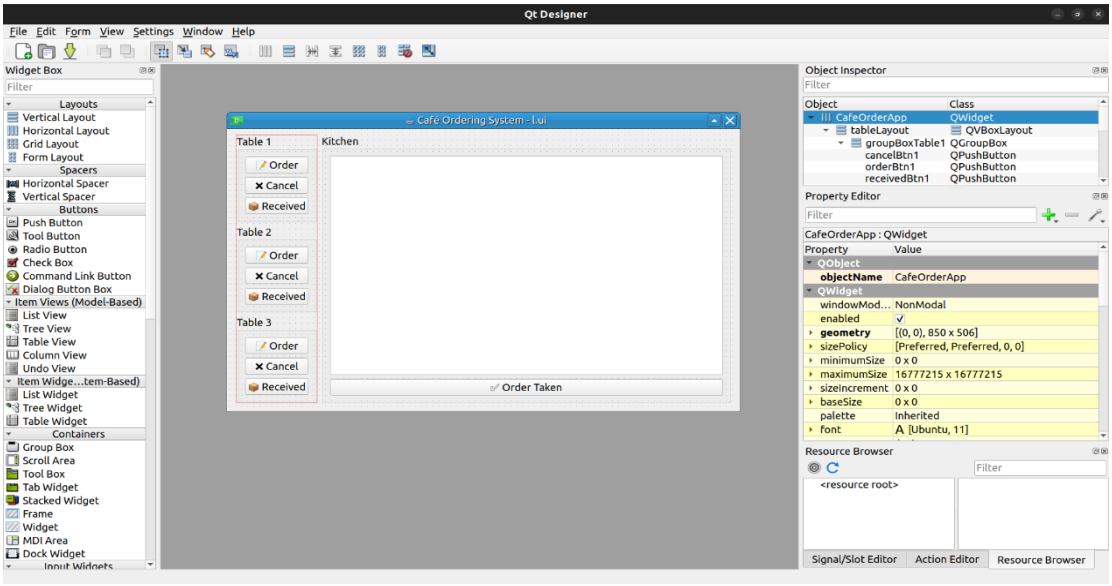
- **To start service:** ros2 run bb_ui cafe_service
- **To lauch client node:** ros2 run bb_ui ui

Note: for easy start ui make alias in .bashrc : alias bb_ui='python3 /src/bb_ui/bb_ui/ui_main.py' **run this inside workspace folder.**

UI and **ROS2 client** are running in parallel by thread and **service** and **nav2 stack** running in parallel.

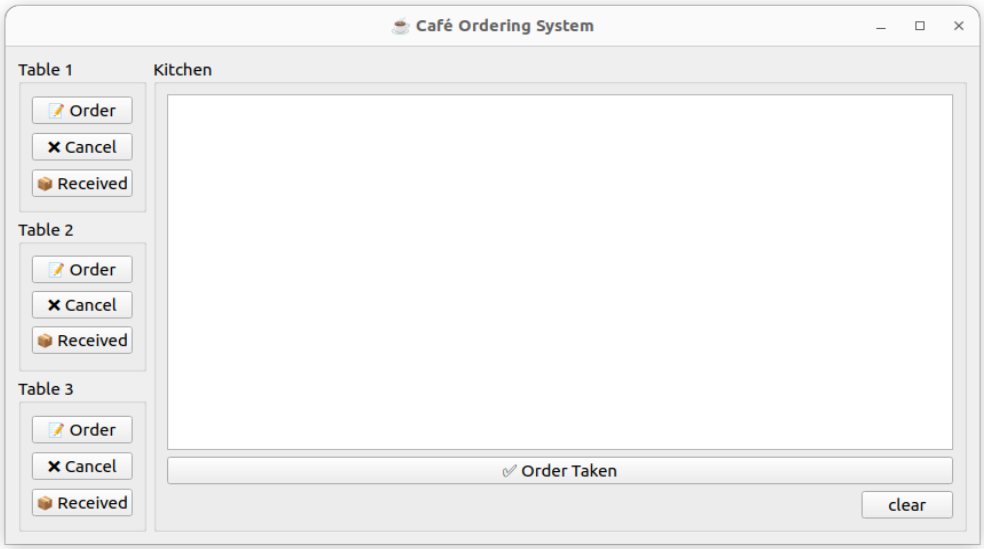
UI_DESIGN:

The user ui developed on pyqt5 desinger in python environment. Then ros2 client is added by thread to send request for service.



