ROS2

for

ROS Developers

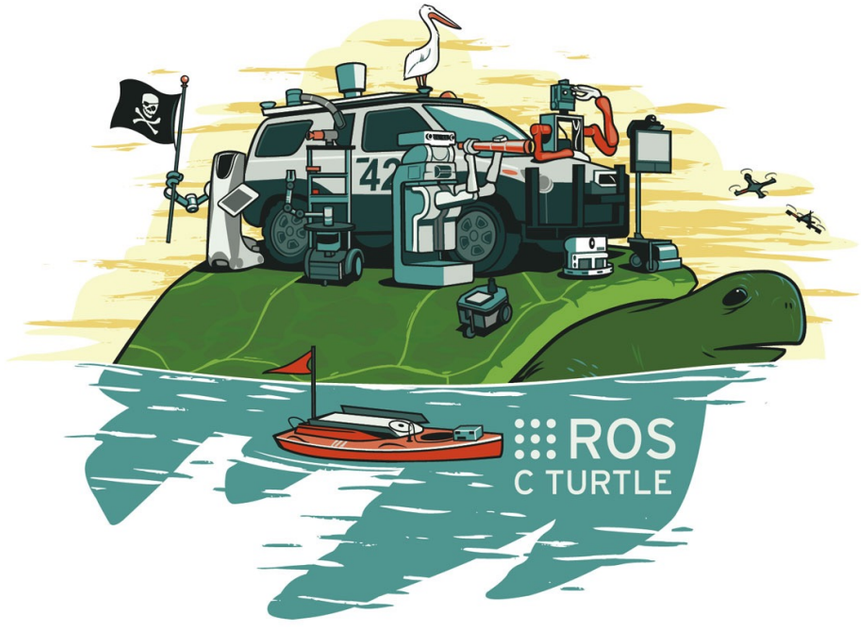
ROS 2

Prof. Dr. Francisco Martín Rico **francisco.rico@urjc.es**
Madrid 2-3-2022 **@fmrico**

Intelligent Robotics Lab

Universidad Rey Juan Carlos

Introduction



Definition

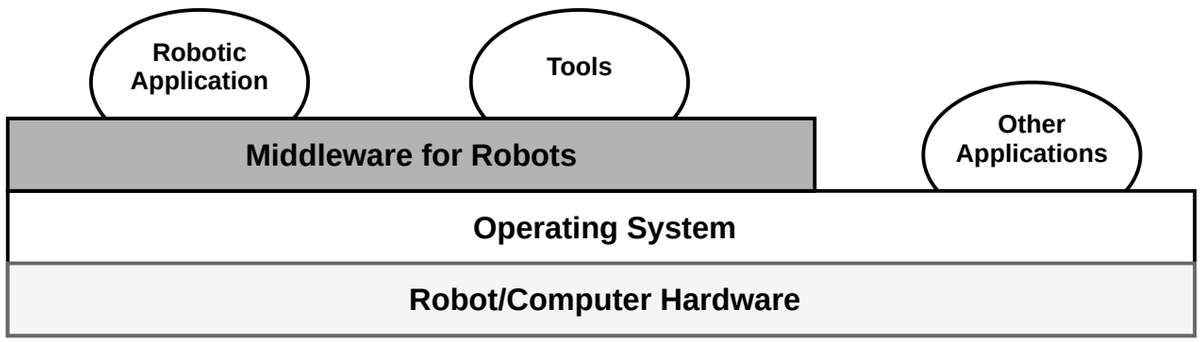
ROS (Robot Operating System)

"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/

ROS

Open Source Robotics Foundation



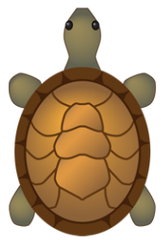
History

- 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

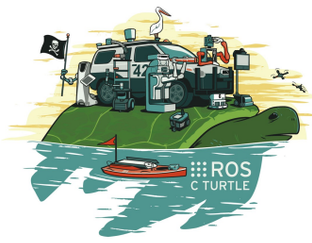
Open Source Robotics Foundation

ROS Distributions



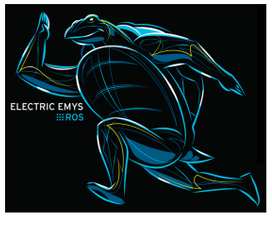
Box Turtle

Mar 2, 2010

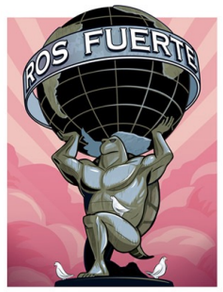


Aug 2, 2010

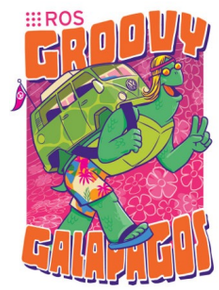
Mar 2, 2011



Aug 30, 2011



Apr 23, 2012



Dec 31, 2012

Sept 4th, 2013

Jul 22nd, 2014

May 23rd, 2015

May 23rd, 2016

May 23rd, 2017

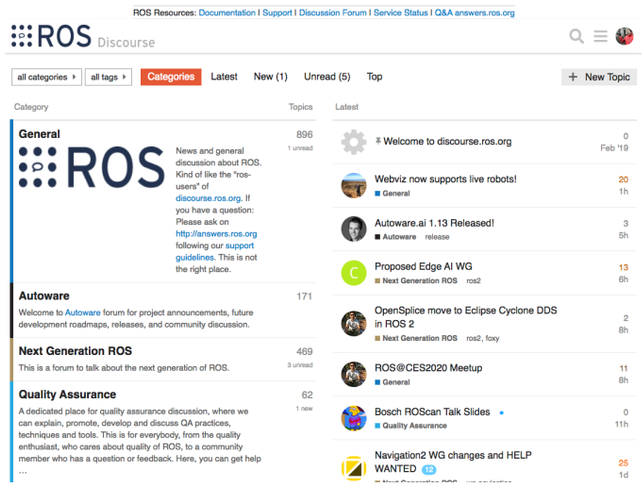
May 23rd, 2018

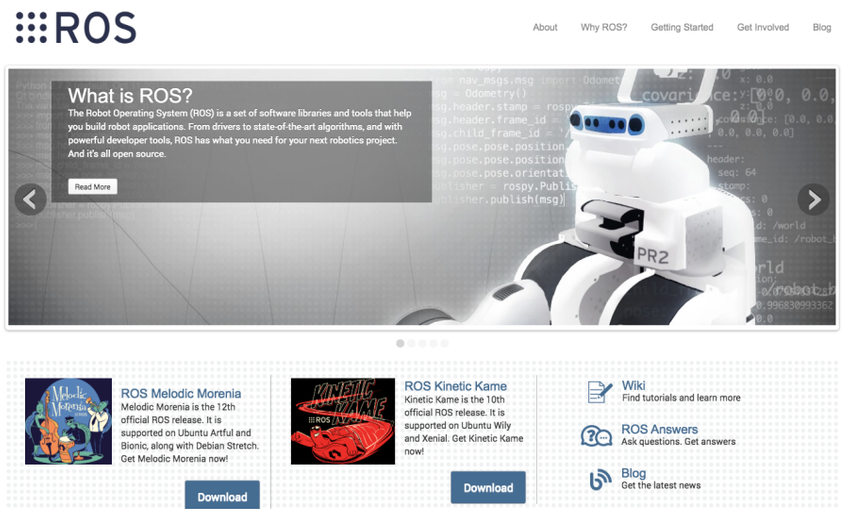
May 23rd, 2020

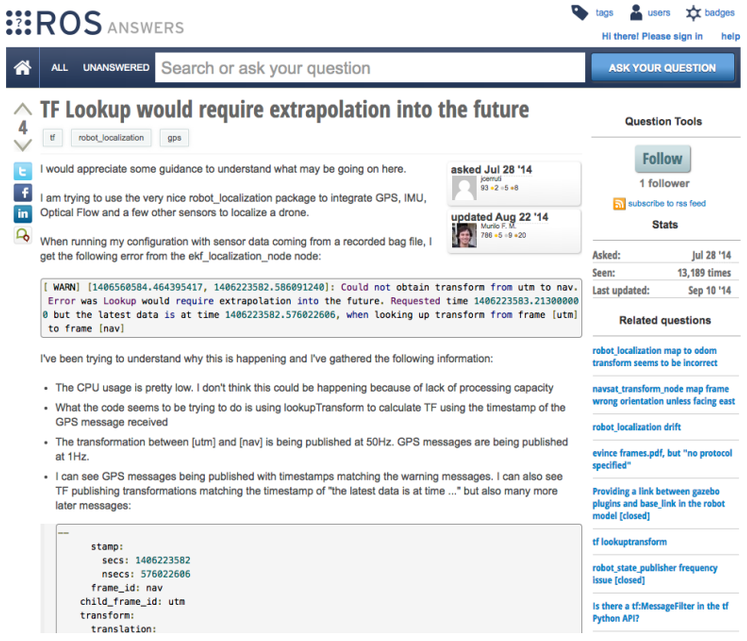
Intelligent Robotics Lab

Resources

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







Intelligent Robotics Lab

Resources

ROS COMMANDS CHEAT SHEET v1.0

ROS.org

MASTER
Run a Master
$ roscore

NODES
Get list of nodes
$ rosnode list

Get info of a node
Given a node name, it return its subscriptions, publications and services
$ rosnode info [node]

