ROS2

for

ROS Developers

ROS

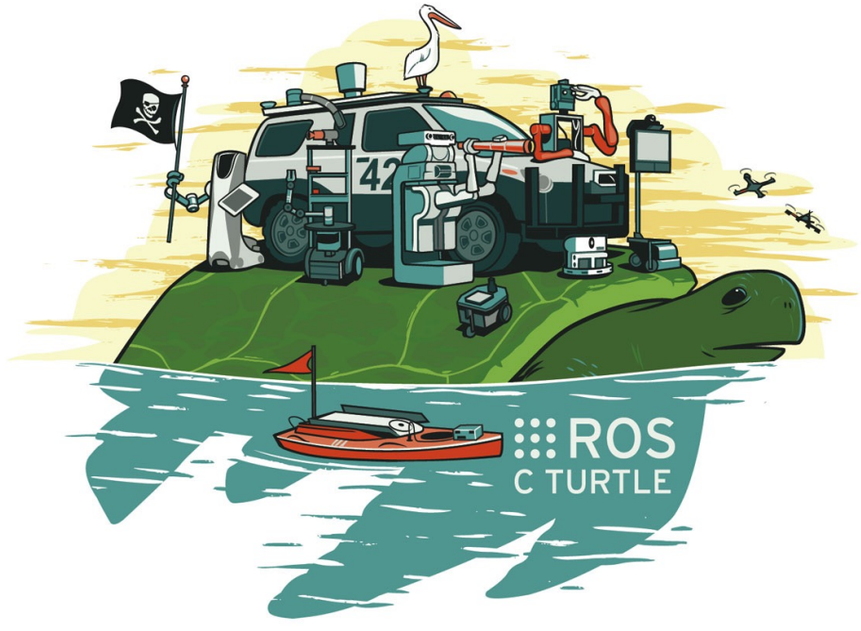
Intelligent Robotics Lab

Prof. Dr. Francisco Martín Rico

Madrid 30-1-2020

Universidad Rey Juan Carlos

Introduction



Definition

ROS (Robot Operating System)

"The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source."

http://www.ros.org/

Open Source Robotics Foundation

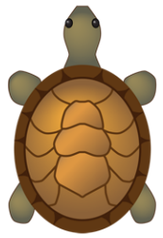
History

- ROS was born in 2006 at Stanford, within STAIR project
- Willow Garage in 2007
- Open Source Robotics Foundation in 2013
- Since 2013 is considered the standard in Robotics

