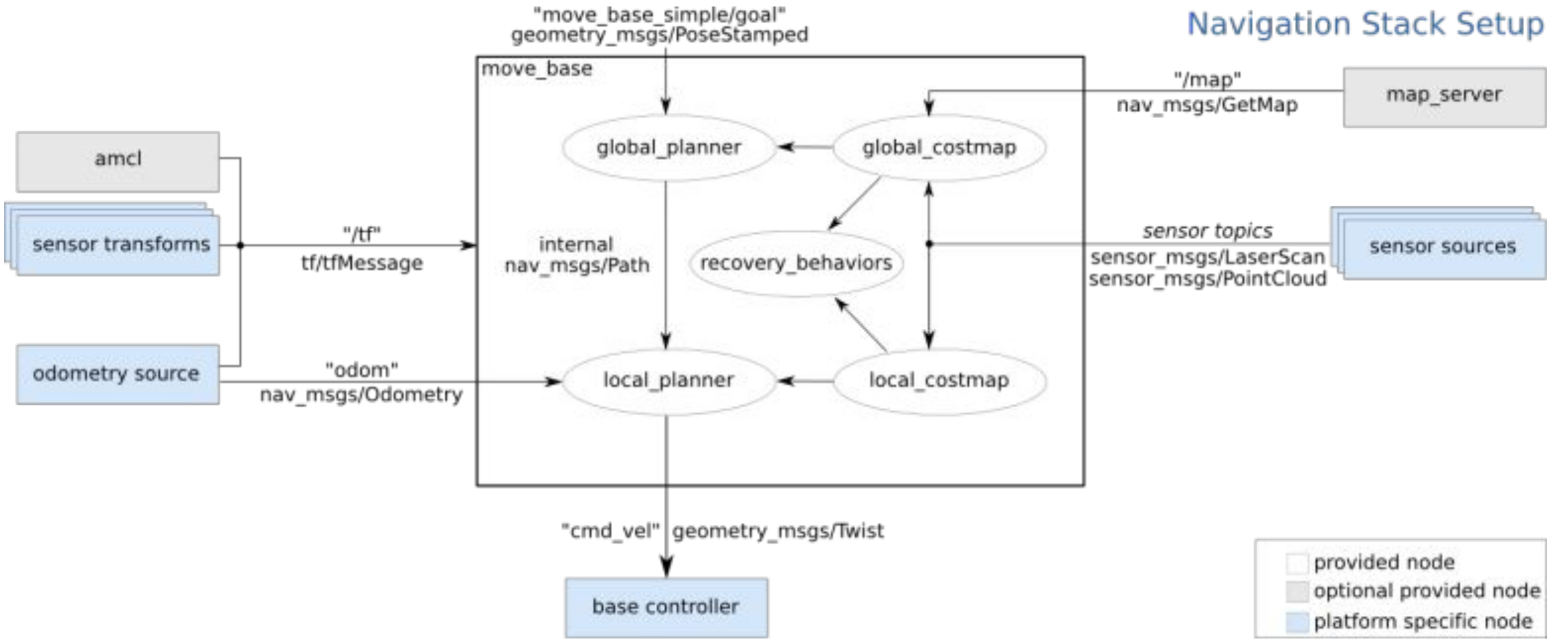
ROS Navigation

Lab 4 - Autonomous Robots

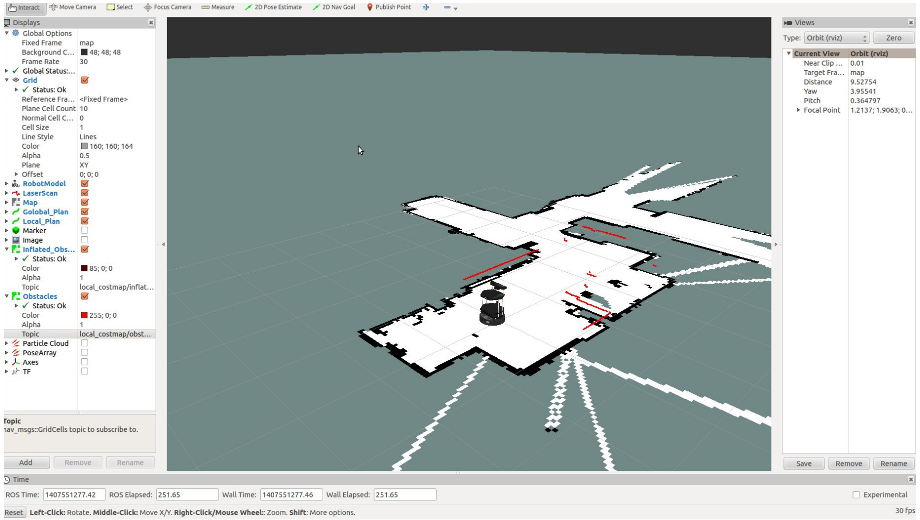
Navigation stack in ROS

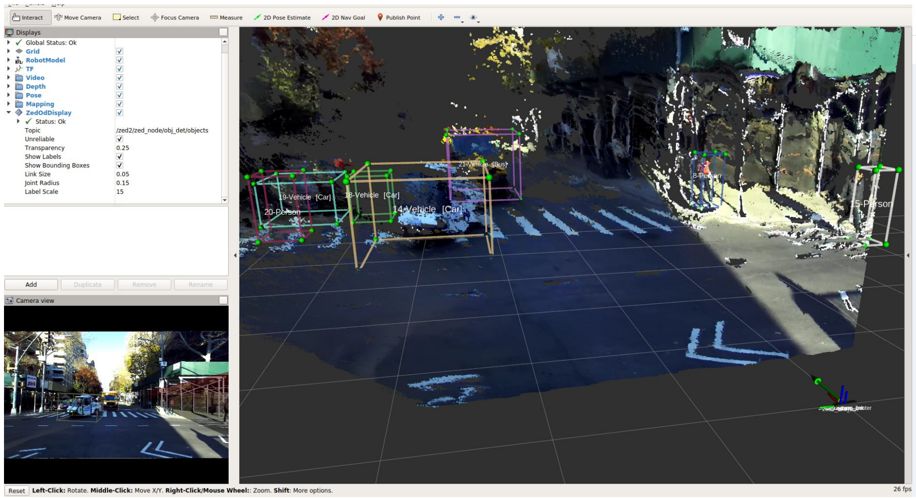
It uses odometry, sensor data, and a goal pose to give safe velocity commands.



RVIZ