Terminate a node
$ rosnode kill [node]

Test if a node is alive
Get latency and if node is alive given its name
$ rosnode ping [node]

TOPICS
Get list of topics
$ rostopic list

Get info of topics
Get publishers and subscribers, as well as the message type
$ rostopic info [topic]

Manual publish
Manually publication to a topic
$ rostopic pub [-r freq] [topic] [msg_type] [msg]

Visualize topic publications
$ rostopic echo [topic]

MESSAGES
Get list of available messages
$ rosmsg list

Show message structure
$ rosmsg show [msg_type]

SERVICES
Get list of available service types
$ rossrv list

Show service type structure
$ rossrv show [srv_type]

Show available running services
$ rosservice list

Show service type of a running service
$ rosservice info [srv]

Call a running service
$ rosservice call [srv] [srv_type] [request]

FILESYSTEM
change working package dir
Change to active workspace dir
$ roscd
Change to a dir starting from a package
$ roscd [package]
List package content
$ rosls [package]

Running a node
Requires a running Master
$ rosrun [package] [node]

Launch an application
$ roslaunch [package] [launcher]

CONFIGURE DISTRIBUTED SYSTEM
***Host**: machine that contains the master*
***Remote**: machine that connects to Host*
Both machines have to ping each other by name

In Remote:
$ export ROS_IP=[REMOTE_IP]
$ export ROS_HOSTNAME=[REMOTE_HOSTNAME]
$ export ROS_MASTER_URI=http://[HOST_HOSTNAME]:11311

Check for problems (even in package):
$ roswtf

DEBUGGING
Show nodes and topics connections
$ rosrun rqt_graph rqt_graph

Show TF transforms tree
$ rosrun rqt_tf_tree rqt_tf_tree

Show TF frames info
$ rosrun tf tf_monitor

Show TF transformation
$ rosrun tf tf_echo [frame_src] [frame_dst]

COMPILATION
Compile a workspace
Compile all the packages in a workspace with num_threads threads
$ catkin_make [-j num_threads]
Compile a specified pkg
$ catkin_make [-j num_threads] [--pkg package]

