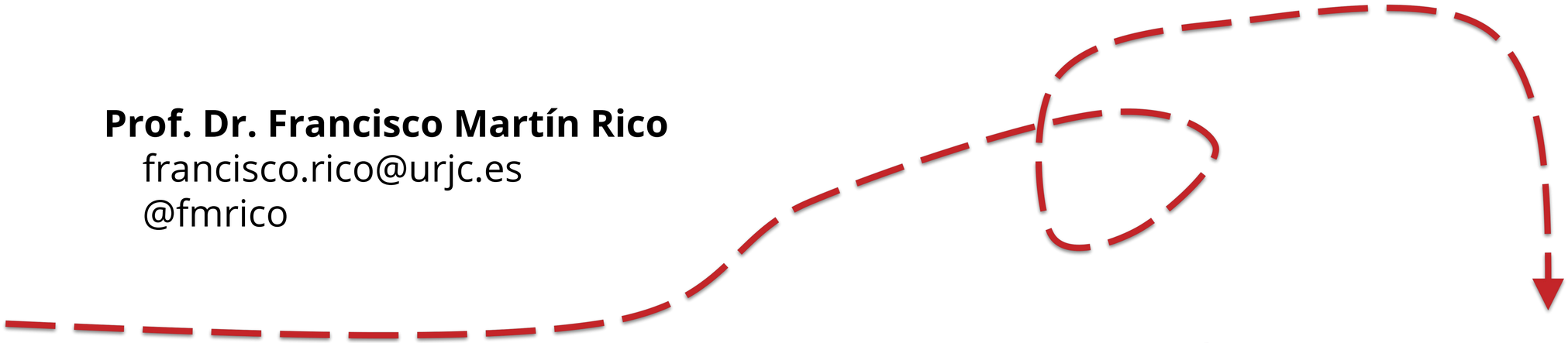
Easy Nav

An open-source framework for navigating everywhere

Prof. Dr. Francisco Martín Rico
francisco.rico@urjc.es
@fmrico

Dr. Francisco Miguel Moreno
franciscom.moreno@urjc.es
@butakus



FOSDEM

Intelligent Robotics Lab

Universidad Rey Juan Carlos

Intelligent Robotics Lab

Universidad Rey Juan Carlos

Francisco Martín Rico
fmrico

Follow

Doctor Engineer on Robotics. Full Professor at Rey Juan Carlos University. Leader of the Intelligent Robotics Lab

586 followers · **14** following

Rey Juan Carlos University
Madrid
www.gsyc.es/~fmartin
@fmrico

Achievements

x3 x3

Overview | Repositories 161 | Projects | Packages | Stars 22

Pinned

PlanSys2/ros2_planning_system Public
This repo contains a PDDL-based planning system for ROS2.
C++ 462 106

ros2_dotnet Public
Forked from ros2-dotnet/ros2_dotnet
.NET bindings for ROS2
C# 1

cascade_lifecycle Public
Managed nodes (or lifecycle nodes, LN) package provides a mechanism to create LifecycleNode activation trees
C++ 82 13

navigation2 Public
Forked from ros-navigation/navigation2
ROS2 Navigation
C++ 1 1

rclcpp Public
Forked from ros2/rclcpp
rclcpp (ROS Client Library for C++)
C++

popf Public
The POPF planner from KCL planning group with some modifications to make it work with "modern" compilers...
C++ 22 8

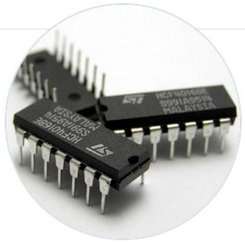
1,446 contributions in the last year

Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan
Mon
Wed
Fri
Learn how we count contributions
Less More

Contribution activity

2026
2025

January 2026



Francisco Miguel Moreno
Butakus

Follow

15 followers · **4** following

Spain

Achievements

x2

Highlights

PRO

Organizations

Block or Report

Overview | Repositories 36 | Projects | Packages | Stars

Pinned

ros2-env Public
A zsh plugin to manage ROS 2 workspaces
Shell 16 3

pikaball-revamped Public
A Revamped version of the Pikachu Volleyball game - Murcia LAN Party Edition
C++ 2 1

EasyNavigation Public
Forked from EasyNavigation/EasyNavigation
C++

ArduComm Public
A data link layer protocol and library for advanced serial communication between a host computer and 8-bit Arduino boards.
Python 1

landmark_placement_optimization Public
Python 1

lsi-uc3m/hypergrid Public
Hypergrid is a ROS package for building local maps from multiple sensors in the blink of an eye
C++ 19 8

405 contributions in the last year

Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan
Mon
Wed
Fri
Learn how we count contributions
Less More

@EasyNavigation @ros @IntelligentRoboticsLa... More

Activity overview

Contributed to **Butakus/pikaball-revamped**, **EasyNavigation/EasyNavigation**, **Butakus/ros2-env** and 26 other repositories

4%
Code review

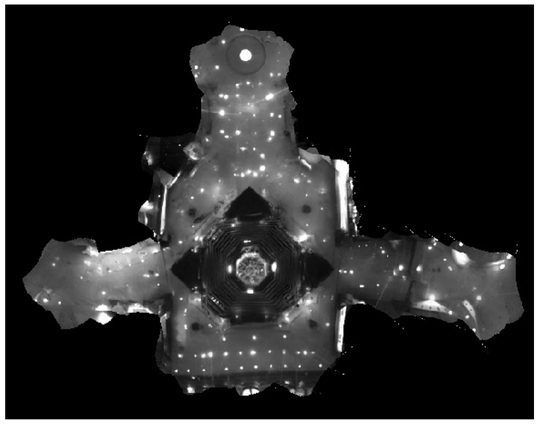
2026
2025
2024
2023
2022
2021
2020
2019

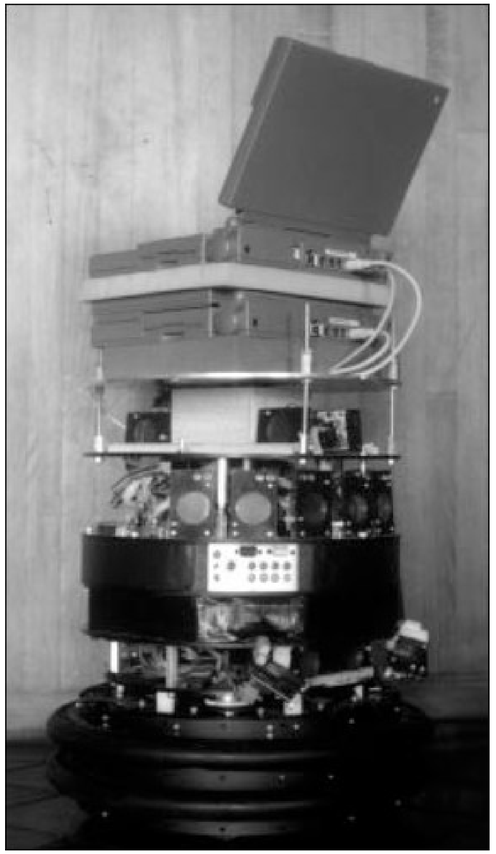
Introduction

Some background…

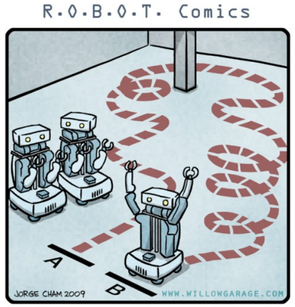
Introduction
Robot navigation

- It is a fundamental skill for a mobile robot
- A very active field since more than half a century ago
- ¿Is it solved?
- Lack of standards → Re-implementation from scratch
- ROS