COMMAND:

The robot main command is given UI.

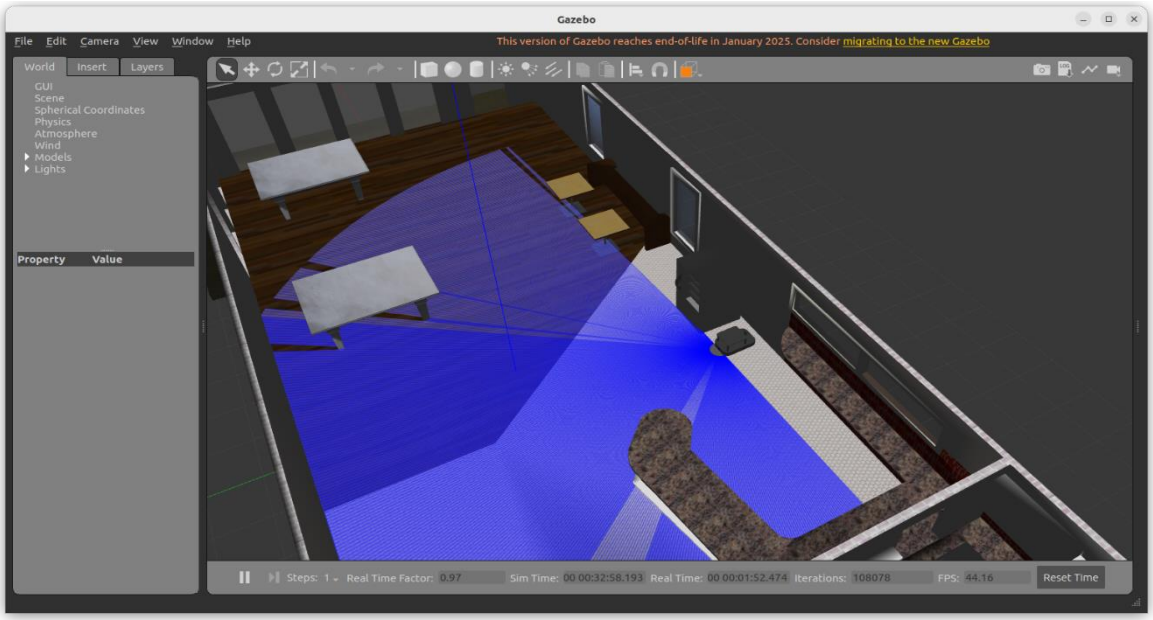


- **Order** can be placed by pressing the **Order** button.
- **Cancel** is performed using the **Cancel** button.
- **Confirmation** is done by pressing the **Received** button.
- The kitchen's order is confirmed using the **Order Taken** button.
- **Clear** is used to clear the kitchen log display.