Intelligent Robotics Lab

www.inrobots.es
www.intelligentroboticslab.robotica.gsyc.es

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

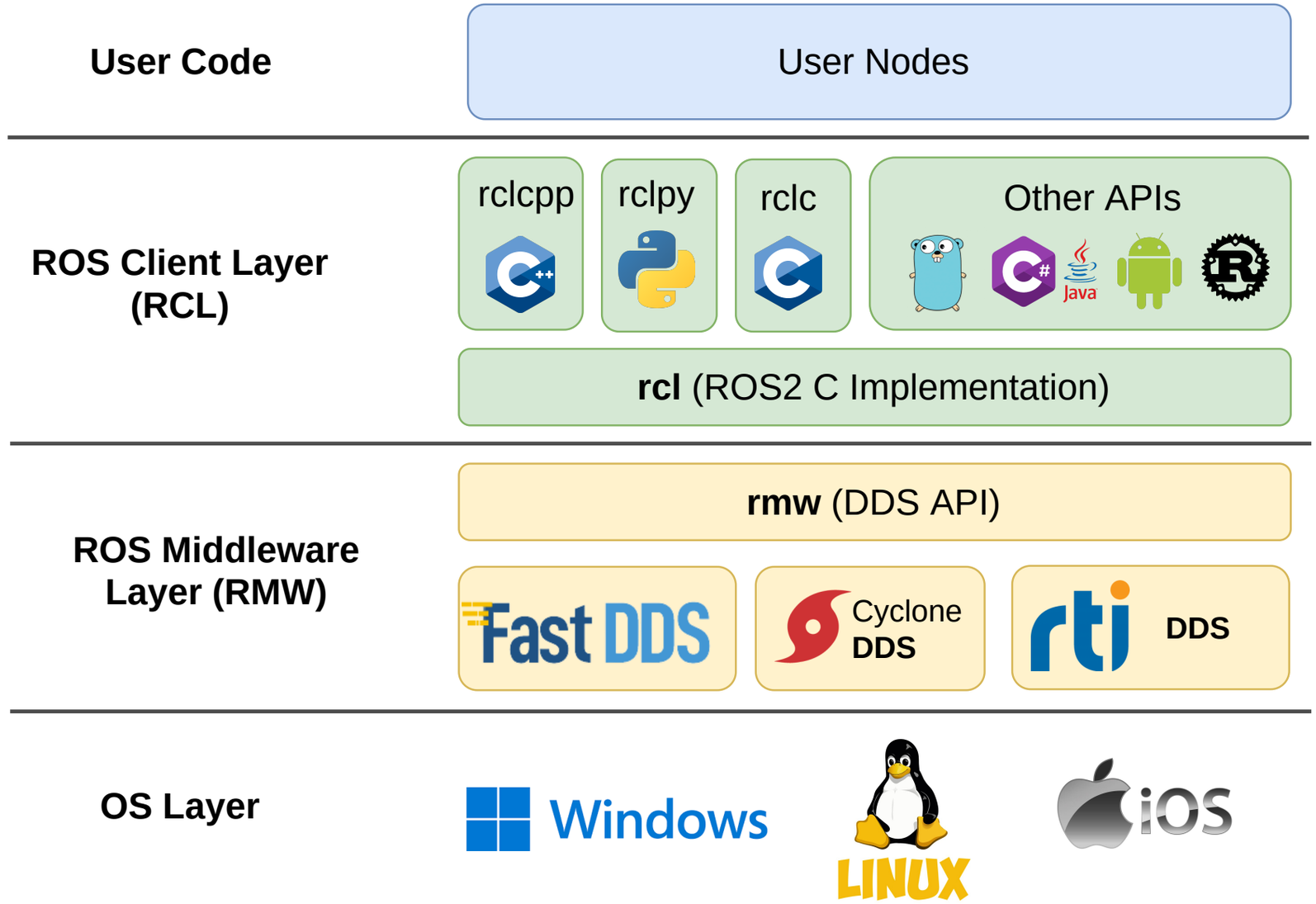
Nov 22nd, 2019

Jun 5th, 2020

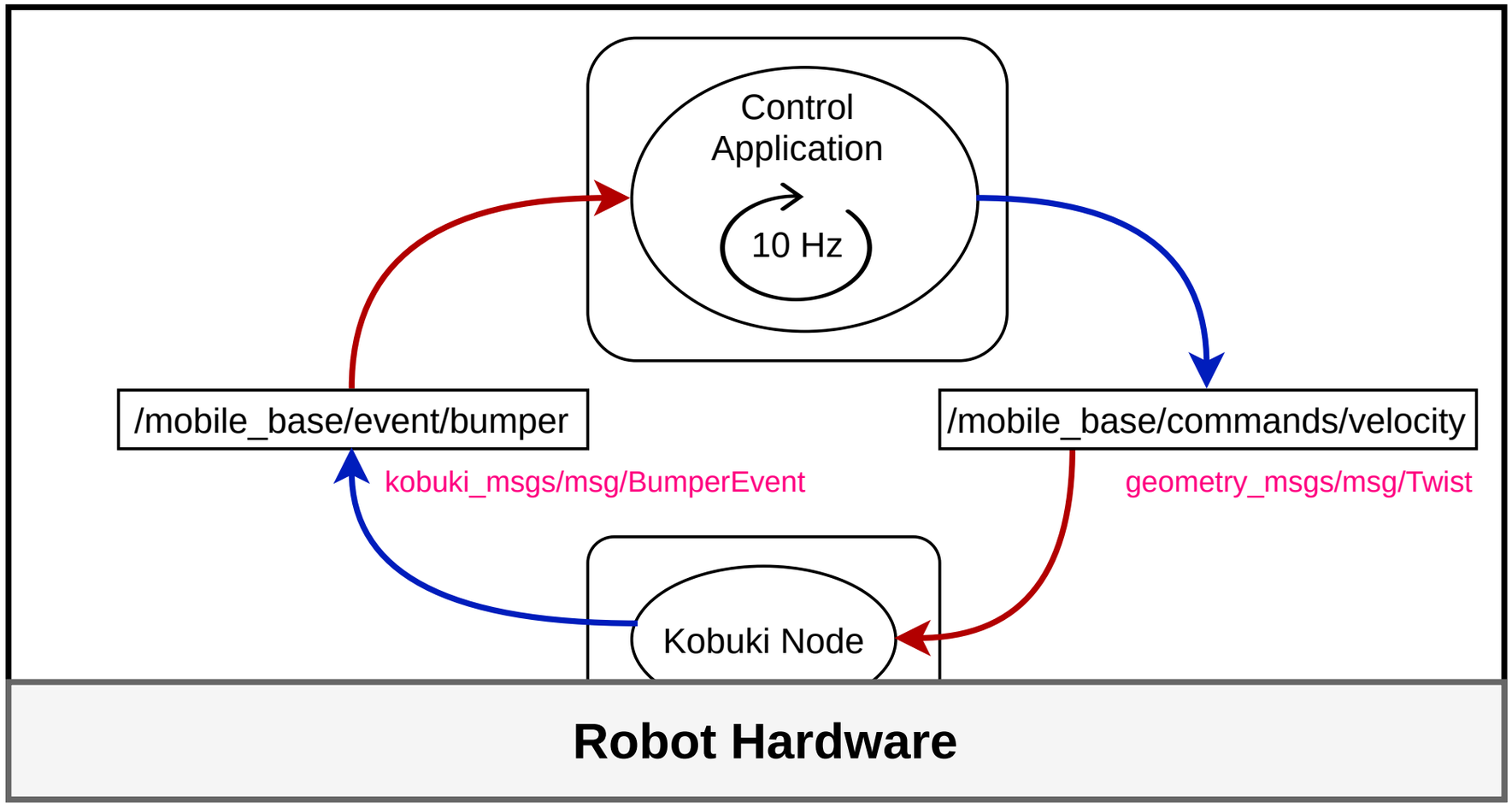
May 23rd, 2022

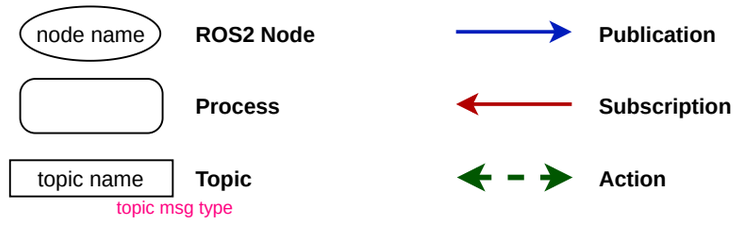
Intelligent Robotics Lab

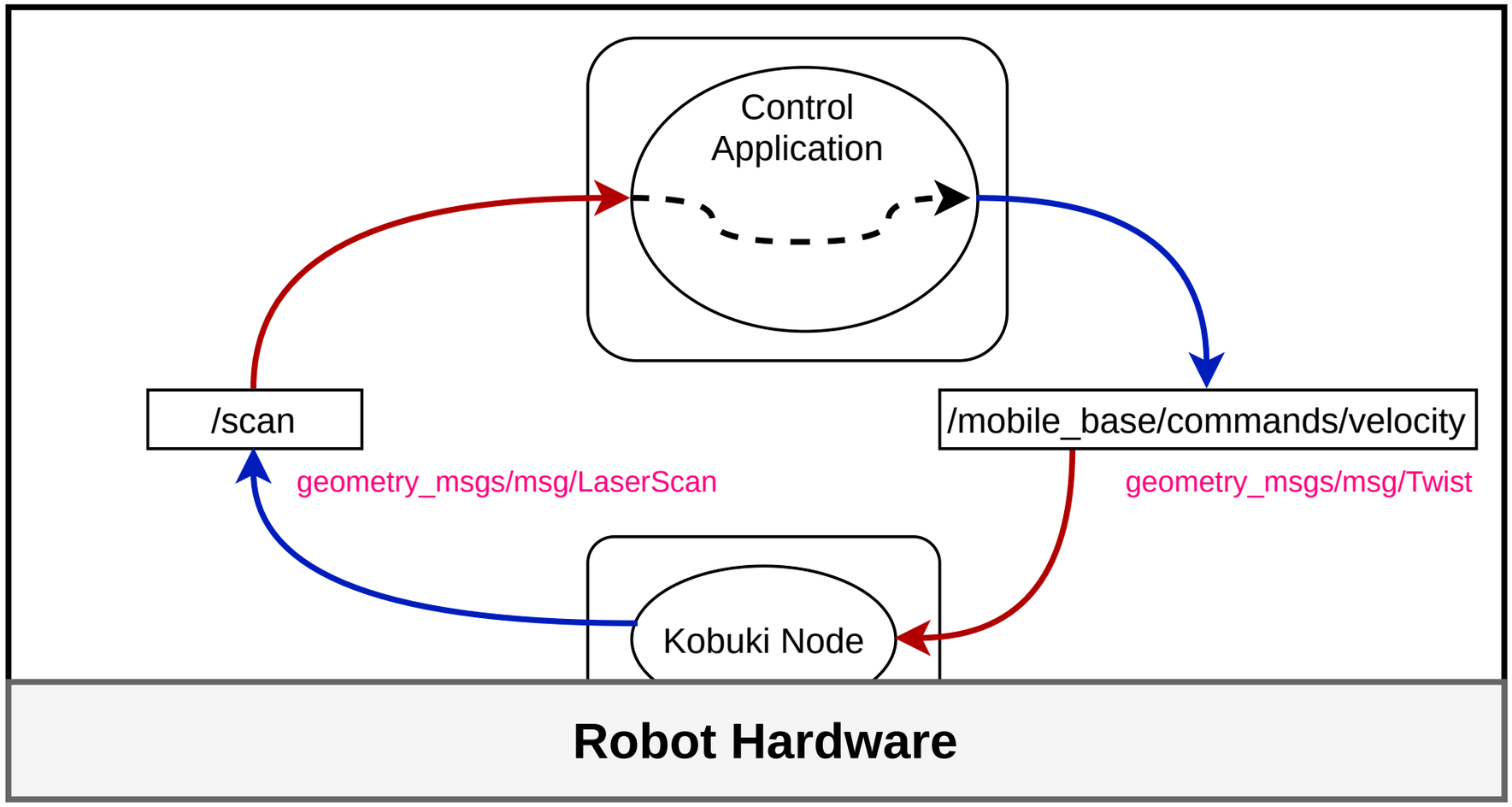
Design

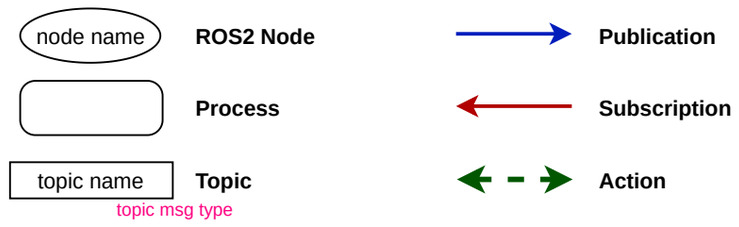

User Code
User Nodes
ROS Client Layer (RCL)
rclcpp
rclpy
rclc
Other APIs
Java
rcl (ROS2 C Implementation)
ROS Middleware Layer (RMW)
rmw (DDS API)
Fast DDS
Cyclone DDS
rti DDS
OS Layer
Windows
LINUX
iOS