Introduction
ROS 1: Move Base



"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

The Office Marathon: Robust Navigation in an Indoor Office Environment

Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, Kurt Konolige

ICRA 2010

[Link to video]

The Office Marathon:
Robust Navigation in an Indoor Office Environment

Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, Kurt Konolige

Willow Garage Inc., USA
{eitan,berger,tfoote,gerkey,konolige}@willowgarage.com

***Abstract*— This paper describes a navigation system that allowed a robot to complete 26.2 miles of autonomous navigation in a real office environment. We present the methods required to achieve this level of robustness, including an efficient Voxel-based 3D mapping algorithm that explicitly models unknown space. We also provide an open-source implementation of the algorithms used, as well as simulated environments in which our results can be verified.**

I. Introduction

We study the problem of robust navigation for indoor mobile robots. Within this well-studied domain, our area of interest is robots that inhabit unmodified office-like environments that are designed for and shared with people. We want our robots to avoid all obstacles that they might encounter, yet still drive through the tightest spaces that they can physically fit. We believe that reliable navigation of this kind is a necessary prerequisite for any useful task that an indoor robot might perform.

While many robots have been shown to navigate in office-like environments, existing approaches invariably require some modification of the environment, or do not allow the robot to negotiate tight spaces. Most indoor robots rely on a planar or quasi-planar obstacle sensor, such as a laser range-finder or sonar array. Because vertical structure dominates man-made environments, these sensors are positioned on the robot to detect obstacles along a horizontal slice of the world. The result is a robot that can easily avoid walls but will miss chair legs, door handles, table tops, feet, etc. Collisions are avoided by either modifying the environment (e.g., removing chairs, covering tables with floor-length tablecloths), or adding artificial padding (e.g., inflate obstacles by the maximum expected length of a person's foot), which prevents

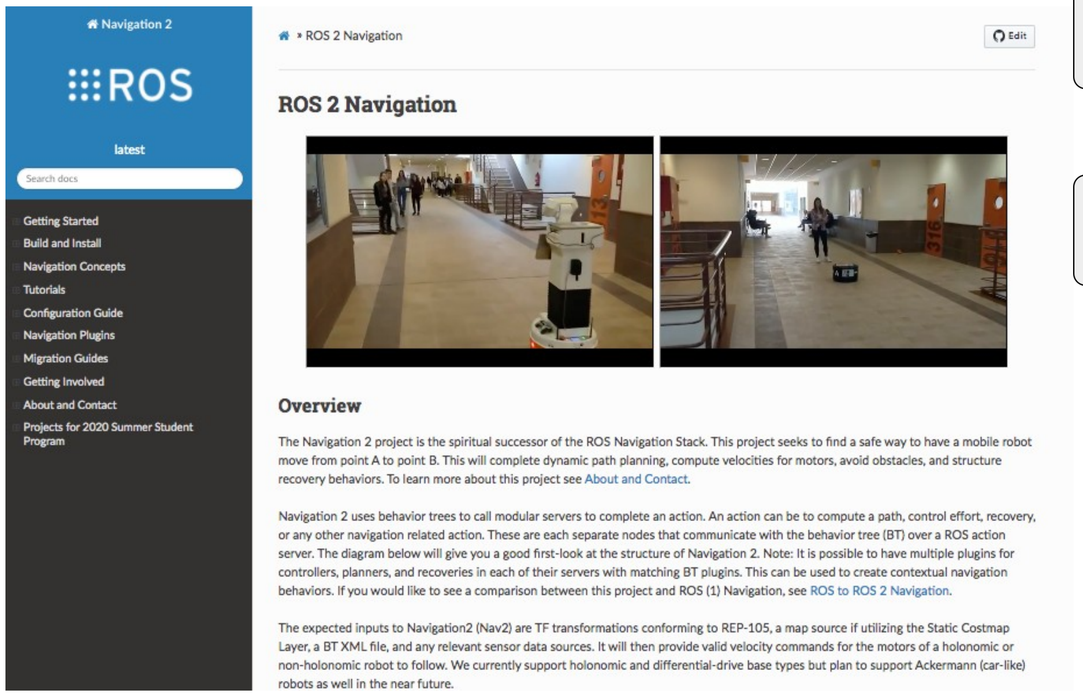
Fig. 1. The PR2 avoiding a table after passing through a narrow doorway.

