*Project: Dynamic Warehouse Navigation with A & MPPI**

1. Objective

The candidate must set up a differential drive robot in a **Gazebo** simulation. The robot must navigate a warehouse environment using an *A global planner** and an **MPPI local controller**, demonstrating smooth path tracking and the ability to avoid dynamic obstacles not present in the static map.

2. Technical Specifications

- **ROS 2 Version:** Humble or later (Jazzy/Kilted).
- **Global Planner:** nav2_smac_planner::SmacPlanner2D (A* implementation).
- **Local Controller:** nav2_mppi_controller::MPPIController.
- **Kinematics:** Differential Drive (DiffDrive).
- **Simulation:** Gazebo (Classic or Ignition/Harmonic) with a warehouse world.

---

3. Task Breakdown

Phase 1: Simulation Setup

- Provide a URDF/Xacro for a differential drive robot with:
    - 2D LiDAR (for SLAM/Localization).
    - IMU (for EKF fusion).
- Load a warehouse world (e.g., aws_robomaker_warehouse_world).
- Verify the **TF Tree** and ensure the robot can be localized using AMCL or Slam Toolbox.

Phase 2: Global Planning (A)*

- Configure the **Planner Server** to use the **Smac Planner 2D** (A*).
- **Requirement:** The candidate must tune the costmap to ensure the robot plans paths through the center of aisles, avoiding "corner cutting" near shelves.

Phase 3: Local Control (MPPI)

- Configure the **Controller Server** with the **MPPI Controller**.
- **Kinematic Constraint:** Set the motion_model to DiffDrive.
- **Tuning:** Configure at least three critics:
    1. **ConstraintCritic:** To keep the robot within physical velocity limits.
    2. **PathAlignCritic:** To ensure the robot stays close to the A* global path.
    3. **ObstaclesCritic:** To handle dynamic obstacle avoidance.

Phase 4: Stress Test

- While the robot is following a path, the candidate must programmatically or manually place a "palette" obstacle in the robot's immediate path.
- The robot must use MPPI's predictive nature to deviate from the global A* path and navigate around the object without stopping.

4. Sample Configuration Reference

The candidate is expected to produce a nav2_params.yaml similar to the following:

YAML

```
None
planner_server:
  ros__parameters:
    planner_plugins: ["GridBased"]
    GridBased:
      plugin: "nav2_smac_planner/SmacPlanner2D" # A*
implementation
      tolerance: 0.1
      use_final_approach_orientation: true

controller_server:
  ros__parameters:
    controller_plugins: ["FollowPath"]
    FollowPath:
      plugin: "nav2_mppi_controller::MPPIController"
      motion_model: "DiffDrive"
      time_steps: 56
      batch_size: 1000
      iteration_count: 1
      vx_max: 0.5
      wz_max: 1.0
      critics: ["ConstraintCritic", "ObstaclesCritic",
"PathAlignCritic", "GoalCritic"]
      ConstraintCritic:
        cost_weight: 4.0
      ObstaclesCritic:
        cost_weight: 10.0
        consider_footprint: true
      PathAlignCritic:
        cost_weight: 15.0
        trajectory_point_step: 4
```

5. Evaluation Criteria

- **MPPI Performance:** Does the robot jitter during obstacle avoidance? MPPI requires high-frequency execution (>30 Hz); look for optimized batch_size and time_steps.
- **Path Quality:** Does the A* planner produce jerky "staircase" paths, or is it properly smoothed via the Smac Planner's built-in smoother?
- **Recovery Behaviors:** How does the robot handle a "blocked path" scenario? (Use of Nav2 Behavior Trees).

---

6. Deliverables

1. **Source Code:** A ROS 2 workspace with all necessary packages.
2. **Launch Scripts:** A single bringup.launch.py that starts Gazebo, Localization, and Nav2.
3. **Technical Brief:** A PDF explaining how the ObstaclesCritic and PathAlignCritic weights were balanced to achieve smooth avoidance.