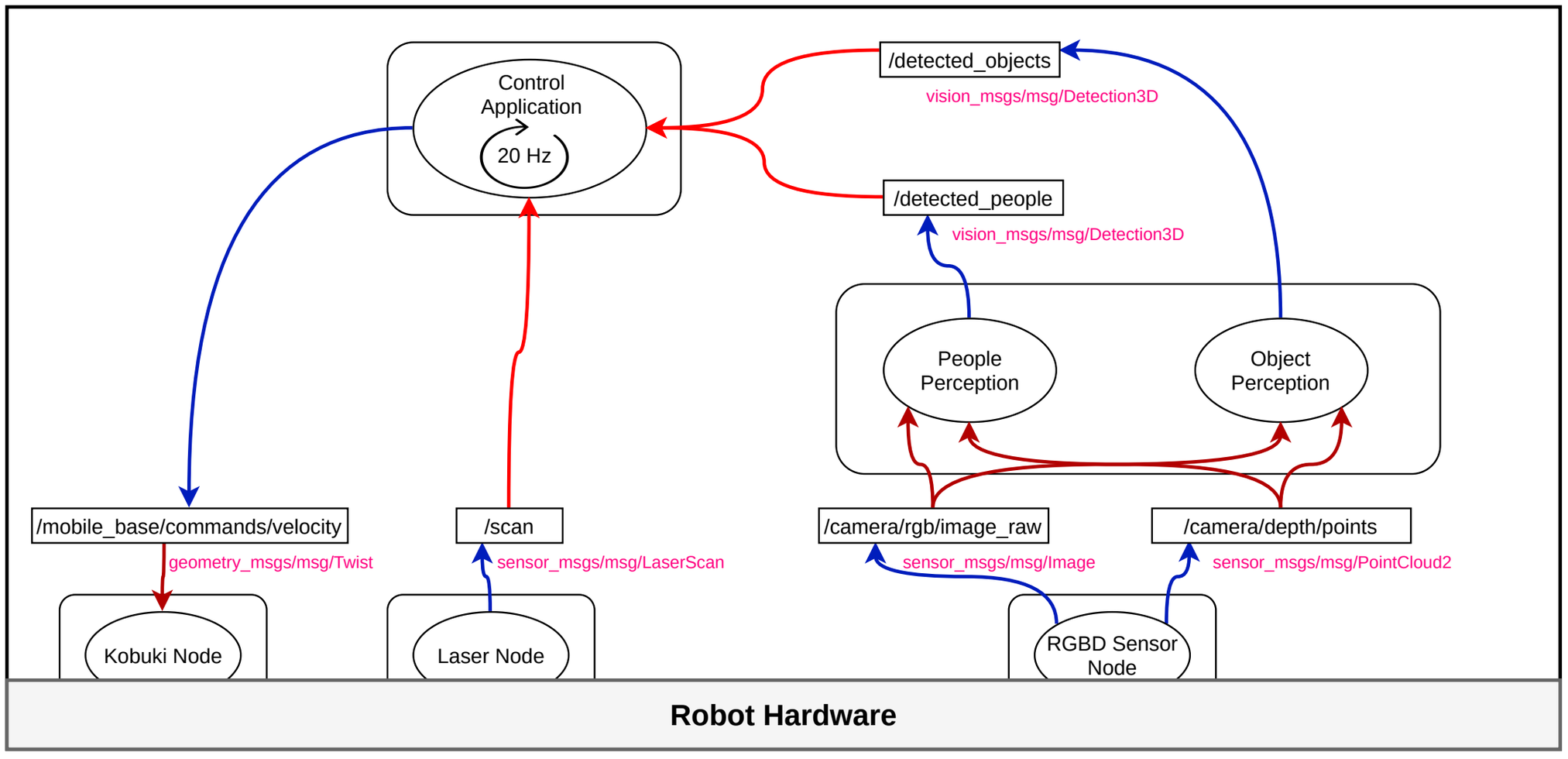
Applications in ROS2

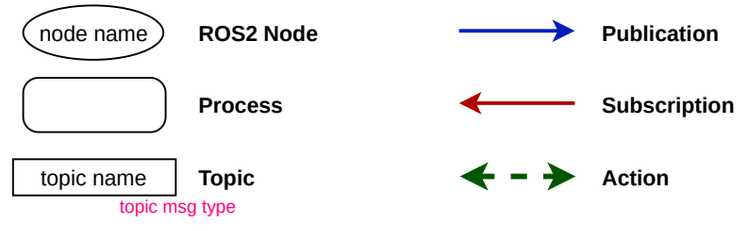

Control
Application
10 Hz
/mobile_base/event/bumper
kobuki_msgs/msg/BumperEvent
/mobile_base/commands/velocity
geometry_msgs/msg/Twist
Kobuki Node
Robot Hardware

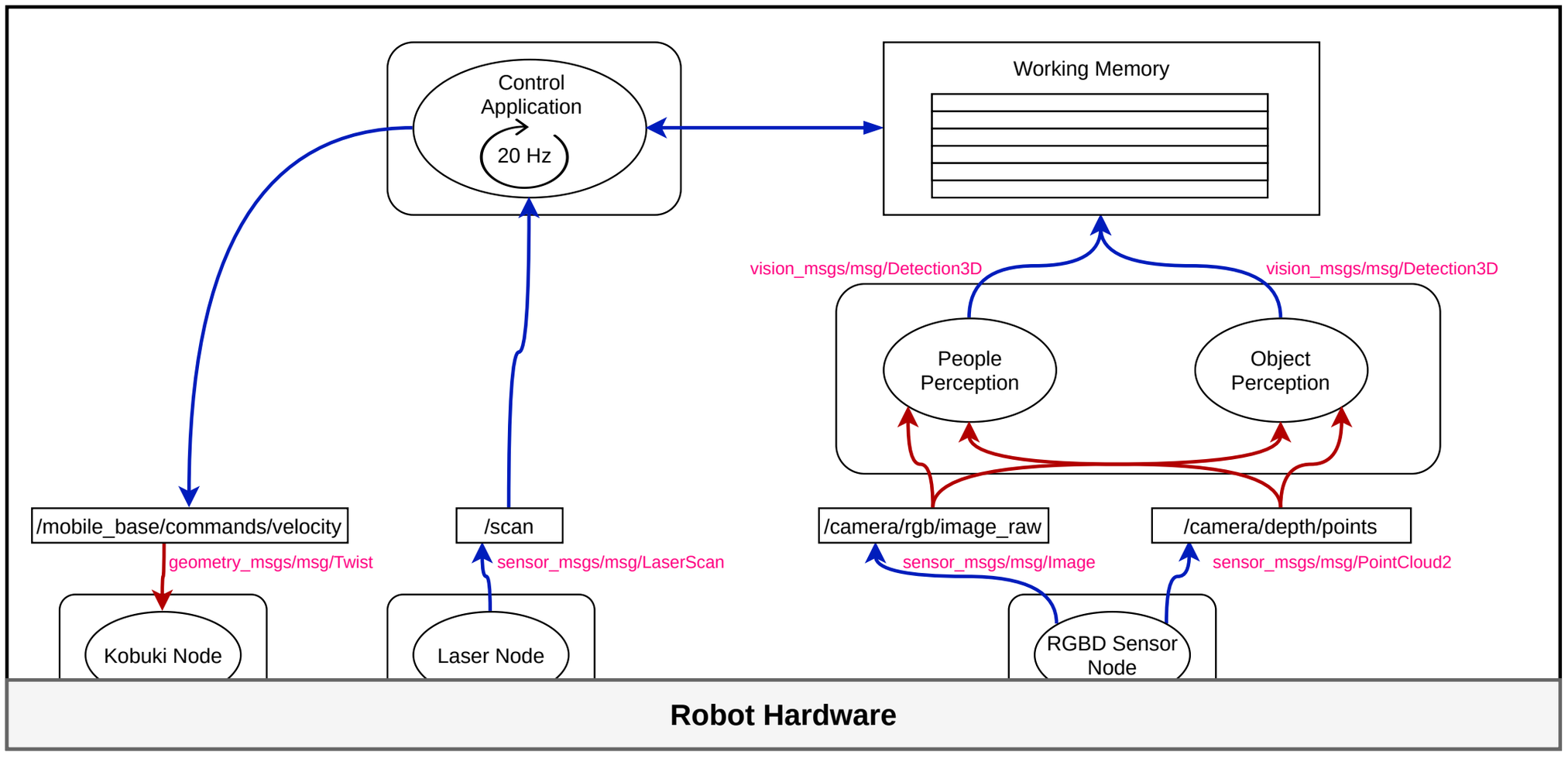

node name
ROS2 Node
Process
topic name
Topic
topic msg type
Publication
Subscription
Action