rviz is a 3D visualizer for the Robot Operating System (ROS) framework

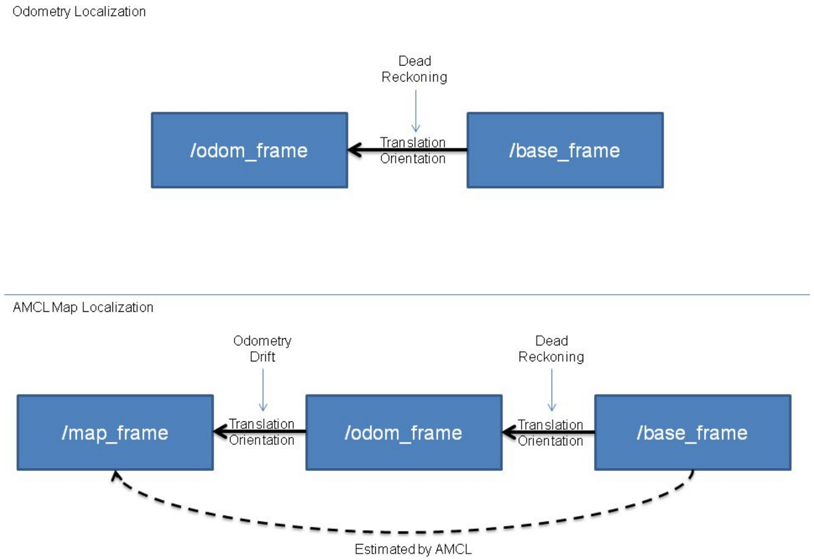


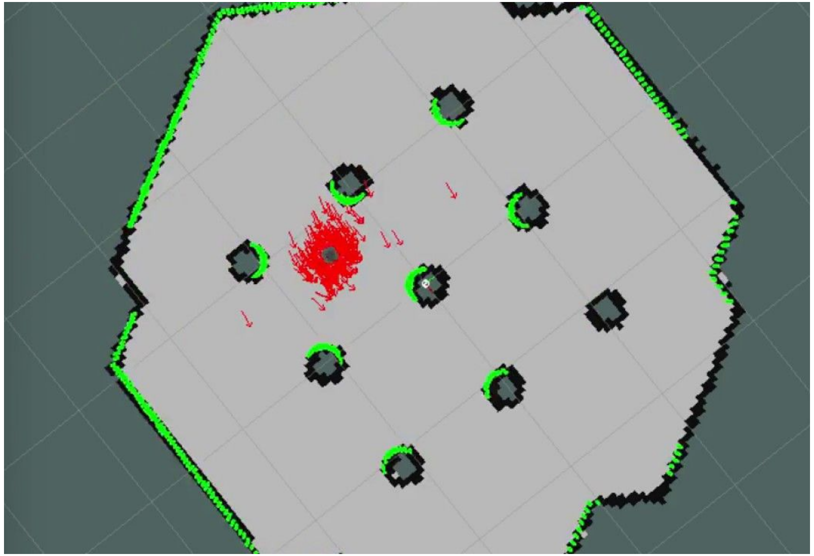


AMCL

amcl is a probabilistic localization system for a robot moving in 2D.

amcl takes in a laser-based map, laser scans, and transform messages, and outputs pose estimates.