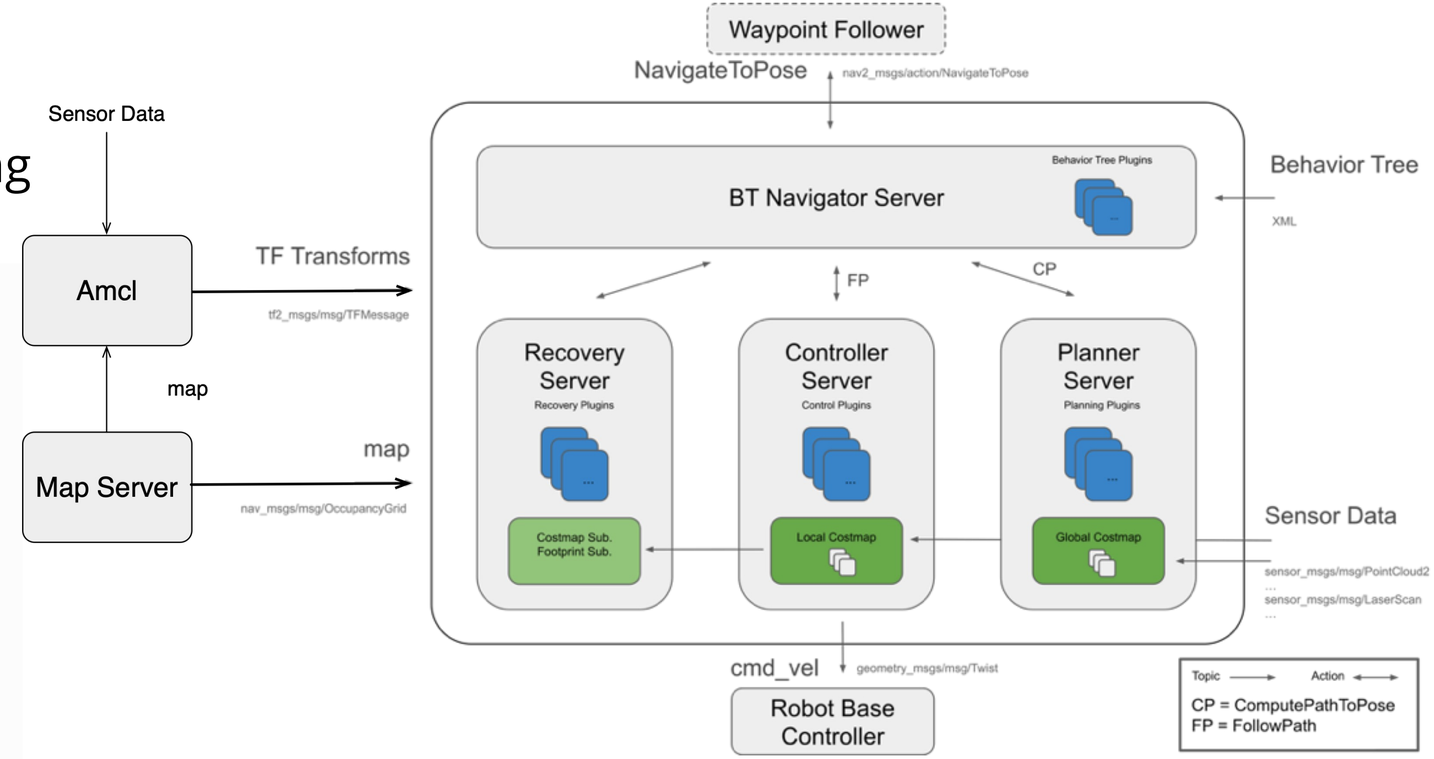
through the tightest spaces that the robot can fit. At the core of our work is an efficient technique for constructing, updating, and accessing a high-precision three-dimensional Voxel Grid. This structure encodes the robot's knowledge about its environment, classifying space as free, occupied, or unknown. Using this grid, the robot is able to plan and execute safe motions in close proximity to obstacles.

Through extensive experimentation, we have established that our approach is safe and robust. During endurance runs with the PR2, we regularly left a robot running unattended overnight in our office building, which was not modified to accommodate the robot. The software described in this paper is available under an open-source license,[1] and we encourage others to experiment with and use our code.

Introduction
ROS 2: Nav2

- Main author: Steven Macenski
- Move Base stack reimplementation
- An example of good practices in ROS 2
- More algorithms, plugins, components…
- Contributions: 59 collaborators, Intel, Samsung





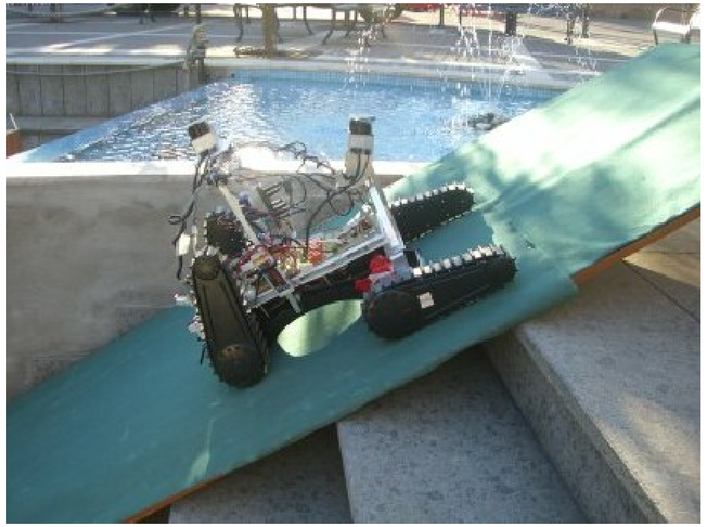
Introduction
ROS 2: Nav2

[Link to video]

EasyNav

Beyond Nav2...

EasyNav
Motivation



EasyNav
¿Why?

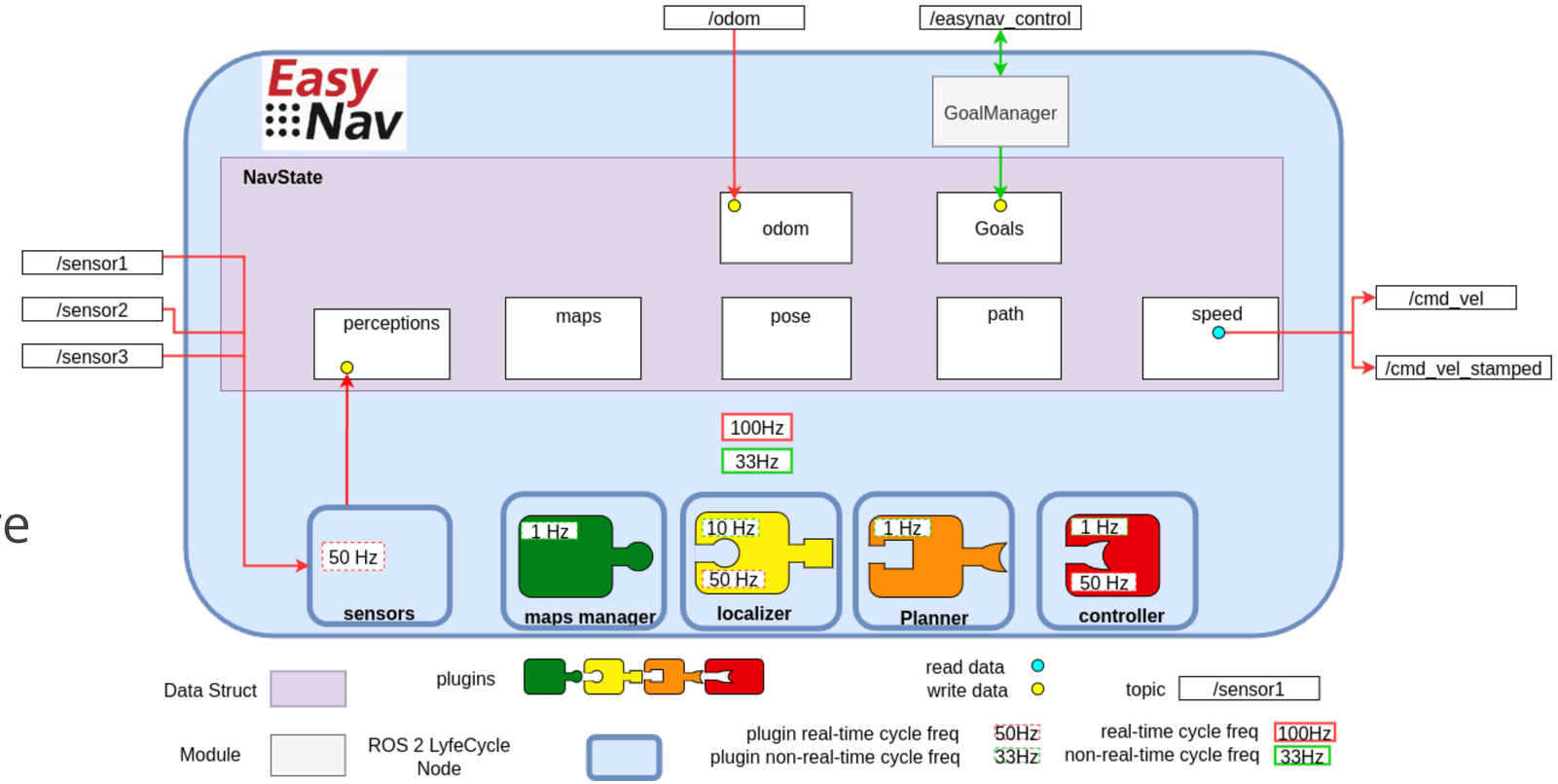
- We work with some outdoor robots, and hacking Nav2 is not always an option.
- We wanted to offer multiple options to the community with the ability to adapt to different scenarios and restrictions.
- We believe in free and open source. Open source is about the freedom to choose. By providing alternatives, we strengthen the entire ROS 2 community.
- EasyNav does not aim to replace Nav2, which we admire and which served as an inspiration. Our goal is to offer flexibility and to satisfy specific requirements in areas where Nav2 would be less appropriate.

