Custom Bot Using ROS2 nav for food serving

PROJECT OVERVIEW

The French Door Café aims to streamline operations by introducing a robotic butler to handle food deliveries during peak hours. This project presents a custom-built mobile robot that autonomously navigates between the kitchen and customer tables using ROS 2 and the Nav2 stack.The robot starts from a home position, picks up orders from the kitchen, delivers them to Table 1, 2, or 3, and returns to its base

SYSTEM ARCHITECTURE

## **1. Core Package Files**

These files define the package and its dependencies.

* package.xml: Describes the package metadata, dependencies (e.g., rclpy, nav2, gazebo\_ros), license, author info.
* setup.py / setup.cfg: Python packaging files used to make this a valid installable ROS 2 Python package.

## **2. URDF and Robot Description (urdf/)**

This folder defines the robot's physical and visual structure using XACRO and URDF.

* turtlebot3\_burger.urdf: URDF for the base TurtleBot3 (used as a reference or base).
* b\_bot.xacro: The main robot description written in XACRO, a macro-enabled XML language to simplify URDF files.
* materials.xacro: Contains color/material definitions used in the robot’s model.
* b\_bot.trans: Likely includes static transform definitions.
* b\_bot.gazebo: Gazebo-specific extensions like plugins for sensors or joints.
* gazebo\_control.xacro: Defines ROS control interfaces and Gazebo plugins for controlling joints/wheels.

## **3. Launch Files (launch/)**

These launch files orchestrate different simulation and real-world tasks.

* gazebo.launch.py: Launches the Gazebo simulator with the robot.
* slam.launch.py: Starts SLAM (e.g., slam\_toolbox) to build a map of the environment.
* navigation2.launch.py: Starts the full Nav2 stack (planner, controller, behavior tree, etc.).
* display.launch.py: Likely opens RViz with the robot model and config.

## **4. Configuration Files (config/)**

Config files for RViz and Nav2 behavior.

* nav2\_params.yaml / nav2\_params2.yaml: Parameters for the Navigation2 stack (planner settings, controller plugins, recovery behavior).
* mapper\_params\_online\_async.yaml: SLAM parameters for async online mapping.
* display.rviz: RViz configuration to load a specific visualization layout.

## **5. Behavior Scripts (b\_bot\_description/)**

These Python scripts implement robot behavior logic.

* send\_goal.py: Sends navigation goals to the NavigateToPose action server.
* order\_timeout.py: Likely handles timeouts or failed delivery goals.
* follow\_path.py: Possibly used to execute custom path-following logic.
* \_\_init\_\_.py: Makes the directory a valid Python module.

## **6. Tests (test/)**

Standard ROS 2 testing framework support.

* test\_copyright.py: Checks if source files have copyright.
* test\_flake8.py: Enforces PEP8 style via Flake8.
* test\_pep257.py: Ensures docstring style compliance.

## **7. Meshes and Models**

Used for realistic visualization in RViz and Gazebo.

* meshes/: STL 3D models for visualization in Gazebo/RViz.
  + Includes parts like lidar\_1.stl, base\_link.stl, left\_wheel\_1.stl, etc.
* model/turtlebot3\_burger/: SDF and config files for the TurtleBot3 simulation model.
  + model.sdf, model-1\_4.sdf, model.config define the model’s appearance and behavior in Gazebo.

## **8. Worlds (worlds/)**

Gazebo world definitions.

* cafe\_world.sdf / coffee\_shop\_world.sdf: Simulated environments for the robot to operate in.

## **9. Maps (maps/)**

Pre-built map used for localization and navigation.

* map.pgm: Image file of the map.
* map.yaml: Metadata describing the origin, resolution, and thresholds of the map image

Launch Instructions:

* Build & source the workspace
* To launch gazebo world:

ros2 launch b\_bot\_description gazebo.launch.py

* To launch nav2:

ros2 launch b\_bot\_description navigation2.launch.py

* To run delivery script:

ros2 run b\_bot\_description order

* To give table num as command, open new terminal and paste

ros2 topic pub -1 /order std\_msgs/msg/String "data: 'table1,table2,table3'"