Willow Garage

Open Source Robotics Foundation

ROS Distributions

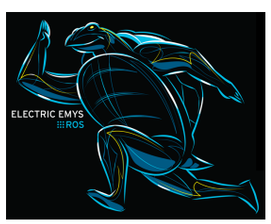


Box Turtle

Mar 2, 2010

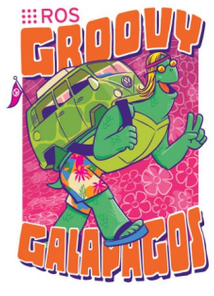
Aug 2, 2010

Mar 2, 2011



Aug 30, 2011

Apr 23, 2012



Dec 31, 2012

Sept 4th, 2013

Jul 22nd, 2014

May 23rd, 2015

May 23rd, 2016

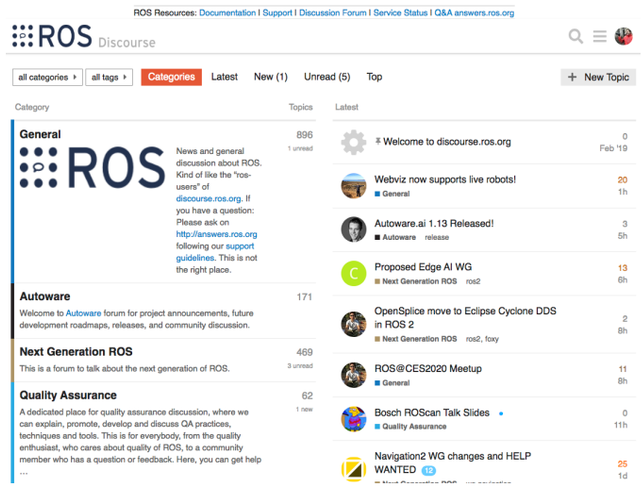
May 23rd, 2017

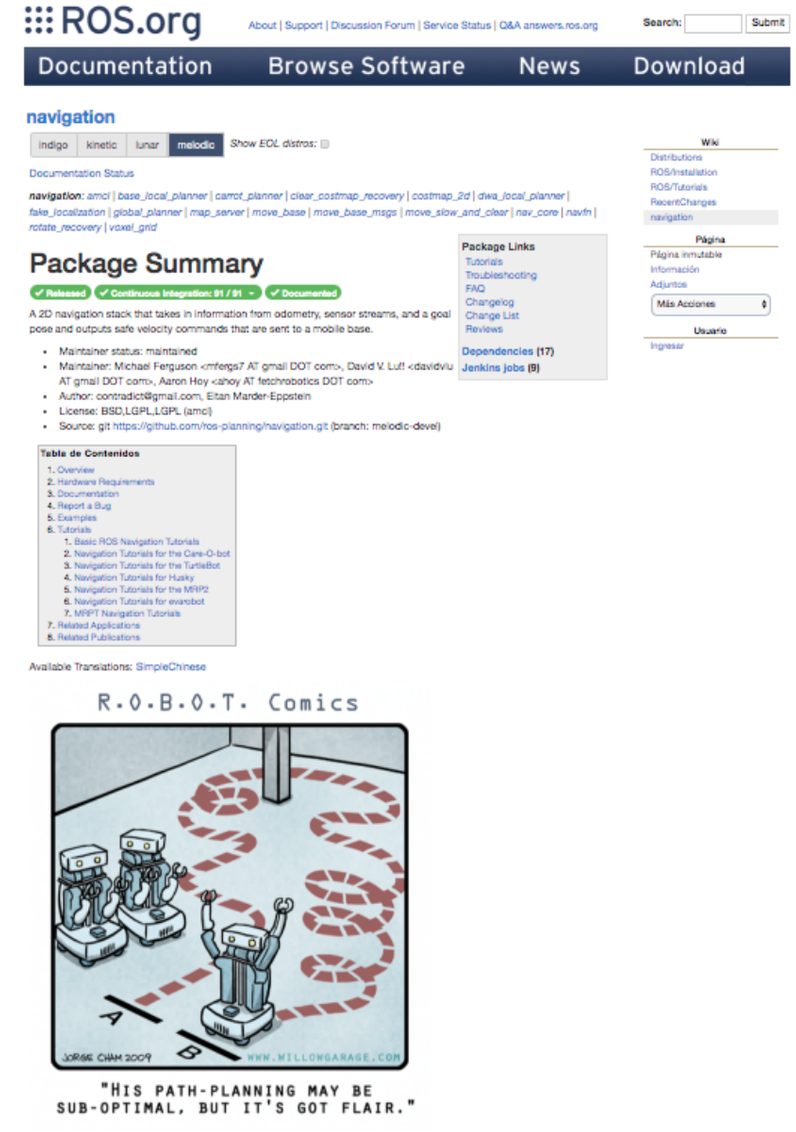
May 23rd, 2018

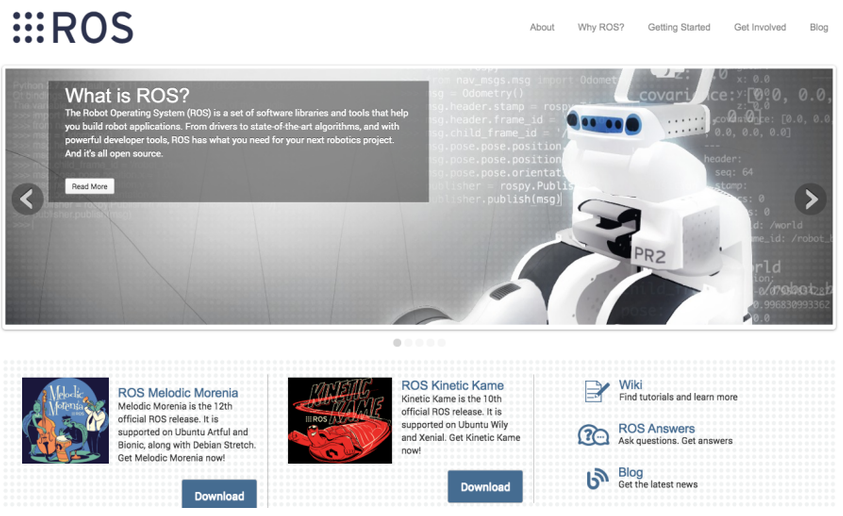
Intelligent Robotics Lab

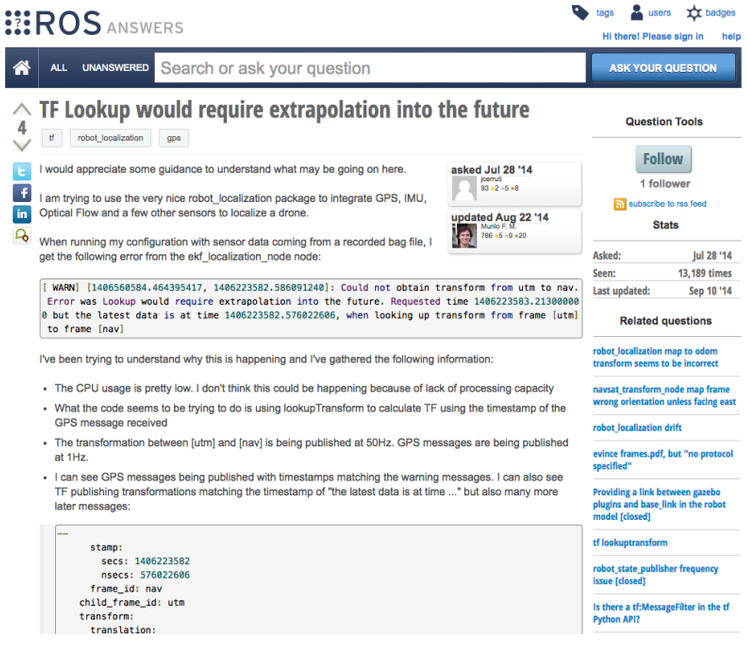
Resources

- ROS.org [link]
- Distributions [link]
- Package documentation [link][link]
- ROS Wiki [link]
- ROS Answers [link]
- ROS Discurse [link]
- Blog [link]









Intelligent Robotics Lab

Limitations

- Centralized in Master
- Own networking implementation
  - No real multicast
  - No QoS
- No multirobot
- No Real Time
- No multiplatform
- Important components are patches
- Only C++ and Python independent implementations
- Security

