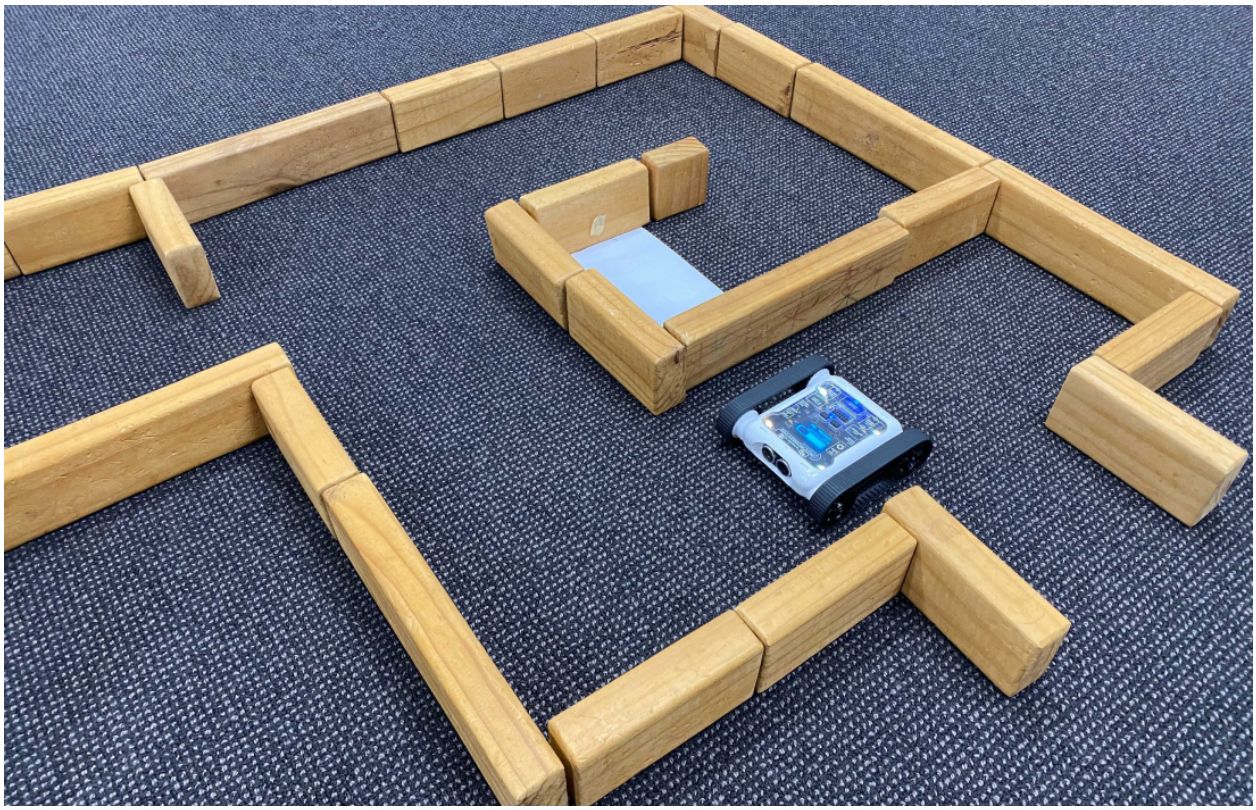
Final Project — Maze SLAM & Relocalization Challenge



Overview

You will be given a **physical maze**. Your job is to:

1. **Build a simulated twin** of that maze in Gazebo using `.sdf` (simulate what you will test physically).
2. Make a mobile robot that can:
    - **Task A (Exploration + Mapping):** When released at a random pose, autonomously navigate and **build a map** of the environment.
    - **Task B (Relocalization + Goal Reach):** After the map exists, place the robot at a new, random pose; it must **relocalize** and **reach a specified target**.

You may use **any hardware/software** you like (ROS or NON ROS), but based on the LEGO NXT ROBOT. Teams with the **best time** (and valid maps) win.

Learning Goals

- Apply **mobile-robot kinematics & control** to navigate safely (e.g., differential-drive constraints, nonholonomy).
- Implement **recursive state estimation** (prediction + update) for localization and mapping.
- Compare/choose between **Gaussian filters** (EKF/UKF) and **non-parametric filters** (Particle) for SLAM and global relocalization.

Project Tasks

1) Simulation Twin (SDF)

- Model the maze as an **SDF world** with proper wall geometry and friction; include your robot model and sensors (e.g., 2D lidar, IMU, other sensors...).
- Automate (With LLMs) a **reproducible launch** (ROS 2 or equivalent) that boots the world and your stack.

2) Task A — Exploration & Mapping

- From a **random initial pose**, the robot should **autonomously explore** and **produce a consistent map** (grid/occupancy or feature-based).