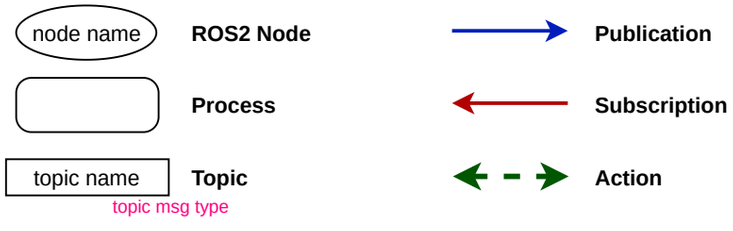

Control
Application
/scan
geometry_msgs/msg/LaserScan
/mobile_base/commands/velocity
geometry_msgs/msg/Twist
Kobuki Node
Robot Hardware

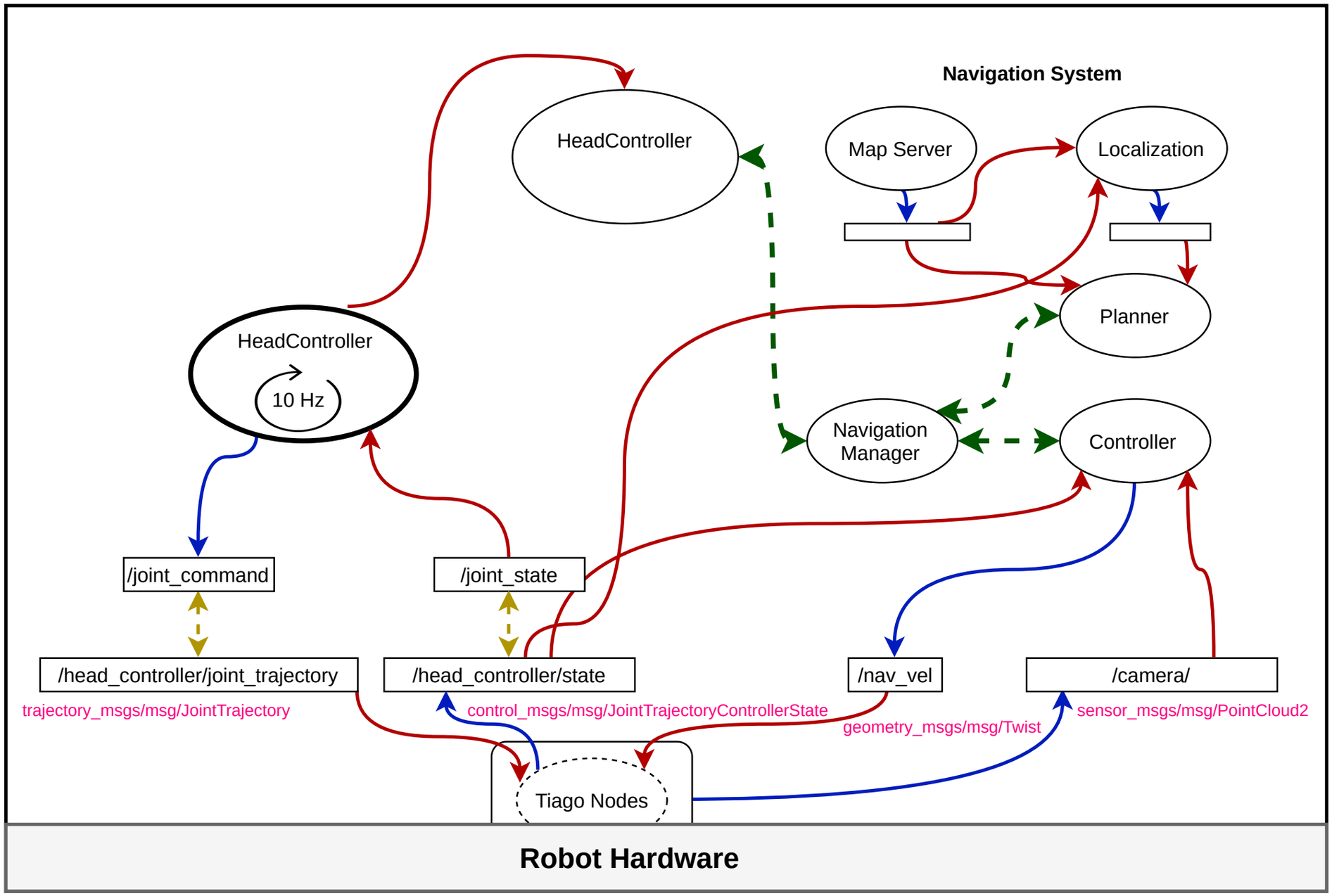

node name
ROS2 Node
Process
topic name
Topic
topic msg type
Publication
Subscription
Action

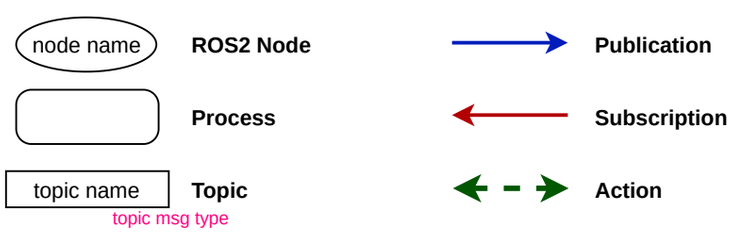

Control Application
20 Hz
/detected_objects
vision_msgs/msg/Detection3D
/detected_people
vision_msgs/msg/Detection3D
People Perception
Object Perception
/mobile_base/commands/velocity
geometry_msgs/msg/Twist
/scan
sensor_msgs/msg/LaserScan
/camera/rgb/image_raw
sensor_msgs/msg/Image
/camera/depth/points
sensor_msgs/msg/PointCloud2
Kobuki Node
Laser Node
RGBD Sensor Node
Robot Hardware


node name
ROS2 Node
Process
topic name
Topic
topic msg type
Publication
Subscription
Action


Control
Application
20 Hz
Working Memory
vision_msgs/msg/Detection3D
vision_msgs/msg/Detection3D
People
Perception
Object
Perception
/mobile_base/commands/velocity
geometry_msgs/msg/Twist
/scan
sensor_msgs/msg/LaserScan
/camera/rgb/image_raw
sensor_msgs/msg/Image
/camera/depth/points
sensor_msgs/msg/PointCloud2
Kobuki Node
Laser Node
RGBD Sensor
Node
Robot Hardware

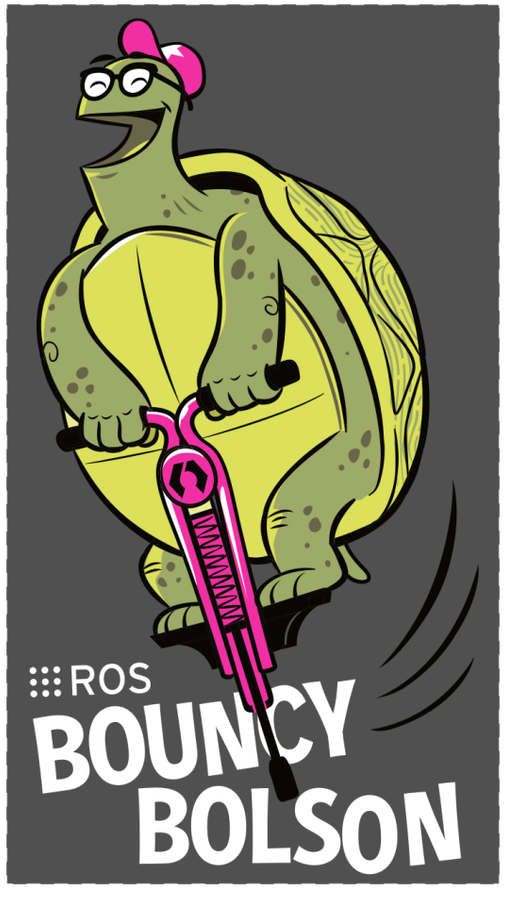

node name
ROS2 Node
Process
topic name
Topic
topic msg type
Publication
Subscription
Action





Intelligent Robotics Lab

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
```

- DDS Selection

Fast-RTPS: Official ROS2 DDS vendor (by default).

OpenSplice: `export RMW_IMPLEMENTATION=rmw_opensplice_cpp`

RTI Connext: `export RMW_IMPLEMENTATION=rmw_connext_cpp`

Eclipse Cyclone: `export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp`

Firsts steps ROS2

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

rclcpp API

intro_ros2/src/simple_node.cpp

- Extensive use of advanced C++ (C++14 mostly)
- `rclcpp::`
- Nodes are no longer processes; they are objects
- No need for `NodeHandle`
- `ros2` as super command (ros2cli)

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

ROS2 Cheat Sheet

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