Intelligent Robotics Lab

ROS2 Distributions

ROS2 Distributions

Dec 8th, 2017

July 2nd, 2018

Dec 14th, 2018

May 31st, 2019

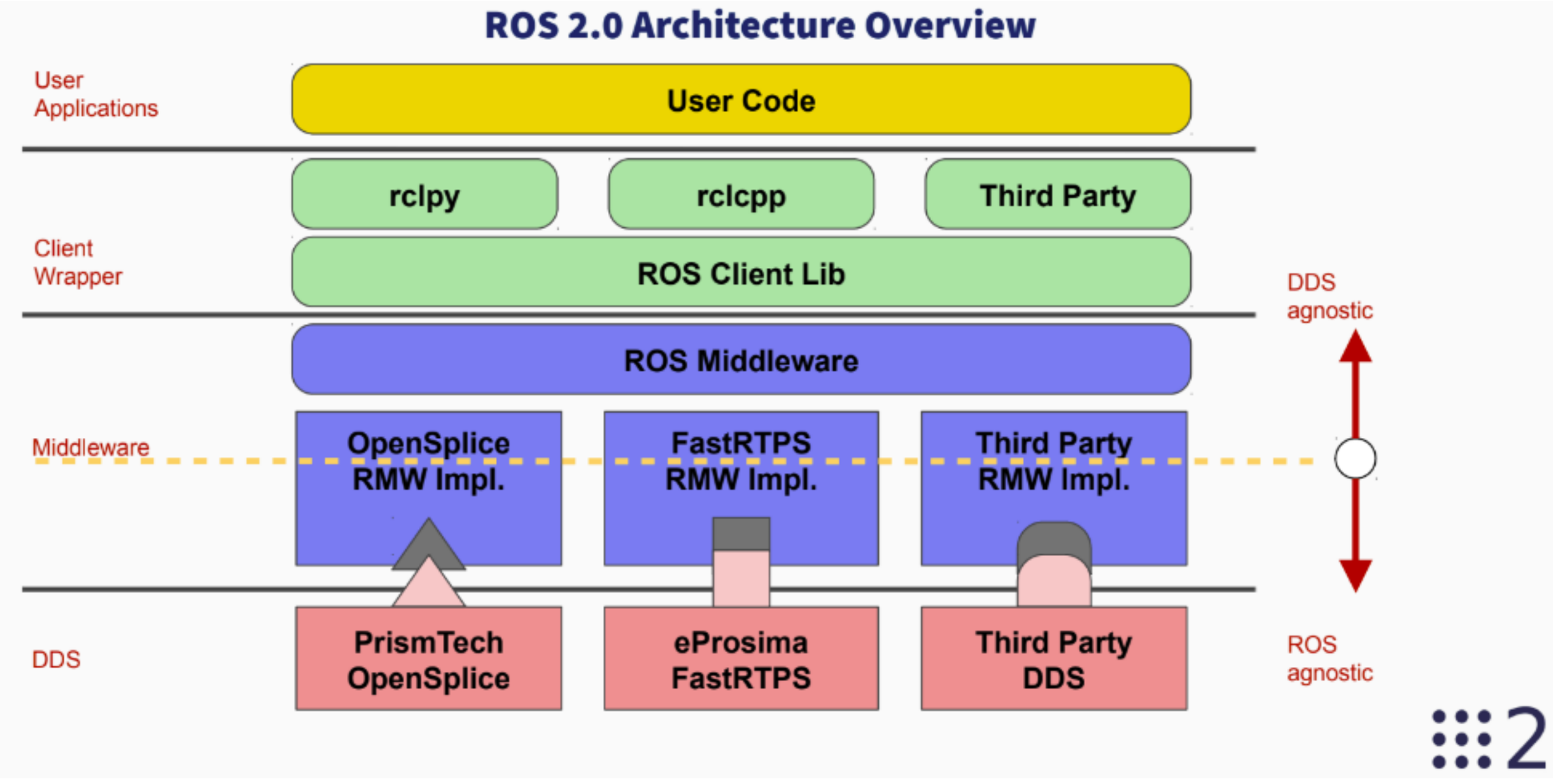
Nov 22nd, 2019

Intelligent Robotics Lab

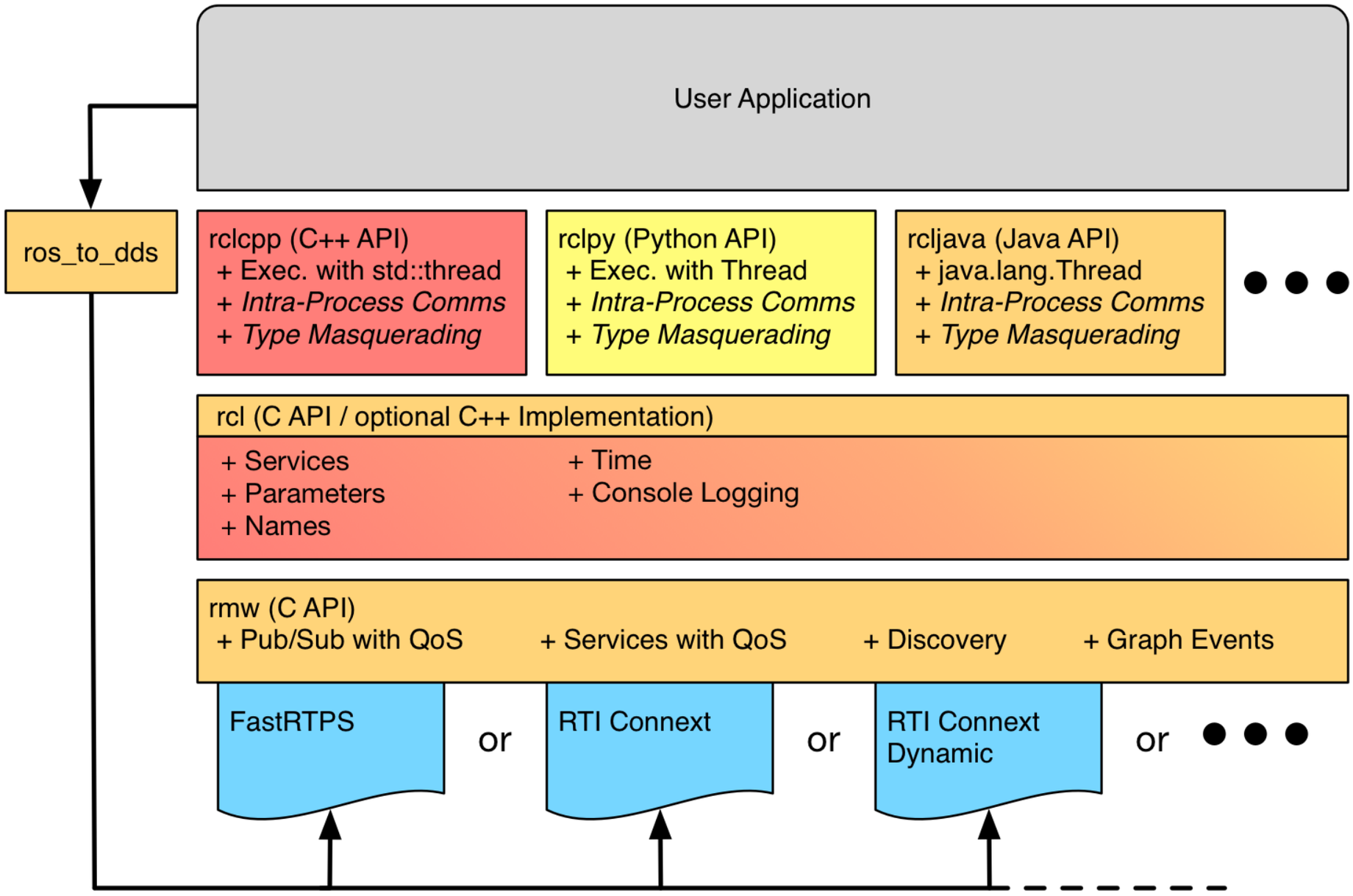
Main Design
ROS2

Intelligent Robotics Lab

Design


ROS 2.0 Architecture Overview
User Applications
User Code
rclpy
rclcpp
Third Party
Client Wrapper
ROS Client Lib
DDS agnostic
ROS Middleware
Middleware
OpenSplice RMW Impl.
FastRTPS RMW Impl.
Third Party RMW Impl.
DDS