EasyNav
In action

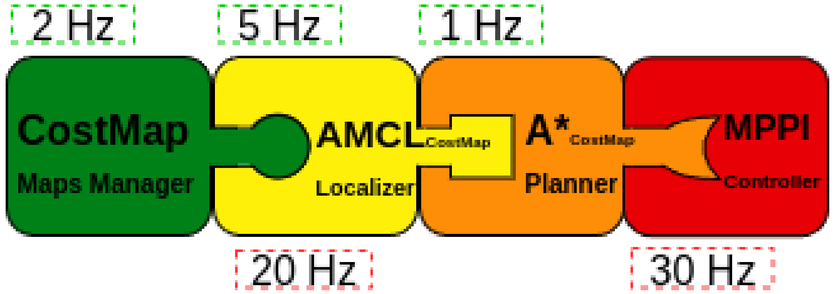
Navigating with the **Costmap Stack**

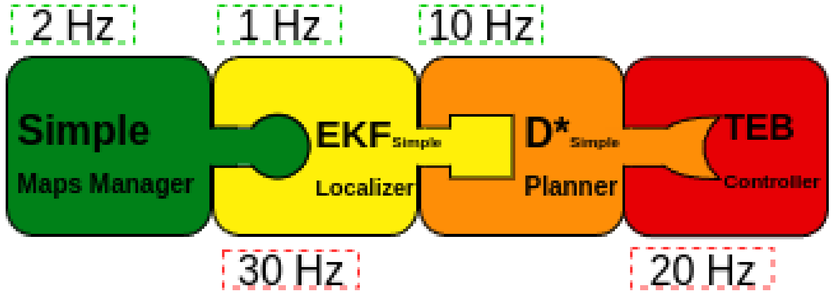
EasyNav
Design

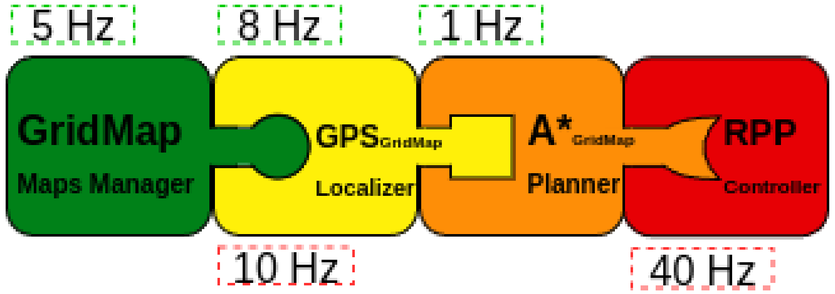
- Independent from representation
- Real-Time execution
- Light and easy to use:
  - Single binary executable
  - Single config file
- Modular, plugin-based architecture and reusable navigation stacks.