```
$ ros2 topic info /chatter –verbose
```

QoS policies

History	Keep last	Keep all
Depth	Size	
Reliability	Best effort	Reliable
Durability	Transient local	Volatile
Deadline	Duration	
Lifespan	Duration	
liveliness	Automatic	Manual by Topic
Lease Duration	Duration	

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

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

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

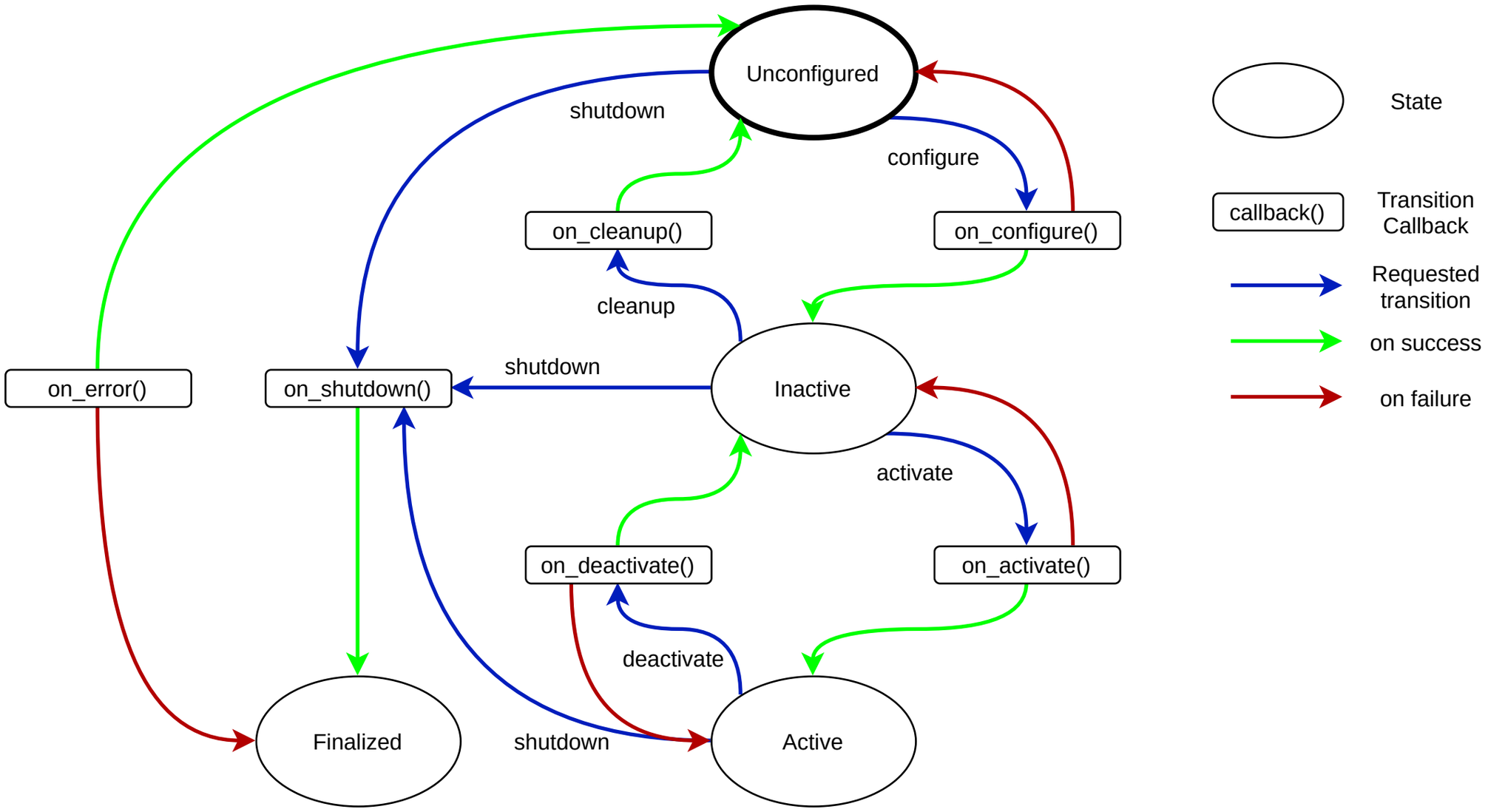
```
$ ros2 run parameters_ros2 reconf_param_node
$ ros2 param set /ReconfigurableNode speed 5.6
```

