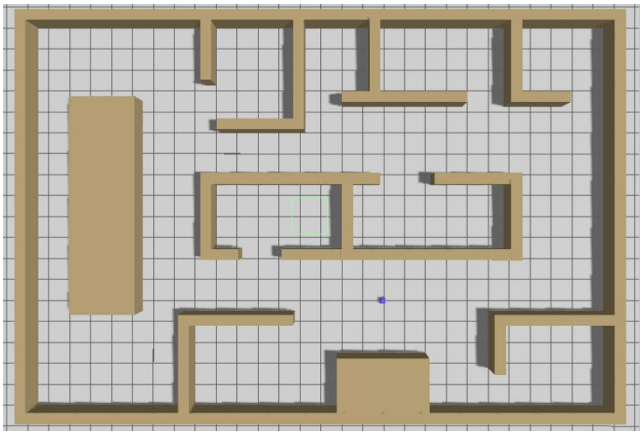
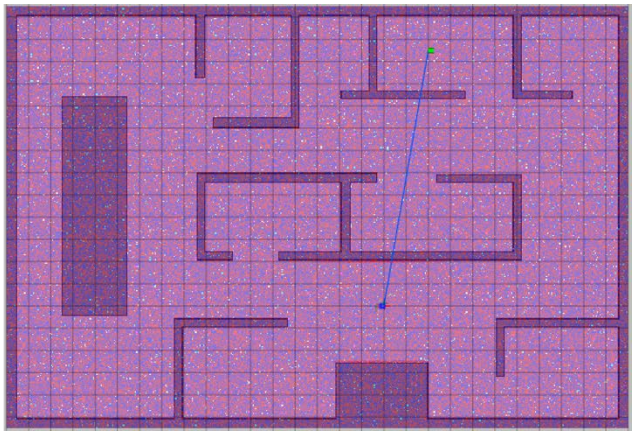
4TM00 Robot Motion Planning and Control
Group Project
Safe Robot Navigation under Intermitted Localization

In this assignment, you are asked to design a safe sensor-based robot navigation framework for the TurtleBot3 Waffle Pi mobile robot platform in the Gazebo simulator, as illustrated below. The objective is to successively visit a finite number of goal positions in a known, large, office-like environment under intermittent global localization at approximately 0.3 Hz (i.e., the */mocap/turtlebot/pose* topic is updated about every 3.33 seconds). To address this, you need to use an odometry method (e.g., based on dead reckoning or scan matching) to achieve accurate pose estimation at a higher rate (e.g., 10 Hz). The occupancy grid map of the environment is available on the */turtlebot/map* topic, published by a map server node. Each goal position is published on the */turtlebot/goal* topic by a goal pose publisher node and is automatically updated to the next goal in the list once the robot is within 0.5 m of the current goal position. Thus, your overall navigation framework should consist of odometry, costmap, path planner, and path-following blocks to visit all reachable goal poses as quickly as possible.



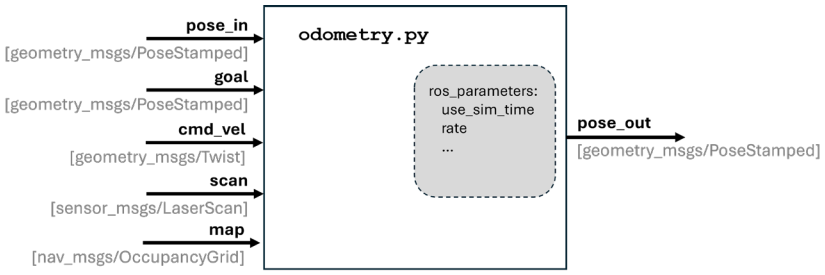


A specific ROS package, *core_tue4tm00_project*, is provided in the 4TM00 GitLab repository for this project. It includes template files for your ROS node scripts, parameter configurations, and a demo launch file that integrates all your designs for a demonstration. Please copy and paste core_tue4tm00_project into your group's GitLab repository and rename it with your group name (e.g., *group0_tue4tm00_project*). Ensure that your final demo code runs properly with the 4TM00 course software. A demo video of the project is available on Canvas at https://canvas.tue.nl/files/6026980.

Part 1) *(Odometry)* In the first part of the assignment, you are tasked with designing a ROS node that receives the robot's pose at a low frequency (e.g., 0.3 Hz) and publishes pose estimates at a higher frequency (e.g., 10 Hz) to minimize delays in planning and control for navigation. You may use the robot's command velocity, scan measurements, and the occupancy map of the environment and available coordinate transformations (e.g., *tf*). A schematic diagram of the ROS node is provided below.

The project ROS package, *core_tue4tm00_project*, includes:

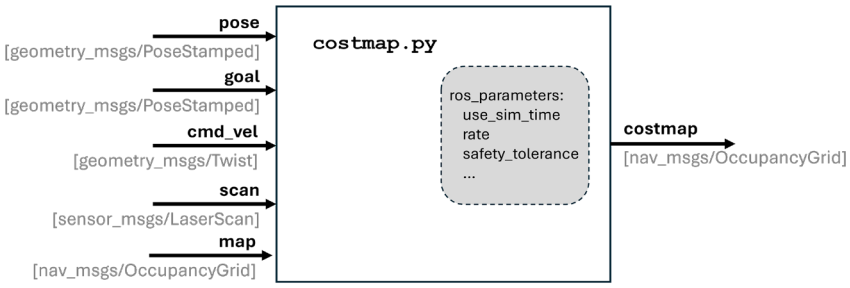
- A ROS node script template: *scripts/odometry.py*
- A parameter configuration file: *config/odometry.yaml*
- An interface launch file: *launch/odometty.launch.py*