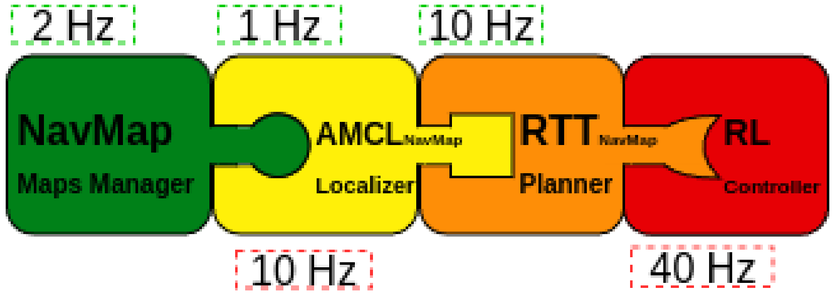


EasyNav
Plugin composition

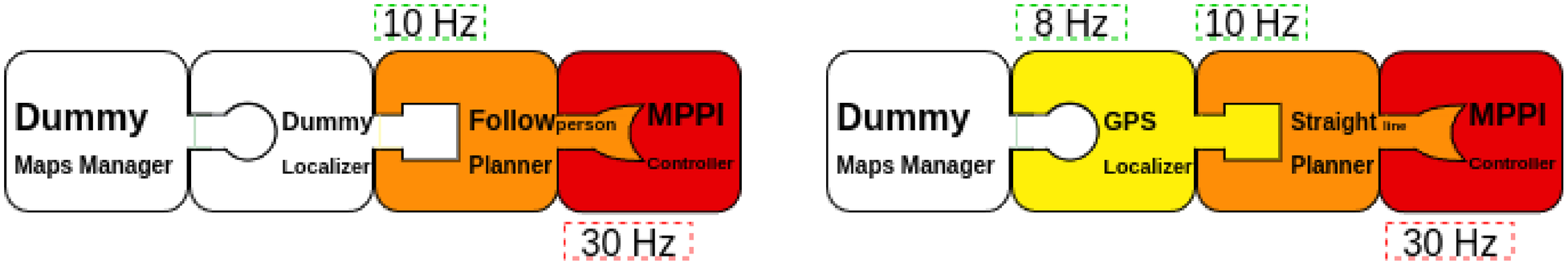








EasyNav
Plugin composition



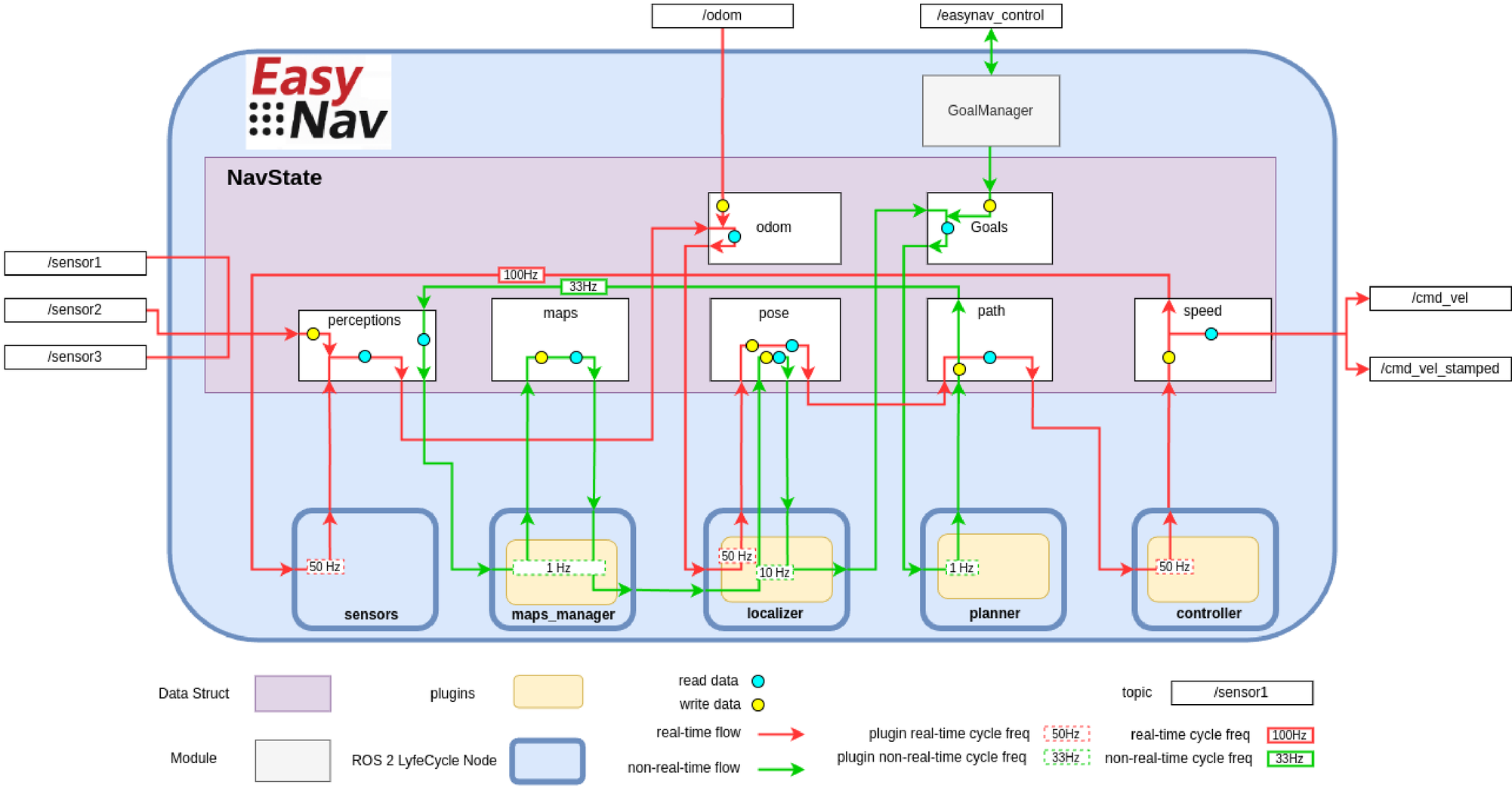
EasyNav
Plugin composition

```yaml
controller_node:
  ros__parameters:
    use_sim_time: true
    controller_types: [simple]
    simple:
      rt_freq: 30.0
      plugin: easynav_simple_controller/SimpleController
      max_linear_speed: 0.6
      max_angular_speed: 1.0
      look_ahead_dist: 0.2
      k_rot: 0.5

localizer_node:
  ros__parameters:
    use_sim_time: true
    localizer_types: [simple]
    simple:
      rt_freq: 50.0
      freq: 5.0
      reseed_freq: 1.0
      plugin: easynav_simple_localizer/AMCLLocalizer
      num_particles: 100
      noise_translation: 0.05
      noise_rotation: 0.1
      noise_translation_to_rotation: 0.1
      initial_pose:
        x: 0.0
        y: 0.0
        yaw: 0.0
        std_dev_xy: 0.1
        std_dev_yaw: 0.01
```

```yaml
maps_manager_node:
  ros__parameters:
    use_sim_time: true
    map_types: [simple]
    simple:
      freq: 10.0
      plugin: easynav_simple_maps_manager/SimpleMapsManager
      package: easynav_indoor_testcase
      map_path_file: maps/home.map

planner_node:
  ros__parameters:
    use_sim_time: true
    planner_types: [simple]
    simple:
      freq: 0.5
      plugin: easynav_simple_planner/SimplePlanner
      robot_radius: 0.3

sensors_node:
  ros__parameters:
    use_sim_time: true
    forget_time: 0.5
    sensors: [laser1]
    perception_default_frame: odom
    laser1:
      topic: /scan_raw
      type: sensor_msgs/msg/LaserScan
      group: points

system_node:
  ros__parameters:
    use_sim_time: true
    position_tolerance: 0.1
    angle_tolerance: 0.05
```

EasyNav
Execution model

- Single Process
- 1 Thread RT
  - Sensor Update
  - Pose predict
  - Control
- 1 Thread no-RT
  - Maps Update
  - Pose correct
  - Path Planner
- 1 TF Buffer RT
  - Shared TF buffer



EasyNav
Composing map representations

- The MapsManager module supports multiple concurrent representations
- Different plugins may require / use different environment representations
- The representations may work together for a common task
- The plugins for the MapsManager implementations may also include layers or filters.