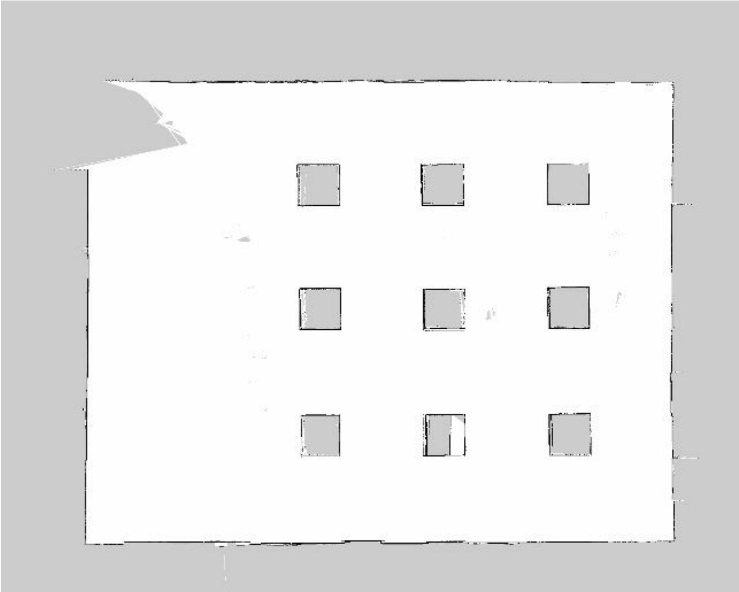




map_server

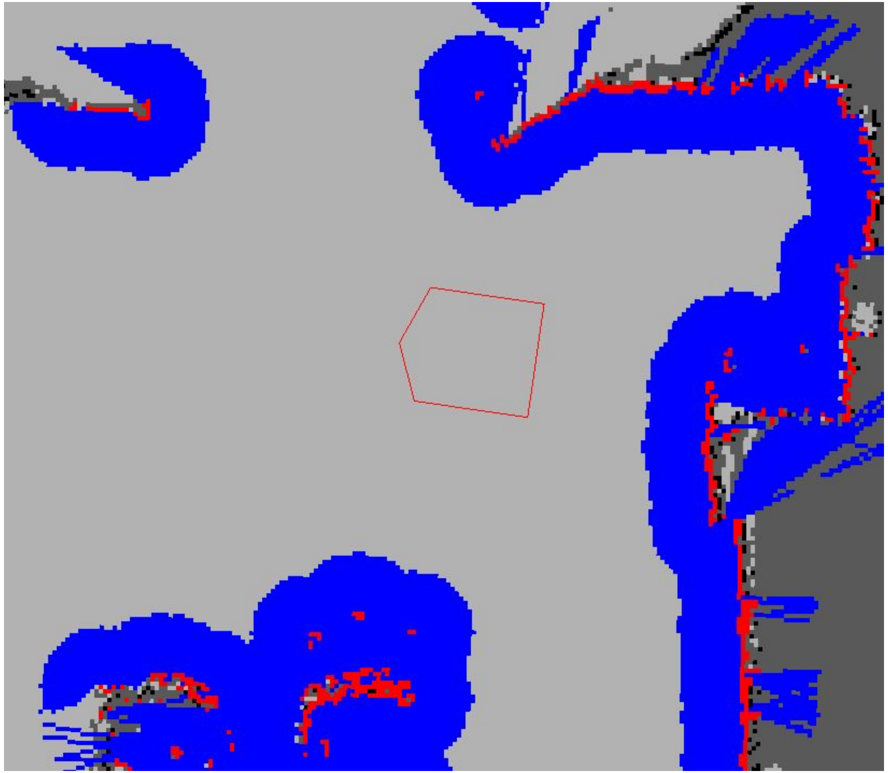
mapmap_server is a ROS node that reads a map from disk and offers it via a ROS service.