PrismTech OpenSplice
eProsima FastRTPS
Third Party DDS
ROS agnostic

Design



* *Intra-Process Comms* and *Type Masquerading* could be implemented in the client library, but may not currently exist.

Hands on!!
ROS2

Intelligent Robotics Lab

ROS2 Setup

- Install ROS2

https://index.ros.org/doc/ros2/Installation/

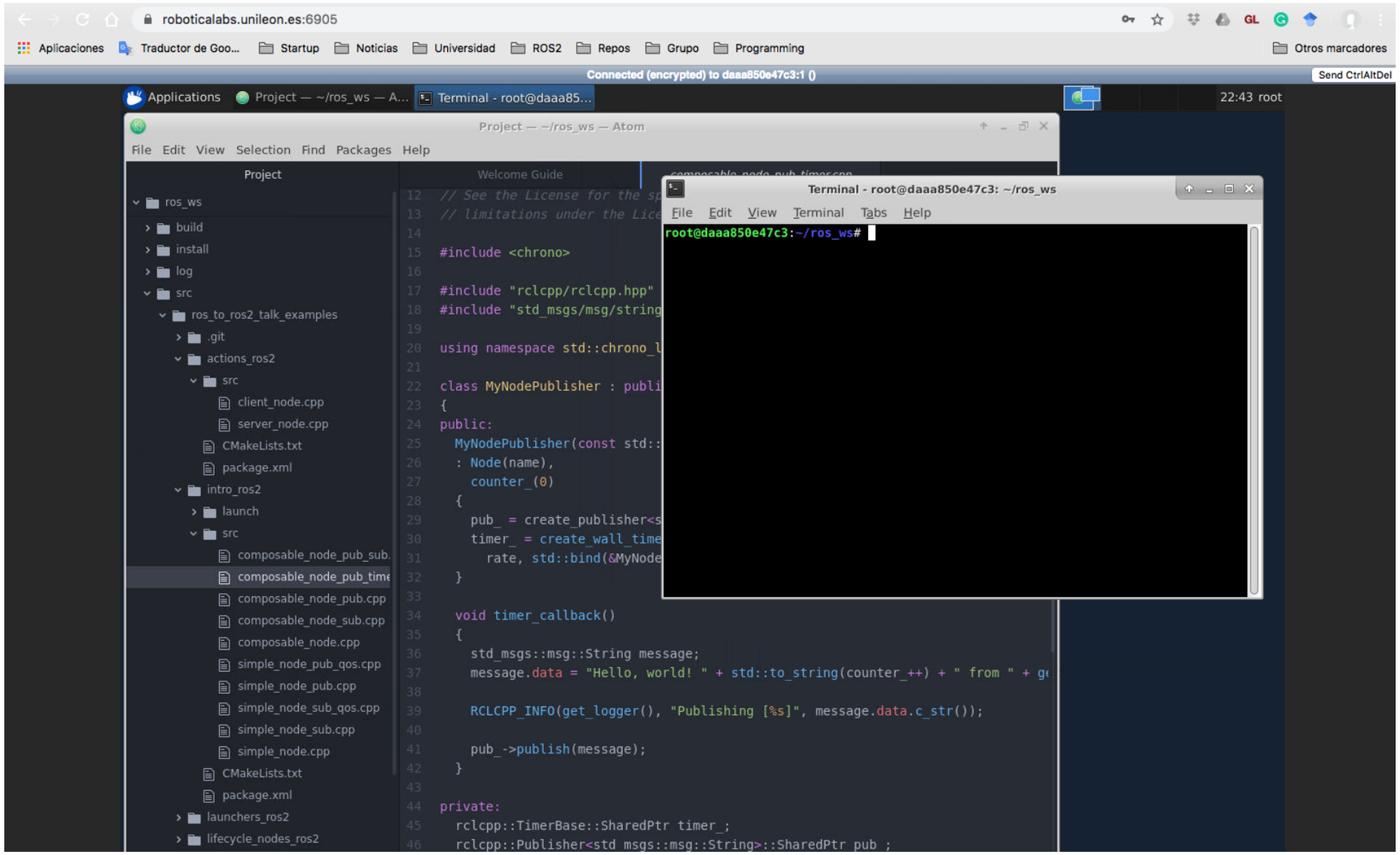
- Setup ROS2

```
source /opt/ros/eloquent/setup.bash
source /home/fmrico/ros/ros2/ros2talk_ws/install/setup.bash
export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
```