Lifecycle nodes

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

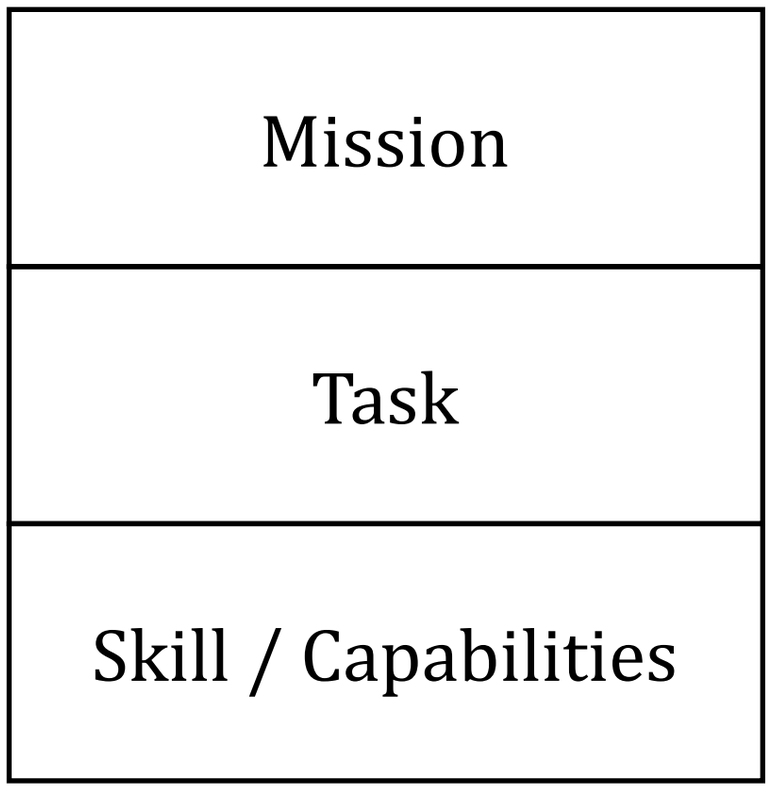
- Beyond Lifecycle… System Modes

Intelligent Robotics Lab

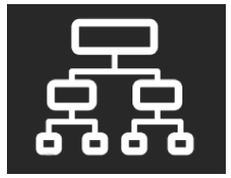
Examples of Software in ROS2

Intelligent Robotics Lab

Software Architectures

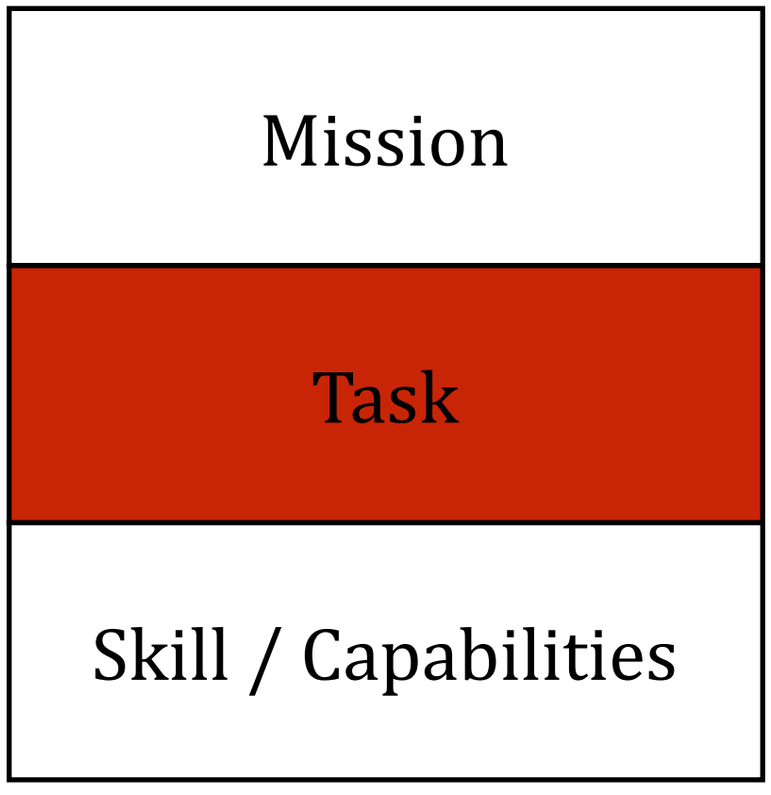


PlanSys2



NAV2

Software Architectures



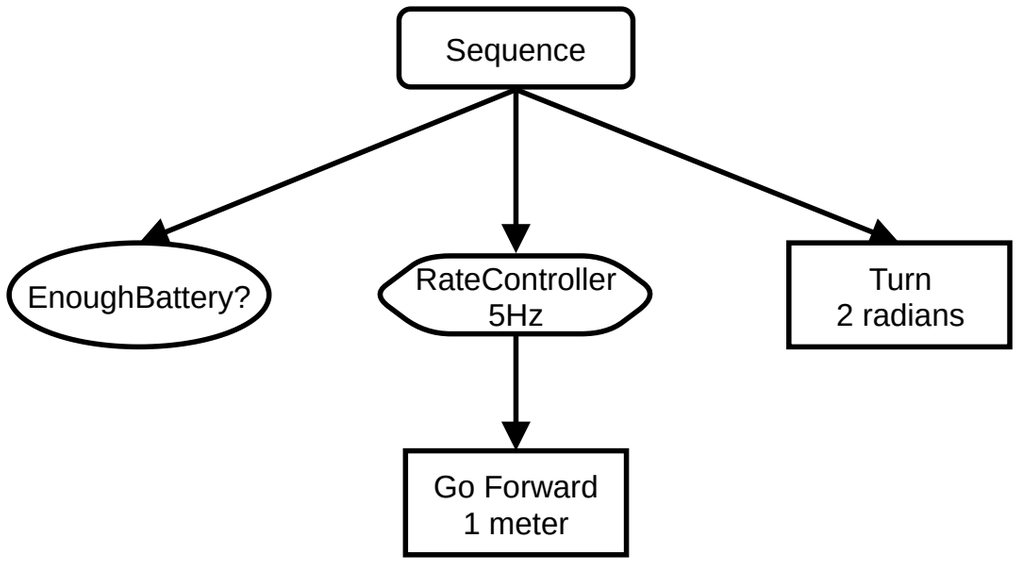
PlanSys2

NAV2

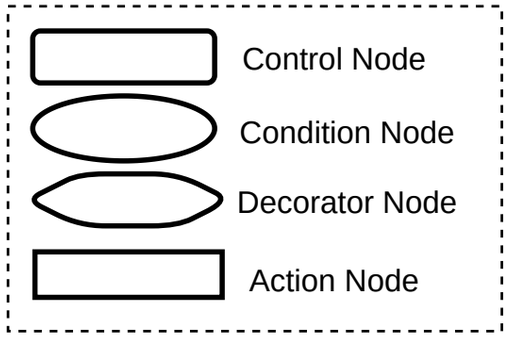
Behavior Trees

- Very popular in recent years
- Compared with FSM