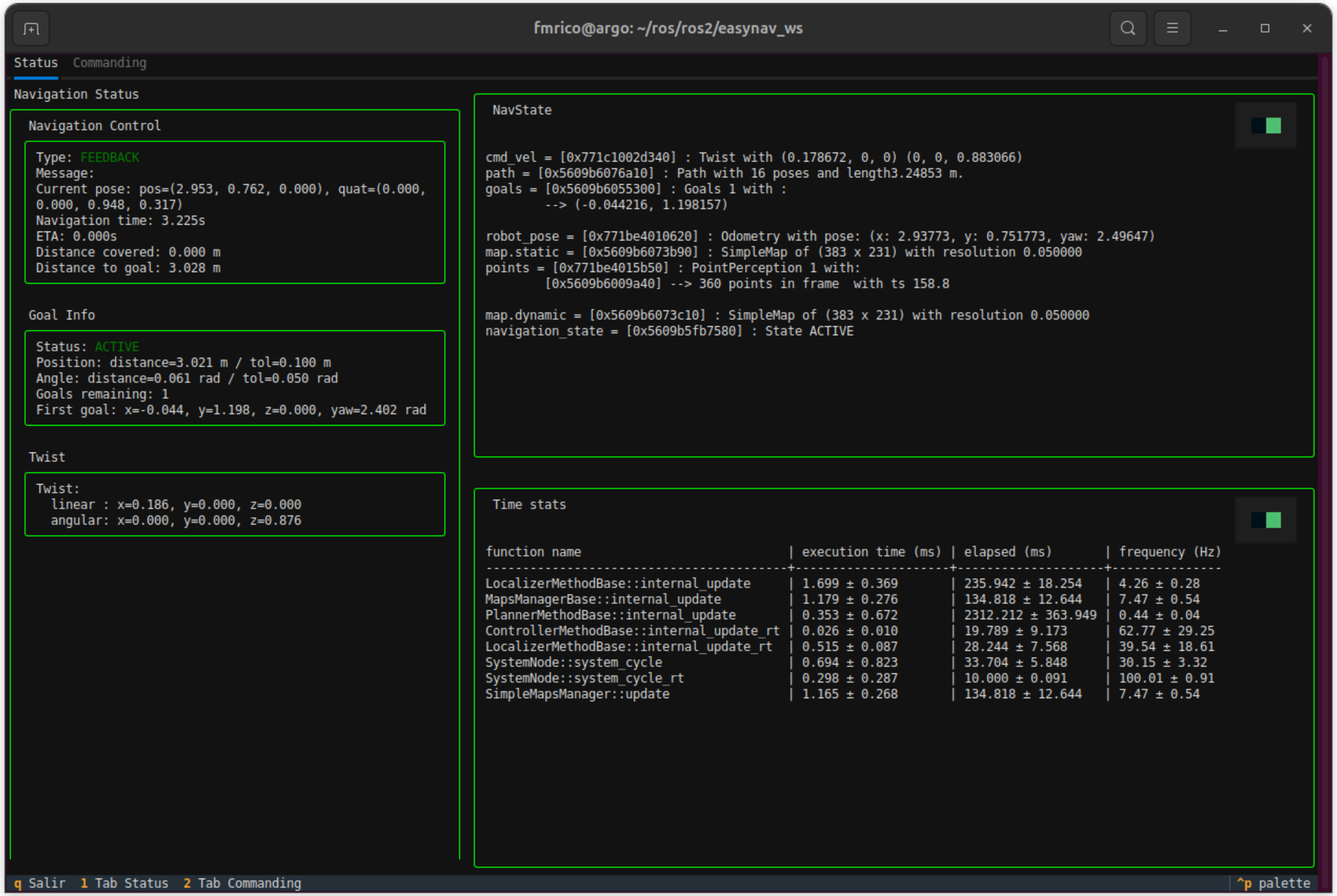
```yaml
maps_manager_node:
  ros__parameters:
    use_sim_time: true
    map_types: [bonxai, navmap]
    bonxai:
      freq: 10.0
      plugin: easynav_bonxai_maps_manager/BonxaiMapsManager
      package: easynav_indoor_testcase
      bonxai_path_file: maps/excavation_urjc.pcd
    navmap:
      freq: 10.0
      plugin: easynav_navmap_maps_manager/NavMapMapsManager
      package: easynav_indoor_testcase
      navmap_path_file: maps/excavation_urjc.navmap
      filters: [obstacles, inflation]
      obstacles:
        plugin: easynav_navmap_maps_manager/NavMapMapsManager/ObstaclesFilter
      inflation:
        plugin: easynav_navmap_maps_manager/NavMapMapsManager/InflationFilter
        inflation_radius: 5.0
        cost_scaling_factor: 1.0
```

EasyNav

In action

Navigating with the **NavMap Stack**

EasyNav
Tools: TUI and CLI



EasyNav
Tools: TUI and CLI

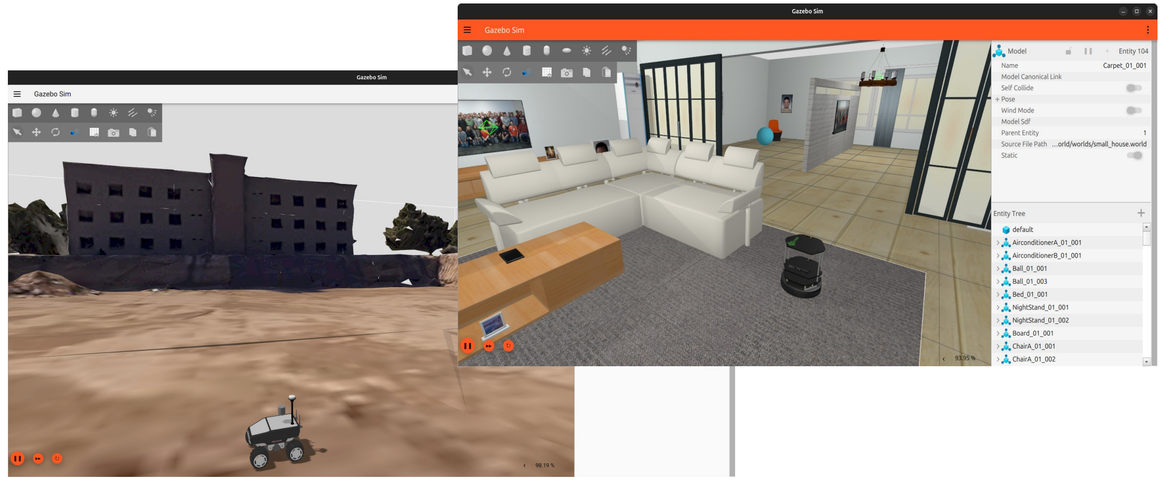
```
ros2 easynav -h
ros2 easynav plugins -h
ros2 easynav nav-state -h
# ... etc.
```

```
ros2 easynav plugins [--mapsmanager] [--localizer] [--planner] [--controller]
                     [--costmap-filters] [--navmap-filters]
                     [--grep SUBSTR] [--show-lib] [--show-xml]
                     [--json] [--pretty] [--debug]
```

```
# Quick inventory of installed EasyNav plugins
ros2 easynav plugins

# Inspect planner plugins and search for "astar"
ros2 easynav plugins --planner --grep astar

# Monitor NavState for 45 seconds
ros2 easynav nav-state --duration 45

# Watch goal manager events while sending goals from RViz
ros2 easynav goal-info --duration 60

# Observe control loop values
ros2 easynav navigation-control --duration 30

# Tail robot velocities
ros2 easynav twist --duration 10

# Render periodic time-stats with screen refresh
ros2 easynav timetats --duration 20
```