Virtual machine with ROS2

GRUPO DE ROBÓTICA

robotica.unileon.es



https://roboticalabs.unileon.es:N/

Intelligent Robotics Lab

Firsts steps ROS2

- Your ROS2 workspace
- `colcon` as build tool
- `build, install, log, ~~devel~~`

rclcpp API

- Extensive use of advanced C++ (C++17 mostly)
- `rclcpp::`
- Nodes are no longer processes; they are objects
- No need for `NodeHandle`
- `ros2` as super command

intro_ros2/src/simple_node.cpp

```
$ colcon build ——symlink-install
$ ros2 run intro_ros2 nodo_simple
$ ros2 node list
$ ros2 node info /simple_node
```

Interfaces, pub/sub

intro_ros2/src/simple_node_pub.cpp

- Messages includes change: lowercase and _
- Differentiation between msg, srv and action

```
$ ros2 msg list
$ ros2 msg std_msgs/msg/String
```

← Deprecated

```
$ ros2 interface list -m
$ ros2 interface show std_msgs/msg/String
```

- publishers and subscriptions are requested to nodes
- `ROS_INFO -> RCLCPP_INFO`
- Logger associated to a node id
- `spin_some` and `spin_once` (and more spinners)

```
$ ros2 run intro_ros2 simple_node_pub
$ ros2 topic echo /chatter
$ ros2 topic info /chatter
$ ros2 topic hz /chatter
$ ros2 topic bw /chatter
```

intro_ros2/src/simple_node_sub.cpp

QoS in ROS2

intro_ros2/src/simple_node_pub_qos.cpp
intro_ros2/src/simple_node_sub_qos.cpp

```
auto subscription = node->create_subscription<std_msgs::msg::String>(
  "chatter", rclcpp::QoS(100).transient_local(), callback);
```

```
auto publisher = node->create_publisher<std_msgs::msg::String>(
  "chatter", rclcpp::QoS(100).transient_local());
```

rclcpp QoS API

QoS policies

History	Keep last	Keep all
Depth	Size	
Reliability	Best effort	Reliable
Durability	Transient local	Volatile

QoS profiles

Default	Reliable	Volatile	Keep last
Services	Reliable	Volatile	normal queue
Sensor	Best effort	small queue	
Parameters	Reliable	Volatile	Larger queue

Intelligent Robotics Lab

QoS in ROS2

intro_ros2/src/simple_node_pub_qos.cpp
intro_ros2/src/simple_node_sub_qos.cpp

```
auto subscription = node->create_subscription<std_msgs::msg::String>(
  "chatter", rclcpp::QoS(100).transient_local(), callback);
```

```
auto publisher = node->create_publisher<std_msgs::msg::String>(
  "chatter", rclcpp::QoS(100).transient_local());
```

rclcpp QoS API

Compatibility of QoS **durability** profiles:		Subscriber	
		Volatile	**Transient Local**
Publisher	**Volatile**	Volatile	No Connection
	Transient Local	Volatile	Transient Local

Compatibility of QoS **reliability** profiles:		Subscriber	
		Best effort	**Reliable**
Publisher	**Best effort**	Best effort	No Connection
	Reliable	Best effort	Reliable

Intelligent Robotics Lab

Better design for nodes

intro_ros2/src/composable_node.cpp

- Inherit from `rclcpp::Node`

intro_ros2/src/composable_node_pub.cpp

- Publishers/subscribers inside nodes

intro_ros2/src/composable_node_sub.cpp

- Multiples nodes per process
- **Executors**
  - `rclcpp::executors::SingleThreadedExecutor` API
  - `rclcpp::executors::MultiThreadedExecutor` API

More executions models

intro_ros2/src/composable_node_pub_sub.cpp

- `spin_until_future_complete`

intro_ros2/src/composable_node_pub_timer.cpp

- Timers for controlling rhythm
  - No extra threads - Graceful degradation

Parameters

- Parameters are stored in nodes
- They must be declared

parameters_ros2/src/simple_param_node.cpp

```
$ ros2 run parameters_ros2 simple_param_node
$ ros2 param get /simple_param_node message
```

- You can subscribe to changes

parameters_ros2/src/reconf_param_node.cpp

```
$ ros2 run parameters_ros2 reconf_param_node
$ ros2 param set /ReconfigurableNode speed 5.6
```

Launchers

- **In Python**
- More control

intro_ros2/launch/composable_pub_timer_launch.py

- `generate_launch_description` function that returns a `LaunchDescription` object

```
$ ros2 launch intro_ros2 composable_pub_timer_launch.py
```

Launchers

- It contains actions: execute node, set env var, include other launcher and launcher args