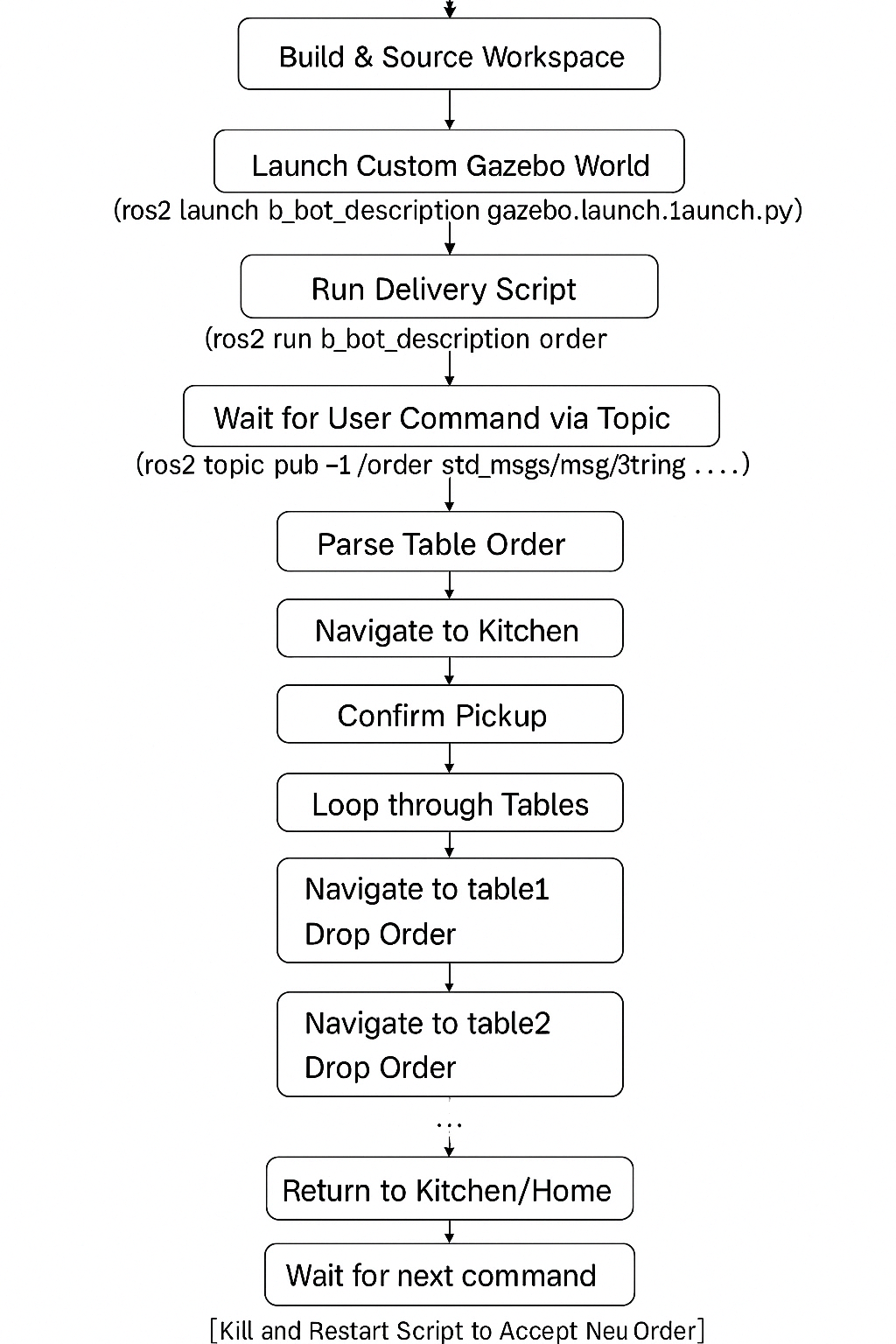
* To give another order after the robot returns to home

kill the delivery script and re-run it.

## Robot Commands

|  |  |
| --- | --- |
| **Command** | **Description** |
| table1 | Deliver to Table 1 |
| Table1,table2,table3 | Multiple table delivery |
| yes | Confirm delivery |
| cancel | Cancel and go home |
| c1, c2, c3 | Cancel a specific table |

**Flow Diagram:**

****

**Output:**