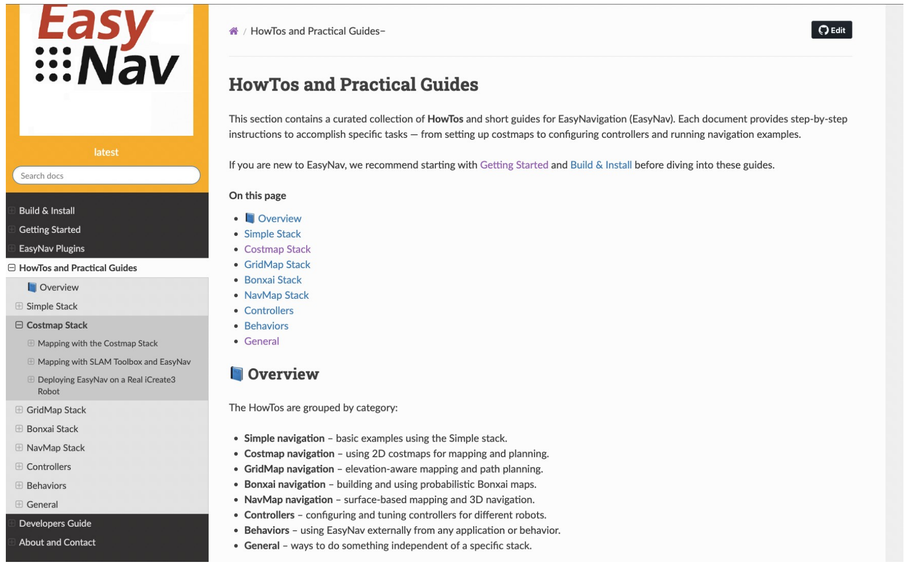
EasyNav
Repositories

- Code hosted on github.com/EasyNavigation
- GPLv3 license
- CI pipelines for multiple distros
- Binary releases for ROS 2 Jazzy and Kilted
- Simulation playgrounds in Gazebo with different worlds and robots

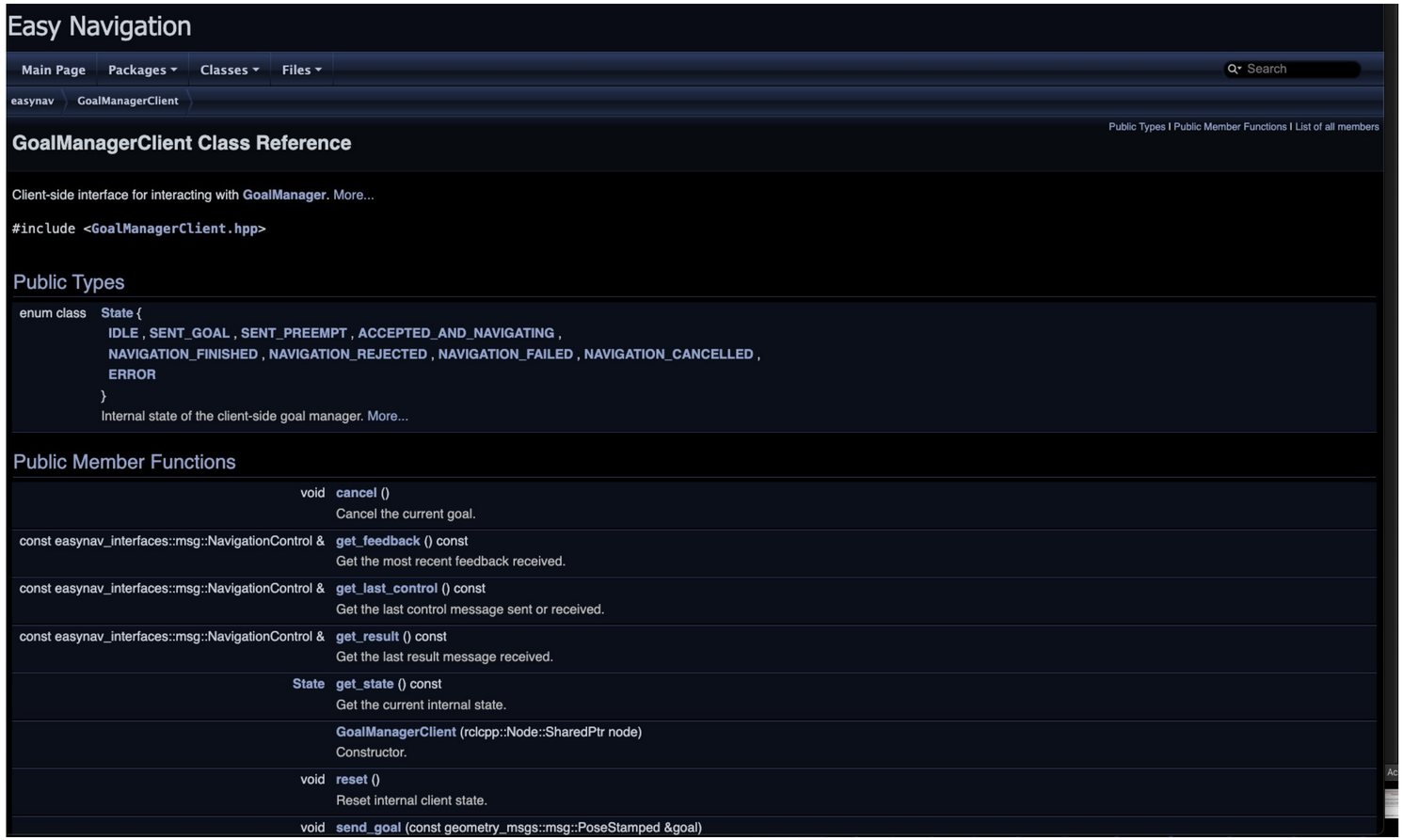


Repository	Doxygen	Rolling	Kilted	Jazzy	Humble
EasyNavigation	Doxygen Deployment passing	rolling passing	kilted passing	jazzy passing	humble passing
NavMap	Doxygen Deployment passing	rolling passing	kilted passing	jazzy passing	humble passing
easynav_plugins	Doxygen Deployment passing	rolling passing	kilted passing	jazzy passing	humble passing
yaets	Doxygen Deployment passing	rolling passing	kilted passing	jazzy passing	humble passing