launchers_ros2/launch/first_launch.py

launchers_ros2/launch/second_launch.py

- Set parameters from files

parameters_ros2/launch/param_file_launch.py

parameters_ros2/config/config_2.yaml

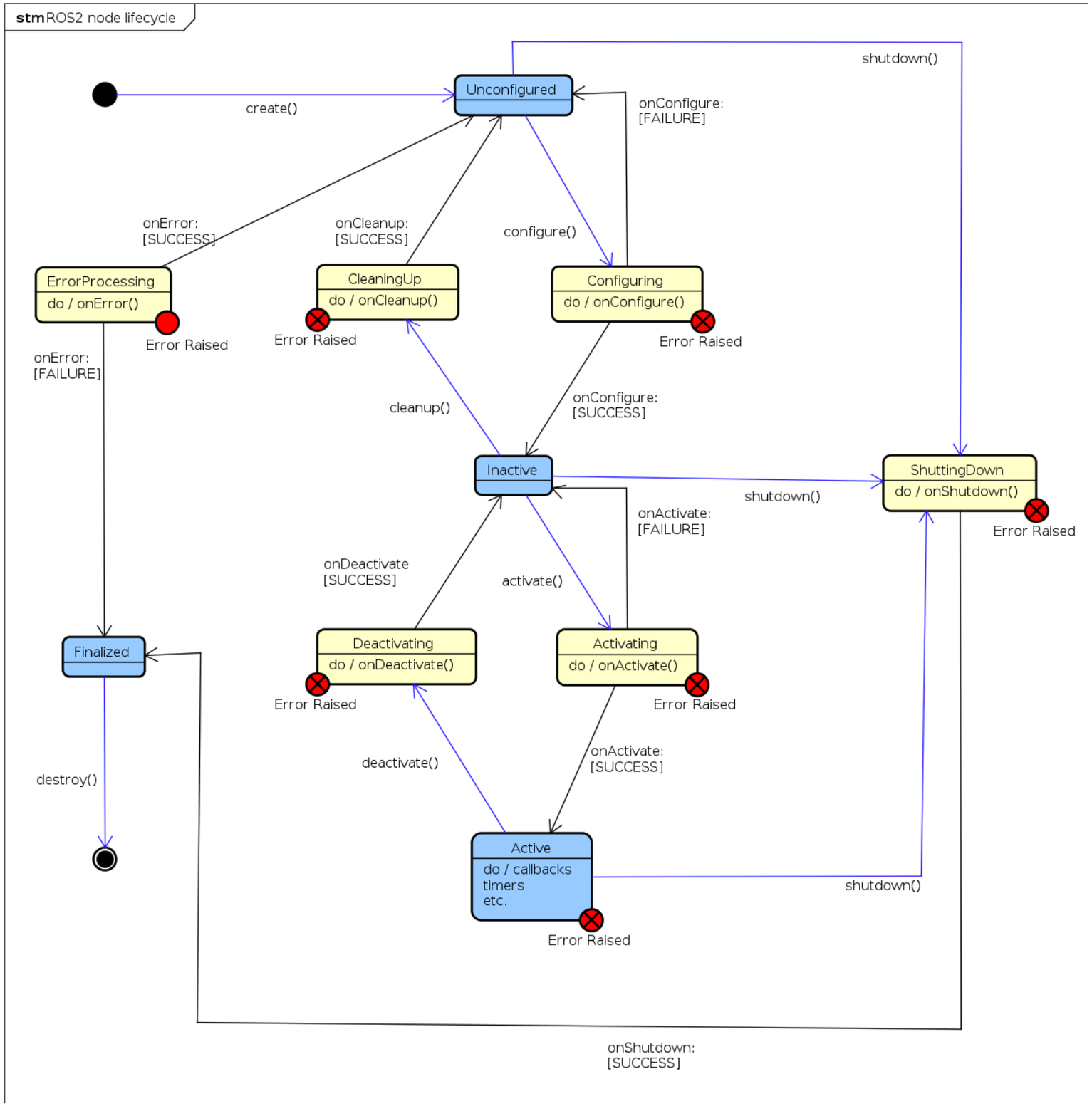
Lifecycle nodes

Intelligent Robotics Lab

Lifecycle nodes

- In ROS, nodes are executing from initial
- Some nodes open devices
- Some nodes read params and configurations
- Specially critical for drivers
- In a launcher, there is not a way to control the boot order
- No way to pause, reconfigure, cleanup and shutdown nodes
- **ROS2: Lifecycle Nodes**
- LF nodes are in a state
- In each state, there are some available transitions
- Observed and commanded from outside (from launchers!!!)

States and Transitions



Lifecycle node structure

lifecycle_nodes_ros2/src/simple_lifecycle_node.cpp

- Inheritance from `rclcpp_lifecycle::LifecycleNode`
- Control of code in transitions
  - `on_configue`
  - `on_activate`
  - …
- You can accept or reject a transition request

```
$ ros2 launch lifecycle_nodes_ros2 simple_lifecycle_node
$ ros2 lifecycle nodes
$ ros2 lifecycle get /lifecycle_node_example
$ ros2 lifecycle list /lifecycle_node_example
$ ros2 lifecycle set /lifecycle_node_example configure
$ ros2 lifecycle set /lifecycle_node_example activate
$ ros2 lifecycle set /lifecycle_node_example deactivate
```

- Lifecycle publishers

lifecycle_nodes_ros2/src/pub_lifecycle_node.cpp

Services and Actions

Services

services_ros2/src/server_node.cpp

- Very similar to ROS

```
$ ros2 run services_ros2 server_node
$ ros2 service list
$ ros2 service type /reverse_string
$ ros2 interface show ros2talk_msgs/srv/ReverseString
$ ros2 service call /reverse_string ros2talk_msgs/srv/ReverseString
'{normal_sentence: "Hello World ROS2!!!!"}'
```

services_ros2/src/simple_client_node.cpp

- A little bit more complicated
- `spin_until_future_complete`

```
$ ros2 run services_ros2 server_node
$ ros2 run services_ros2 simple_client_node
```

Intelligent Robotics Lab

Actions

actions_ros2/src/server_node.cpp

- Some callbacks
  - `handle_goal`
  - `handle_cancel`
  - `handle_accepted`
- Use of thread not mandatory
- Sending back feedback and result using `goal_handle`

```
$ ros2 run actions_ros2 server_node
$ ros2 action list
$ ros2 action info /repeat_sentence
```

actions_ros2/src/client_node.cpp

- Callbacks `feedback_callback result_callback`

```
$ ros2 run actions_ros2 server_node
$ ros2 run actions_ros2 client_node
```