- Use a **recursive estimator** with motion and measurement models (odometry/IMU + range/vision). Show prediction → correction clearly in your code/report.
- Deliver a **final map ** (e.g., picture or table format).

3) Task B — Relocalization & Target Reach

- With the map available, start the robot at a **new random pose**; perform **global relocalization** and then **navigate** to a **given goal** (x, y, θ or zone).
- Robust global relocalization is encouraged to use **particle filters** (Monte Carlo Localization) (Through ROS2 or Custom)

Constraints & Rules

- **Randomization:** Starting pose will be randomized in both A and B; your procedure must not depend on hard-coded starts.
- **Sensing:** Any sensor suite is allowed; document models/noise. The estimator must fuse **control (u_t)** and **measurements (z_t)**.
- **Safety:** No collisions in sim, penalize sustained wall scraping; in lab, use a safe speed limit.
- **Reproducibility:** One-command launch for sim and for system.

What to Deliver

Code & Simulation

- Public or zipped repo:
  - `worlds/maze.sdf` (`.sdf`) + any models
  - `launch/` scripts (ROS 2 or python or equivalent)
  - `README.md` with setup, **exact run commands**, and expected outputs

Demonstrations

- **Sim demo:** Short screen-capture: Task A (mapping) + Task B (relocalize & reach goal).
- **Physical demo:** Same two tasks on the real maze. You may pre-map with your sim twin, but the **final judged map** must come from the physical run in Task A.

Report (Explaining your strategy)

1. **Problem & system overview** (robot, sensors, actuation, kinematics).
2. **Models:** motion model, sensor model, and **Bayes filter** formulation (state, control, measurement, belief).
3. **Estimator choice:** KF/EKF/UKF vs Particle; justify with **linearity**, **uncertainty**, **multimodality**.
4. **Mapping:** occupancy-grid (log-odds) or feature map; include update equations.
5. **Navigation & control:** planner + controller; note non-holonomic constraints if diff-drive.
6. **Experiments:** timing tables, success rate, map quality metrics (see below), failure cases, parameter sweeps.
7. **Ablations:** e.g., effect of particle count vs. convergence time; EKF vs UKF linearization quality.

Scoring & Competition

Timing (50%)

- **Task A time (20%)**: release → **map complete** (frontier exhausted or coverage ≥ 95%). Faster is better.
- **Task B time (20%)**: placement → **goal reached** within tolerance (e.g., ≤10 cm, ≤10°).
- **Start-to-finish integration (10%)**: minimal operator intervention; clean launch; robust recovery.

Quality (30%)

- **Map quality (15%)**: occupancy grid IoU vs reference (or repeatability across two runs ± noise).

- **Localization accuracy (10%)**: final pose error at goal (or RMSE over a fiducial-tracked segment).
- **Navigation smoothness (5%)**: path length vs shortest path; collision penalties.

Engineering & Report (20%)

- **Clarity of models & estimators** (Bayes filter structure; linear vs nonlinear assumptions).
- **Code quality & reproducibility** (README, configs, parameterization).
- **Insightful analysis** (what worked/failed, evidence-based choices).

Penalties: hard-coded starts, non-reproducible launches, unsafe behavior, large drift without recovery.
Tiebreakers: smaller compute budget (CPU-only vs GPU), cleaner code, clearer write-up.

Suggested Technical Path (Optional)

- **Kinematics & control:** Respect non-holonomic constraints; implement a velocity controller and pose-tracking law for your base.
- **Estimator selection:**
  - **EKF/UKF** if your models are near-linear locally and you expect **unimodal** beliefs.
  - **Particle Filter** (MCL/AMCL-like) if you expect **global uncertainty** or **multiple pose hypotheses** during relocalization; mind resampling and variance.
- **Mapping: Occupancy grid** with binary Bayes/Log-odds updates; fuse lidar/range beams with inverse sensor models.
- **Exploration:** Frontier-based or wall-following with loop closures if available.
- **Validation:** Compare sim vs real logs; run with increased sensor noise to test robustness.
- **Performance:** Tune particle count vs latency; for Gaussian filters, check linearization regime and innovation consistency.

Logistics

- **Teams:** 2–3 students.
- **What to submit (single ZIP + PDF):**
  - Code repo (as above), **built map files**, short **sim video** (≤2 min), **report PDF** (≤6 pages).
- **Demo day:** Live **physical** runs for Task A and B on the lab maze under staff timing.
- **Academic integrity:** Cite all external packages and models you use.

RPI HARDWARE

-USB C Cable for powering (5A high power) -Power Bank To power remotely -microSD card (A1/A2, 32–128 GB) + USB card reader -micro-HDMI → HDMI cable (two if dual display) -Active cooler or case with fan -Raspberry Pi Camera Module 3 (Optional If you need visual feedback)

Room Policy

-1 member per team gets room access. -Don't use hardware not for the course (drones, other PIs, room hardware)