# TECHNICAL REPORT: AUTONOMOUS NAVIGATION AND MOBILE ROBOT SIMULATION

**Student:** Abdurrahman GÜLMEZ

**Student Email:** abdurrahmangulmez44@gmail.com

## 1.1. Project Overview

This project involves the development and simulation of an autonomous mobile robot using the ROS 2 (Robot Operating System) framework. The primary goal is to create a robot capable of Simultaneous Localization and Mapping (SLAM) and autonomous navigation within a simulated environment provided by Gazebo. The project utilizes the my_autonomous_robot package, specifically designed to integrate custom URDF models with standard ROS 2 navigation stacks.

## 1.2. Objectives

The core objectives of this simulation study are:

- **Physical Modeling:** Designing a custom robot chassis and sensor configuration using URDF/Xacro.
- **Environment Simulation:** Utilizing Gazebo to simulate real-world physics, collision detection, and sensor-environment interactions.
- **Real-Time Mapping:** Implementing SLAM using the slam_toolbox to generate accurate 2D maps of unknown environments.
- **Autonomous Navigation:** Configuring the Nav2 (Navigation 2) stack for global and local path planning and obstacle avoidance.
- **Mission Control:** Developing a Python-based node to manage autonomous tasks and coordinate navigation to specific goal locations.

## 2.1. Robot Modeling (URDF)

The robot's physical properties are defined in robot.urdf.xacro. The design follows a differential drive kinematics structure:

- **Base Chassis (base_link):** A box-shaped body with dimensions of 0.4m x 0.3m x 0.15m and a mass of 1.0 kg.

- **Wheels:** Two cylindrical tekerleks (wheels) with a radius of 0.05m and a length of 0.04m, mounted on continuous joints. The left wheel is offset at y=0.175 and the right wheel at y=-0.175 relative to the base.
- **Sensors (laser_frame):** A LIDAR sensor is modeled as a cylinder (0.05m radius, 0.04m length) and fixed at a position 0.1m forward and 0.1m high from the base center.

### 2.2. Simulation Environment

The simulation is conducted within the Gazebo physics engine. The environment is configured to use the TurtleBot3 waffle_pi model parameters as a baseline for environment interactions while using the custom robot model for physical maneuvers.

### 3.1. Package Dependencies

The project relies on several key ROS 2 packages:

- **rclpy:** The Python client library for ROS 2.
- **turtlebot3_gazebo:** Provides the simulation worlds.
- **slam_toolbox:** Used for mapping.
- **nav2_bringup:** Essential for launching the Navigation 2 stack.
- **explore_lite:** Included for future autonomous exploration enhancements.

### 3.2. Launch System

The system is orchestrated through Python launch files:

- **launch_sim.launch.py:** This file initializes the robot_state_publisher (RSP), spawns the robot entity into Gazebo, and opens RViz with a pre-configured profile (my_robot.rviz) for visualization.
- **project_master.launch.py:** Acts as the main entry point for full system operation. It sets the TURTLEBOT3_MODEL environment variable to waffle_pi and includes launch descriptions for Gazebo, slam_toolbox (in online_async mode), and the Nav2 stack with custom parameters.

### 4.1. Mapping and Localization

For mapping, the slam_toolbox is utilized with simulation time enabled. The online_async_launch.py configuration allows the robot to build a 2D occupancy grid map

while simultaneously localizing itself within the environment using LIDAR data and odometry.

### 4.2. Autonomous Task Execution (explorer_node.py)

The high-level autonomous logic is implemented in the ProjectAutonomousExplorer node:

- **Navigator Integration:** Uses the BasicNavigator from the nav2_simple_commander library to interact with the navigation stack.
- **Coordinate Goals:** The robot is programmed to navigate through a sequence of specific coordinates: (1.0, 1.0), (2.0, -1.0), and (-1.0, 1.5).
- **Task Management:** The node sends each goal to the Nav2 stack using goToPose(), waits for task completion via isTaskComplete(), and logs success or failure based on the getResult() output.
- **Optimization:** A critical fix was implemented to skip the initial AMCL service check loop, as the SLAM module handles the (0,0,0) initial pose automatically in this setup.

### 5.1. Performance Analysis

The simulation tests demonstrated high reliability in the robot's physical stability and navigation accuracy. The inertia values defined in the URDF (0.01 for the base and 0.001 for wheels) provided a realistic balance between responsiveness and weight. The slam_toolbox successfully handled loop closure, ensuring that the generated map remained consistent even after multiple passes through the same area.

### 5.2. Success Metrics

- **Mapping Accuracy:** The occupancy grid map generated in RViz closely matched the geometric layout of the Gazebo world.
- **Navigation Success:** The robot reached all three predefined coordinate points with minimal deviation from the planned path.
- **Stability:** No physical collisions or software crashes were observed during the multi-point mission.

### 5.3. Conclusion and Future Work

This project successfully bridged the gap between theoretical robot modeling and practical autonomous deployment within a ROS 2 environment. The modular architecture allows for easy expansion. Future work will focus on:

- Integrating dynamic obstacle avoidance for moving objects.
- Implementing the explore_lite package to automate the discovery of unknown map frontiers.
- Utilizing the multirobot_map_merge dependency to allow cooperative mapping with multiple robot entities.

Project Youtube Link: https://youtu.be/mZAzNPDXZd8