Intelligent Robotics Lab

Software in ROS2

Intelligent Robotics Lab

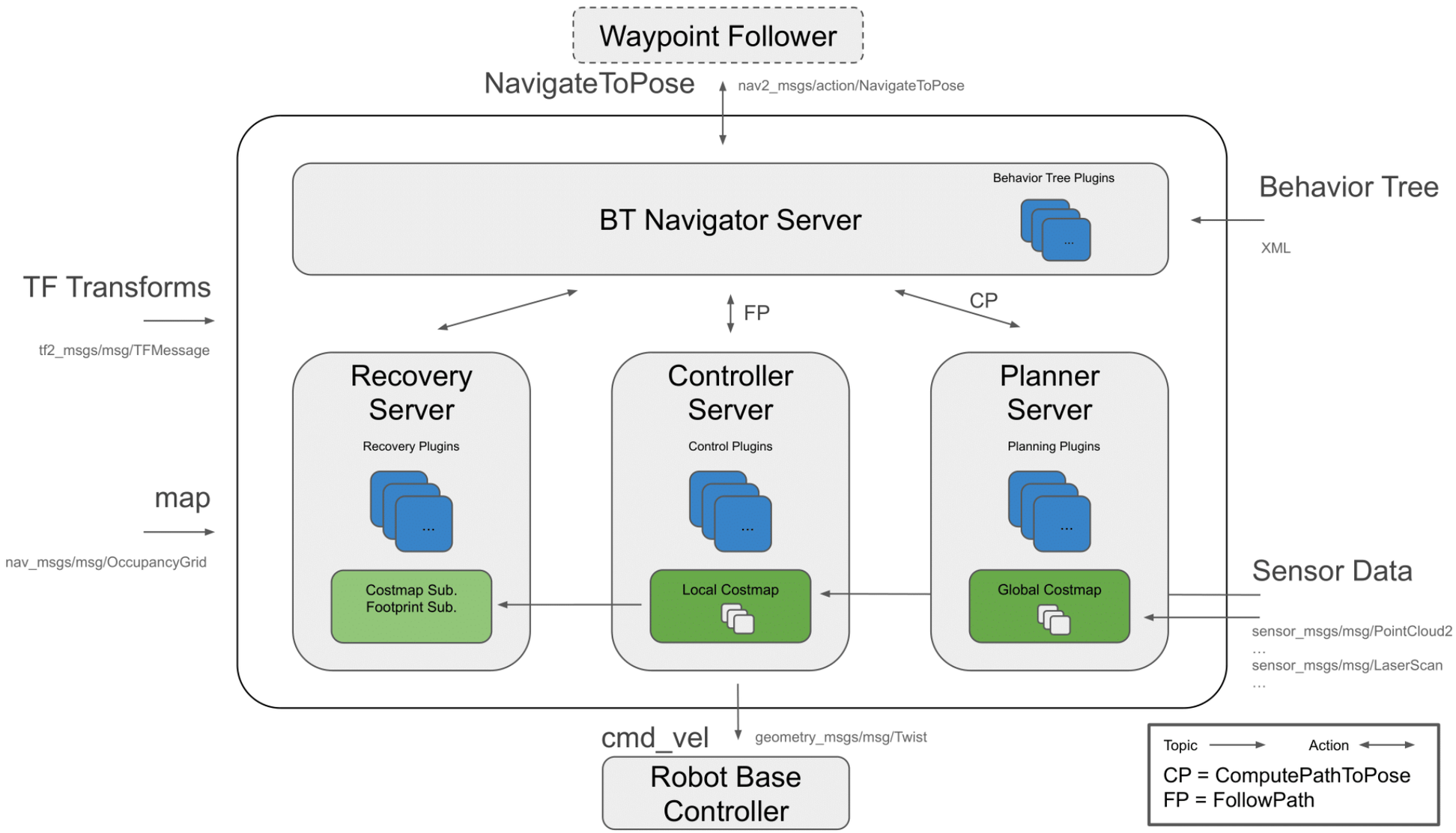
Software growing up!!

- Growing last months
- Principal packages ported to ROS2
  - ImageTransport
  - TF2
  - Robots: Kobuki, TB3….
  - Drivers: Cameras, Lasers..
  - Behavior Trees
  - Navigation2
  - PlanSys2
  - …

