ROS2 Robotics Software Engineer - Technical Assignment

Overview

This take-home assignment is designed to evaluate your practical ROS2 Humble skills, debugging ability, and architectural thinking in a mobile manipulation scenario.

Problem Statement

You are tasked with developing a coordination system for a mobile manipulator robot that must navigate to a shelf, pick an object, and place it at a delivery location. The system should integrate navigation and manipulation into a unified behavior framework.

Requirements

1. Mobile Manipulator Integration

In the starter code that is given, the `turtlebot3_sim_bringup` package has the launch file to start a simulation with TurtleBot3, and the `my_doosan_pkg` contains the launch file for introducing the Doosan arm into an independent simulation.

Your first task is to mount this arm (or any other smaller independent arm) onto the TurtleBot3 and load it in any industrial environment (example: using the `aws-robomaker-hospital-world` package or some warehouse).

Note: You can use the turtlebot3_manipulation URDF as well, but note that it would lead to a penalty while grading your application.

2. Mission Control System

Implement a mission control system (in **C++**) using a **State Machine** or **Behavior Tree** to coordinate the following stages:

- Navigate to pickup location (using Nav2)
- Perform pick action (mock or MoveIt2-based)
- Navigate to place location
- Perform place action and return to idle

3. Real-World Edge Case Handling

Handle the following real-world-inspired edge cases:

- **Temporary navigation blockages** - Retry navigation
- **Manipulation failure** - Re-attempt or abort mission
- **Sensor feedback timeout** - Graceful recovery
- **Emergency stop event** - Safe halt and reset

4. Sensor Integration

Integrate at least two sensors (e.g., LiDAR + camera or LiDAR + external IMU).

Deliverables

- ROS2 package(s) with working nodes and launch files
- Readable, modular C++ / Python code with comments
- README explaining system architecture and how to run the solution
- Short video or GIF showing the robot completing the task

Bonus Tasks (Optional)

- Integrate additional sensor feedback (camera-based verification)
- Implement concurrent task execution (queue multiple pick/place goals)
- Add energy- or safety-aware behaviors (battery, obstacle zones)

- Extend system to operate under MoveIt2 for arm planning and perception

Evaluation Criteria

- Code clarity, modularity, and documentation
- Correct and robust state machine or behavior tree implementation
- Proper use of ROS2 tools, topics, services, and parameters
- Handling of errors and recovery states
- Overall system integration quality and testability

Technical Specifications

Estimated Time: 6 hours
ROS2 Distribution: Humble
Languages: C++ and/or Python
Environment: Gazebo

Getting Started

1. Clone the repository and navigate to the workspace
2. Install dependencies: `rosdep install --from-paths src --ignore-src -r -y`
3. Build the workspace: `colcon build`
4. Source the workspace: `source install/setup.bash`
5. Test individual packages before integration

Quick Launch Commands

TurtleBot3 alone:

```
export TURTLEBOT3_MODEL=burger
ros2 launch turtlebot3_sim_bringup bringup.launch.py
```

Doosan arm alone:

```
ros2 launch my_doosan_pkg my_doosan_gazebo_controller.launch.py
```

Hospital environment:

```
ros2 launch aws_robomaker_hospital_world hospital.launch.py
```

Good luck with the assignment!