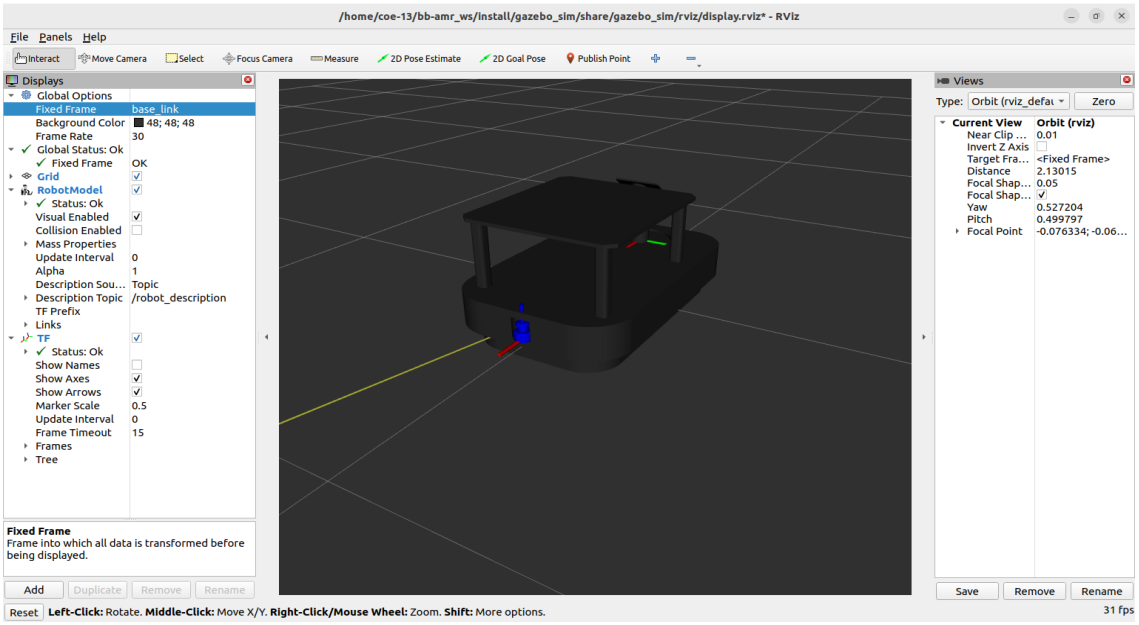
For every action in here client will make request in server and nav2 move accordingly.

SIMULATION OUTPUT:

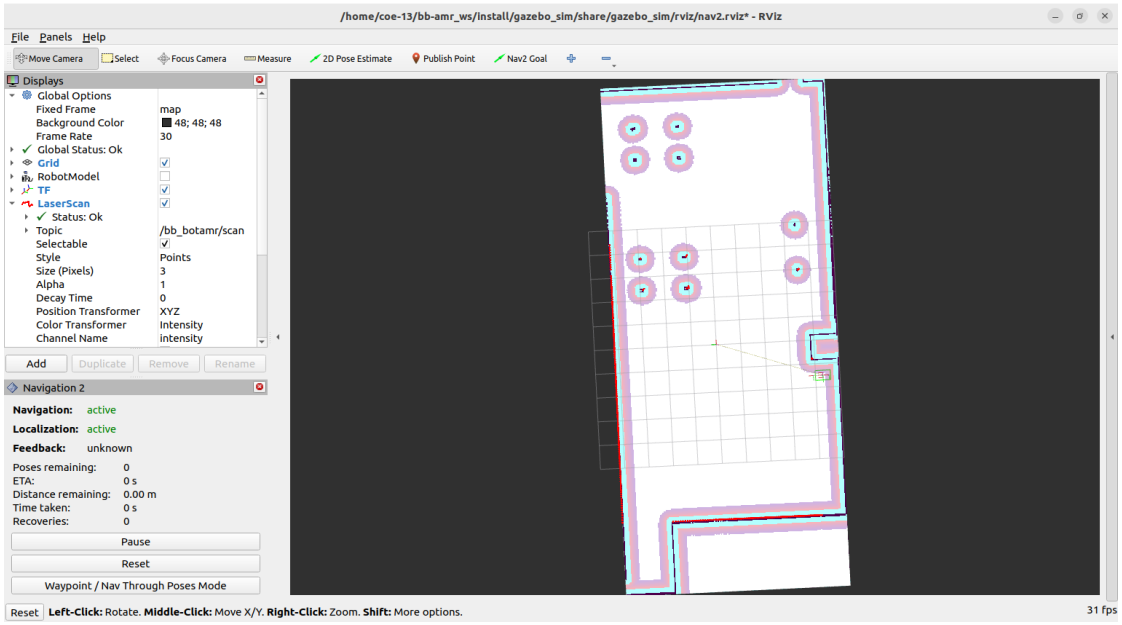
Gazebo sim:



Display Rviz:



Nav2 Costmap:



CONCLUSION:

This workspace demonstrates a ROS 2 project aimed at simulating task flows for mobile robots in service environments. The use of service-client, custom interfaces, Gazebo simulations, and a clear behavioral model supports real-world application development for AMR systems.