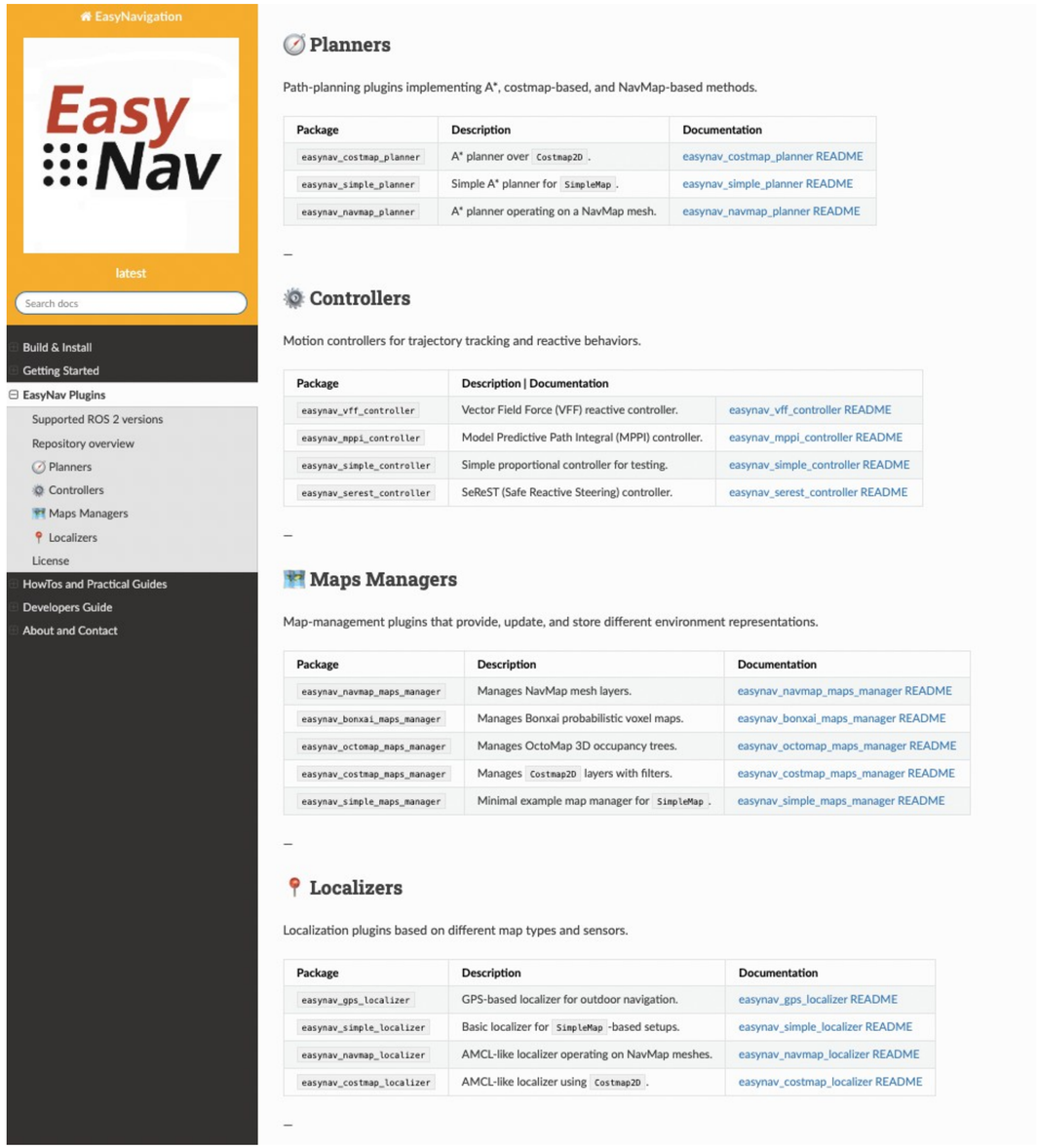
EasyNav
Documentation

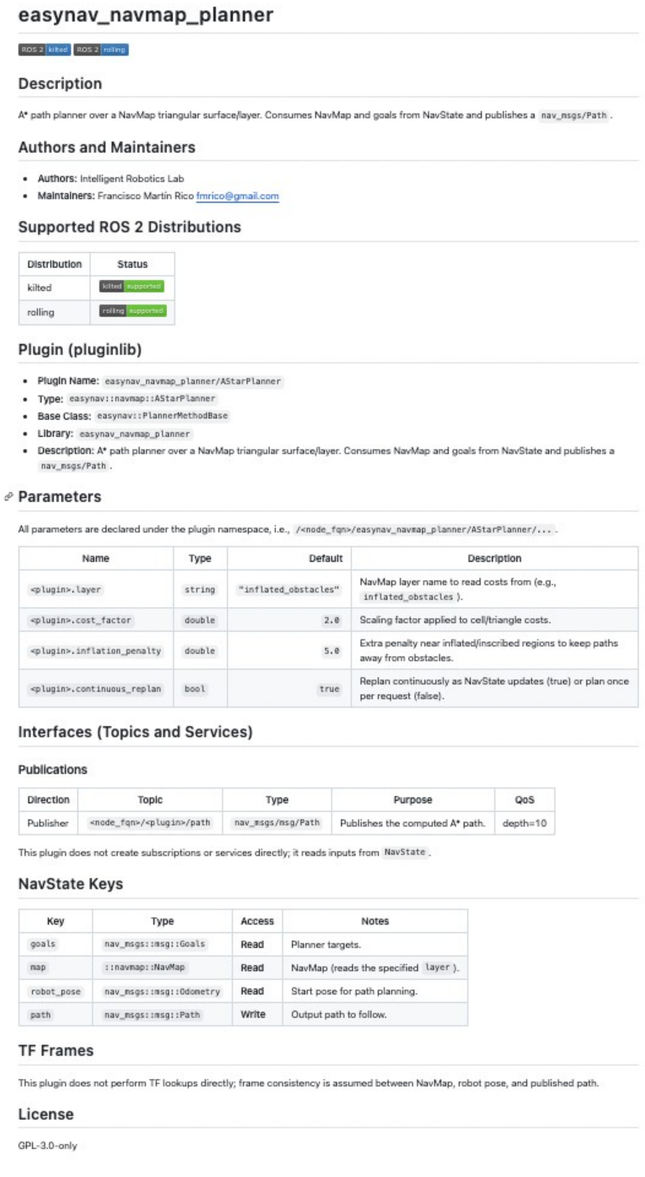


https://easynavigation.github.io



EasyNav
Documentation





Conclusions

Where are we?
Where are we going?

Conclusions

- We have developed a navigation system which is robust, easy to use and well designed
- It is light and determinist
- It adapts to problems in which using Nav2 requires hacks
- Many plugins and navigation stacks already operative: Costmaps, gridmaps, Bonxai, NavMap...
- We want to build community around it
- Still lacking robust and effective controllers

Easy Nav

An open-source framework for navigating everywhere

Intelligent Robotics Lab

¿Questions?

Prof. Dr. Francisco Martín Rico
francisco.rico@urjc.es

Dr. Francisco Miguel Moreno
franciscom.moreno@urjc.es

FOSDEM

Universidad Rey Juan Carlos