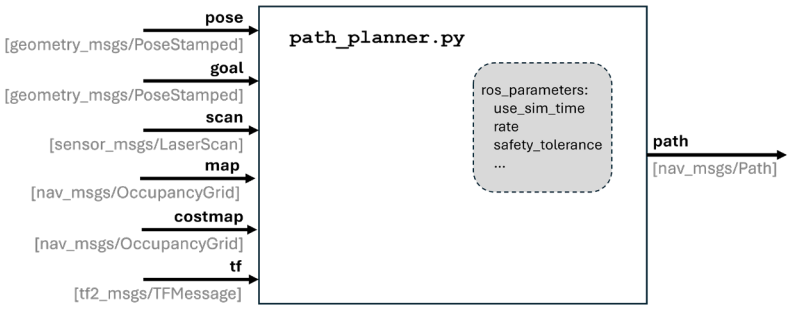
Part 2) *(Costmap)* In the second part of the assignment, you are asked to design a ROS node that receives an occupancy grid map and generates a safe navigation costmap for reference path planning. This may involve using the robot's scan measurements, command velocity, pose, and goal pose to ensure safe and effective navigation around obstacles, as shown in the schematic diagram of the ROS node below. Note that you can build upon and further improve your results from Assignments 1–3.

The project ROS package, *core_tue4tm00_project*, includes:

- A ROS node script template: *scripts/ costmap.py*
- A parameter configuration file: *config/ costmap.yaml*
- An interface launch file: *launch/ costmap.launch.py*



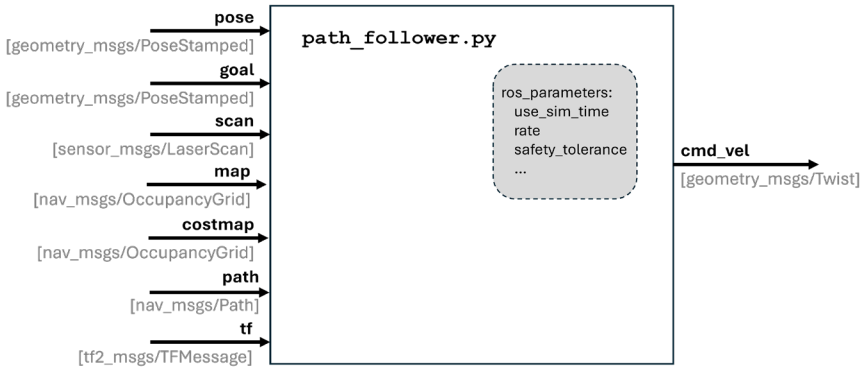
Part 3) *(Path Planner)* In the third part of the assignment, you are tasked with designing a ROS node that receives the robot's pose and a goal position, then generates a safe navigation path over an occupancy grid map. The path should guide the robot safely and effectively to reach the goal as quickly as possible. You may use the robot's costmap, scan measurements, and command velocity, as illustrated in the schematic diagram below. It is important to note that it may not always be possible to reach all given goal positions in a complex environment, and executing reference paths that are too close to obstacles can be challenging. Therefore, your objective is to find a reference path with adequate clearance from obstacles; if no such path exists, indicate that no safe navigation path is available and wait for a new goal. Note that you can build upon and further improve your results from Assignments 1–3.



The project ROS package, *core_tue4tm00_project*, includes:

- A ROS node script template: *scripts/ path_planner.py*
- A node parameter configuration file: *config/ path_planner.yaml*
- An interface launch file: *launch/path_planner.launch.py*

Part 4) *(Path Follower)* In the fourth part of the assignment, you are asked to design a ROS node that receives a reference path toward a global goal position and generates command velocities to safely follow it while avoiding sensed obstacles. This can be achieved using the robot's scan measurements, pose, goal pose, the occupancy map of the environment, and the navigation costmap, as shown in the schematic diagram below. It is important to note that it may not be possible to safely follow all reference paths from start to end in complex environments. Therefore, your objective is to follow the path as much as possible. If the path cannot be followed safely, you should indicate that replanning is needed (e.g., by printing a message on the terminal). If a new path is planned due to a new goal, your path-following method should accommodate this and continue following the new path. Note that you can build upon and further improve your results from Assignments 1–3.



The project ROS package, *core_tue4tm00_project*, includes:

- A ROS node script template: *scripts/ path_follower.py*
- A parameter configuration file: *config/ path_follower.yaml*
- An interface launch file: *launch/ path_follower.launch.py*

Part 5) *(Multi-Goal Navigation)* The overall objective of this project is to systematically and properly integrate odometry, costmap, path planner, and path follower into a complete safe navigation framework that can successively visit a finite number of goal positions, published one after another, once the robot is within 0.5 m of the current goal position, under intermitted global localization. You are expected to demonstrate that your framework can visit all reachable goal points as quickly as possible. Please report how long it takes you to visit the three given example goal poses. You may also provide additional examples. Your demo files will also be tested with another set of goal poses. Please see the example demo video on Canvas at https://canvas.tue.nl/files/6026980.

The project ROS package, *core_tue4tm00_project*, includes

- A demo launch file: *launch/ demo_navigation.launch.py*
- A configuration file for goal points: *config/ goal_poses.yaml*

For all parts, please clearly describe your design approach and reasoning in the report. Your solution approach should be clear, concise, and justified. Please systematically demonstrate the pros and cons of your design in your report and presentation. Ensure that any figures and videos included are neat (e.g., with captions, proper labels, readable axes, and adequately thick lines). Provide all the necessary information to reproduce your results. The report should be concise and to the point. Your code should be clean, well-commented, and as modular as possible to ensure reusability by you and other. Please make sure that your code runs properly with the 4TM00 course software.