Intelligent Robotics Lab

Navigation2

https://github.com/ros-planning/navigation2



Intelligent Robotics Lab

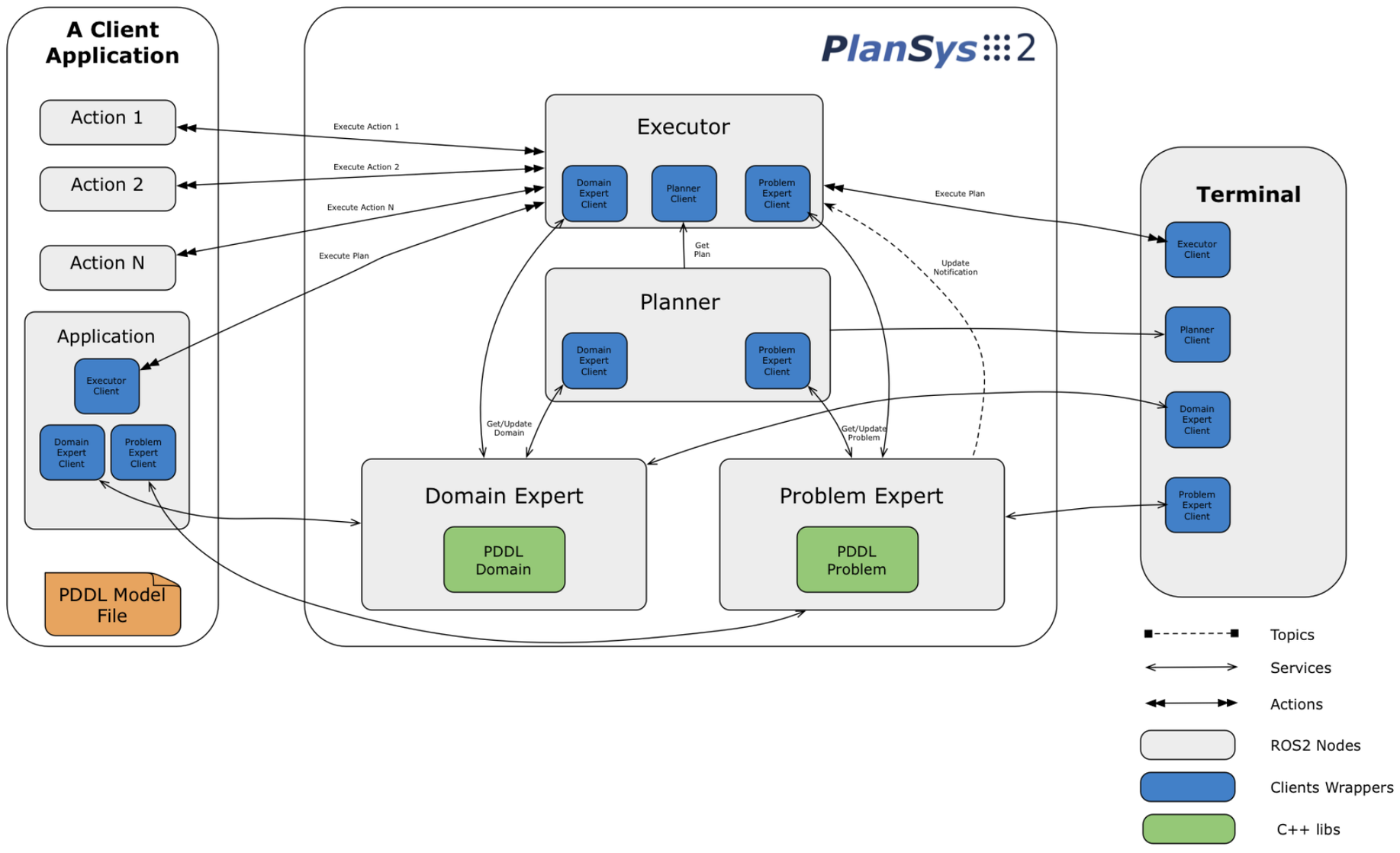
Navigation2

https://github.com/ros-planning/navigation2

Intelligent Robotics Lab

Intelligent Robotics Lab

https://github.com/IntelligentRoboticsLabs/ros2_planning_system



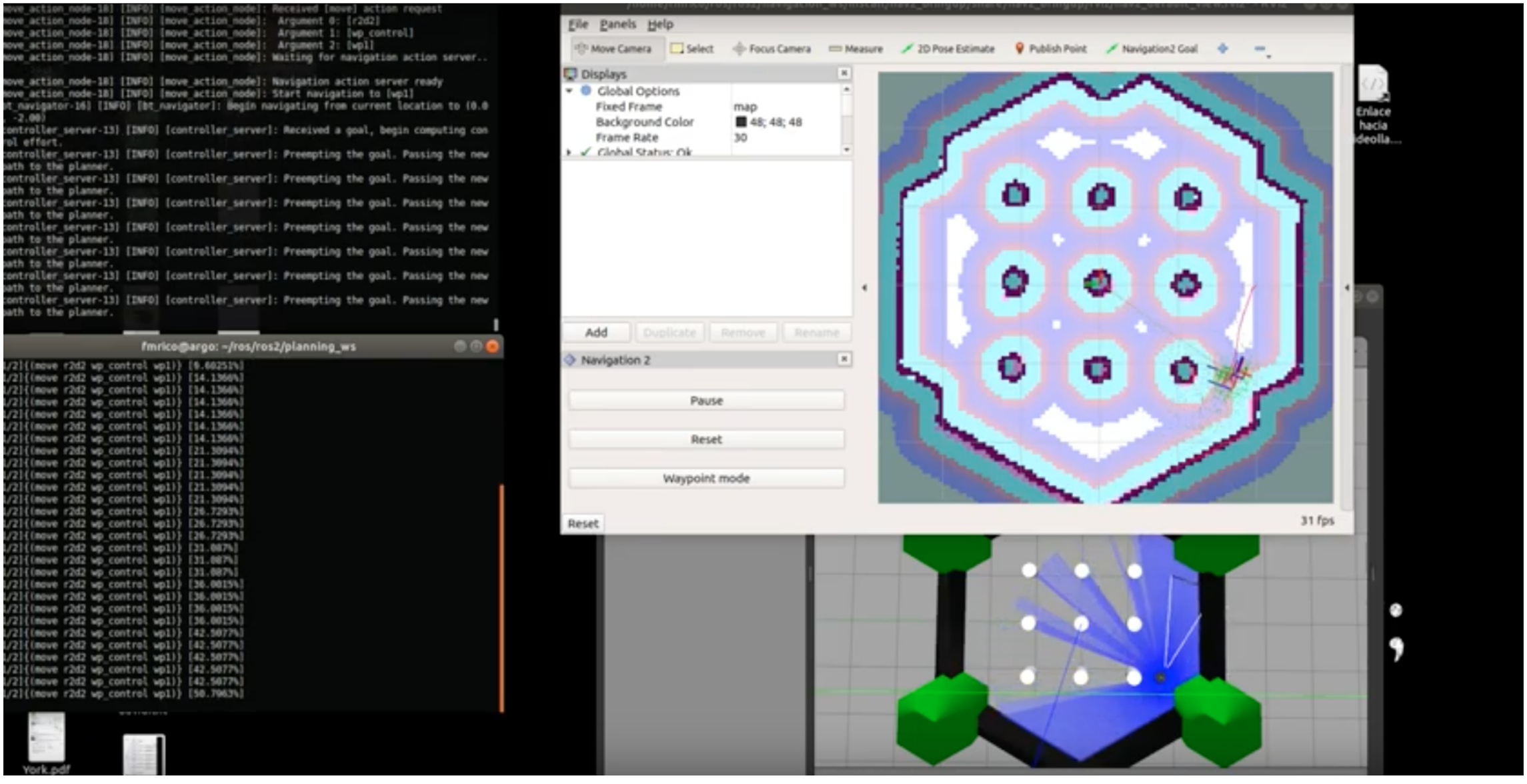
2 Planning System

Intelligent Robotics Lab

PlanSys2

ROS 2

https://github.com/IntelligentRoboticsLabs/ros2_planning_system



2 *Planning System*

Intelligent Robotics Lab

ROS2

for

ROS Developers

Muchas gracias!!!

Encuesta

Prof. Dr. Francisco Martín Rico
Madrid 30-1-2020

Intelligent Robotics Lab

Universidad Rey Juan Carlos