- **image** : Path to the image file containing the occupancy data; can be absolute, or relative to the location of the YAML file
- **resolution** : Resolution of the map, meters / pixel
- **origin** : The 2-D pose of the lower-left pixel in the map, as (x, y, yaw), with yaw as counterclockwise rotation (yaw=0 means no rotation). Many parts of the system currently ignore yaw.
- **occupied_thresh** : Pixels with occupancy probability greater than this threshold are considered completely occupied.
- **free_thresh** : Pixels with occupancy probability less than this threshold are considered completely free.
- **negate** : Whether the white/black free/occupied semantics should be reversed (interpretation of thresholds is unaffected)



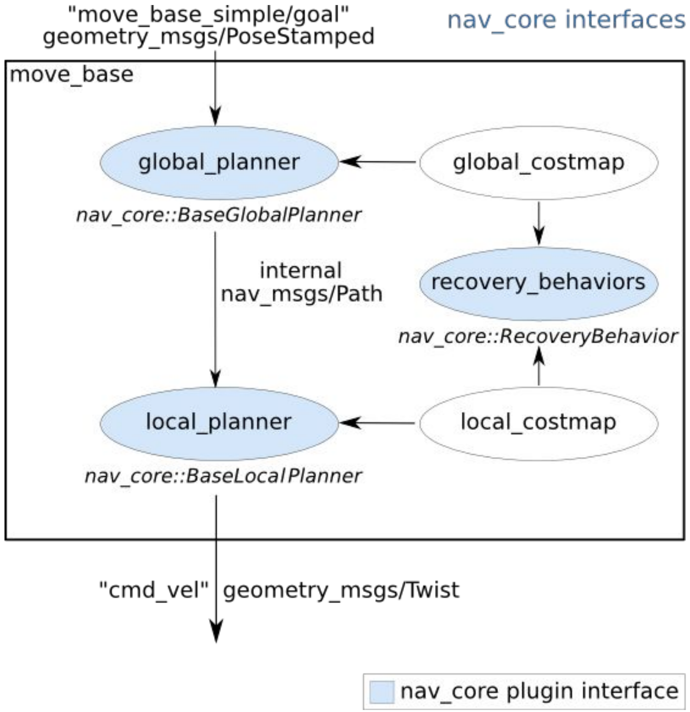
costmap_2d

- Implements a 2D grid-based costmap for environmental representations
- It is used in the planner and controller servers for creating the space to check for collisions or higher cost areas to negotiate around.
- Provides a configurable structure that maintains information about where the robot should navigate in the form of an occupancy grid.
- Each cell in the costmap can have different cost values. Specifically, each cell in this structure can be either free, occupied, or unknown.



Move_base

- Represents the core of the navigation stack
- It handles all the planning layers, including the reactive navigation and high-level planner.
- Given a goal, read from the iit computes and publishes the command velocities to drive the robot
- To compute the velocities it needs pose feedback from the AMCL, odometry data and information from the robot sensors.
- This node is highly configurable, since it enables the use of different global and local planners.



Exercise 1

- Download pXX_arob_lab4 package from github https://github.com/luisriazuelo/pXX_arob_lab4.git and include it into your workspace.

Exercise 2

- Launch the file arob-p4-navigation-rviz.launch that includes an instance of rviz.
- Observe the value of the different topics sending different goals to the robot.

Exercise 3

Analyze how some parameters can affect robot navigation:

- Try different global and local planners (planner_selection.yaml)
- Number of particles in AMCL
- Use the dynamic obstacle to make it difficult for the robot!

rosbag command-line tool: rosbag record

- **$ rosbag record -h** ← **Display all the available options for saving the information.**
- **$ rosbag record -a** ← **Store all the topics published. Press ctrl+c for stopping the execution.**
- **$ rosbag info name_of_the_file.bag** ← **Show the información contained on the file.**

rosbag command-line tool: rosbag play

- **$ rosbag play -h** ⟵ **Display all the available options for playing the information.**

- **$ rosbag play name_of_the_file.bag** ⟵ **Reproduce the information of all the topics.**

Exercise 4

- Explore the files llc_local_planner.h (in folder include/arob_lab4) and llc_local_planner.cpp (in folder src) to understand the functions defined.
- Register the controller as a plugin and export it in the ROS system.

Exercise 5

Complete functions in llc_local_planner.cpp file to implement your low level controller:

- computeVelocityCommands()
- isGoalReached()

Exercise 6

Launch file arob-p4-navigation-plugin.launch with llc_local_planner as the local planner and send different goals to evaluate the robot behavior:

Exercise 7

Testing the code implemented in the previous exercise on a real platform:

- **Send** the package "pXX_arob_lab4" **to one of the robots** available for real-world testing.

```
$ scp -r pXX_arob_lab4 arob@ip_robot:arob_ws/src/
```

- To launch the code on the robot, you should use the launch file real_robot.launch.

ip address:

- turtlebot1: 10.1.31.215

robot user account:

- user: arob
- password: unizar

Laboratory 4 evaluation

- **Submit** the **code** for all exercises. Send the complete *pxx_arob_lab4* package before the beginning of the next session.
- **Run** Exercise 7 on the **real platform**.
- **Multiple-choice test** through Moodle at the **beginning** of the **next session**. Test will be conducted **individually**, **without** any **extra material.**