- A Behavior Tree (BT) es a mathematical model to encode the control of a system
- It is a hierarchical data structure defined recursively from a root node with several child nodes.

- The basic operation is the **tick**:
  - SUCCESS
  - FAILURE
  - RUNNING

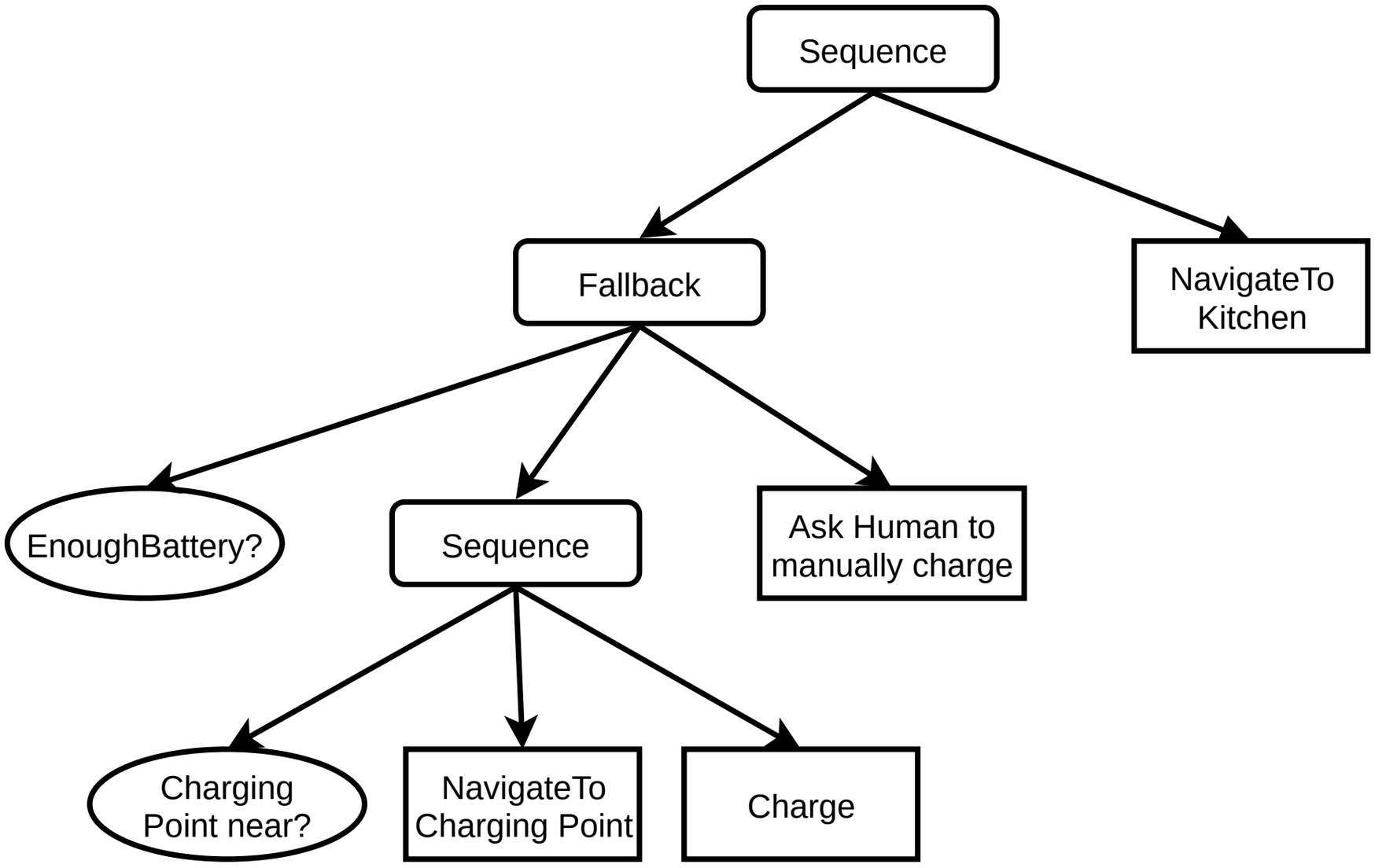


- BT node types:

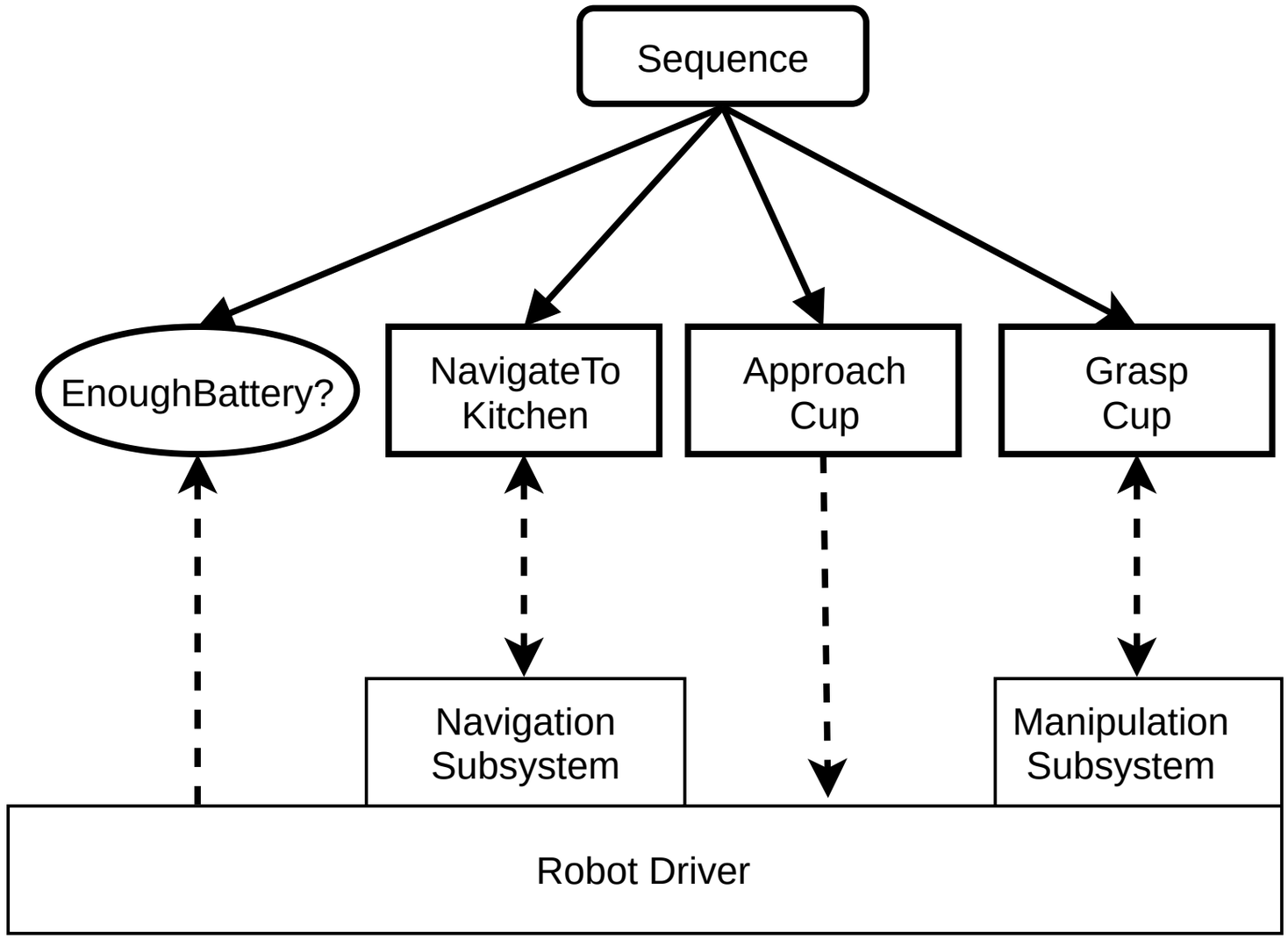


Intelligent Robotics Lab

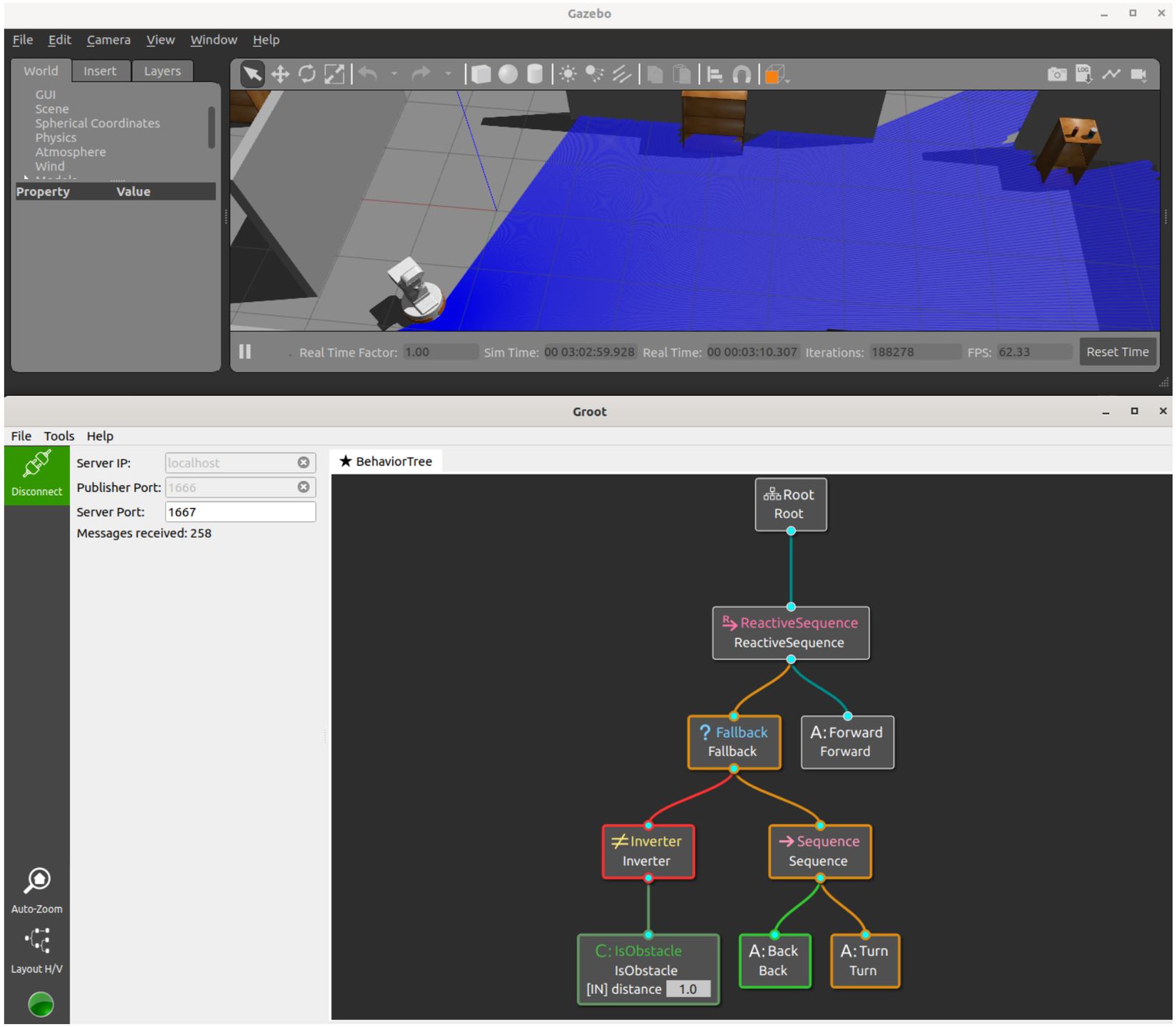
Behavior Trees



Behavior Trees

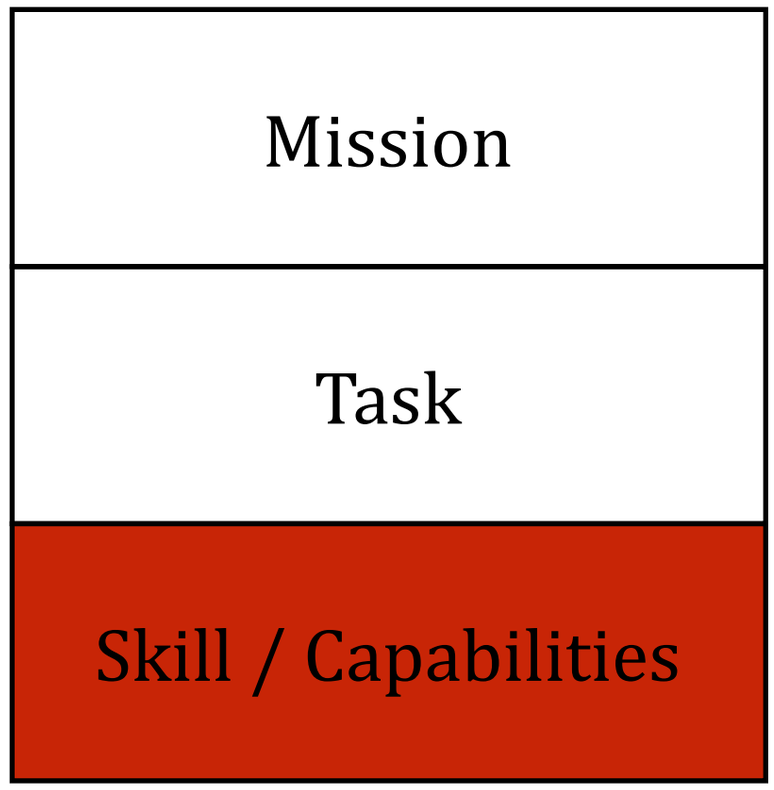


Behavior Trees

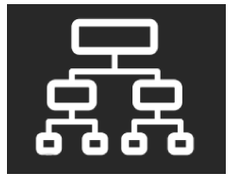


Intelligent Robotics Lab

Software Architectures



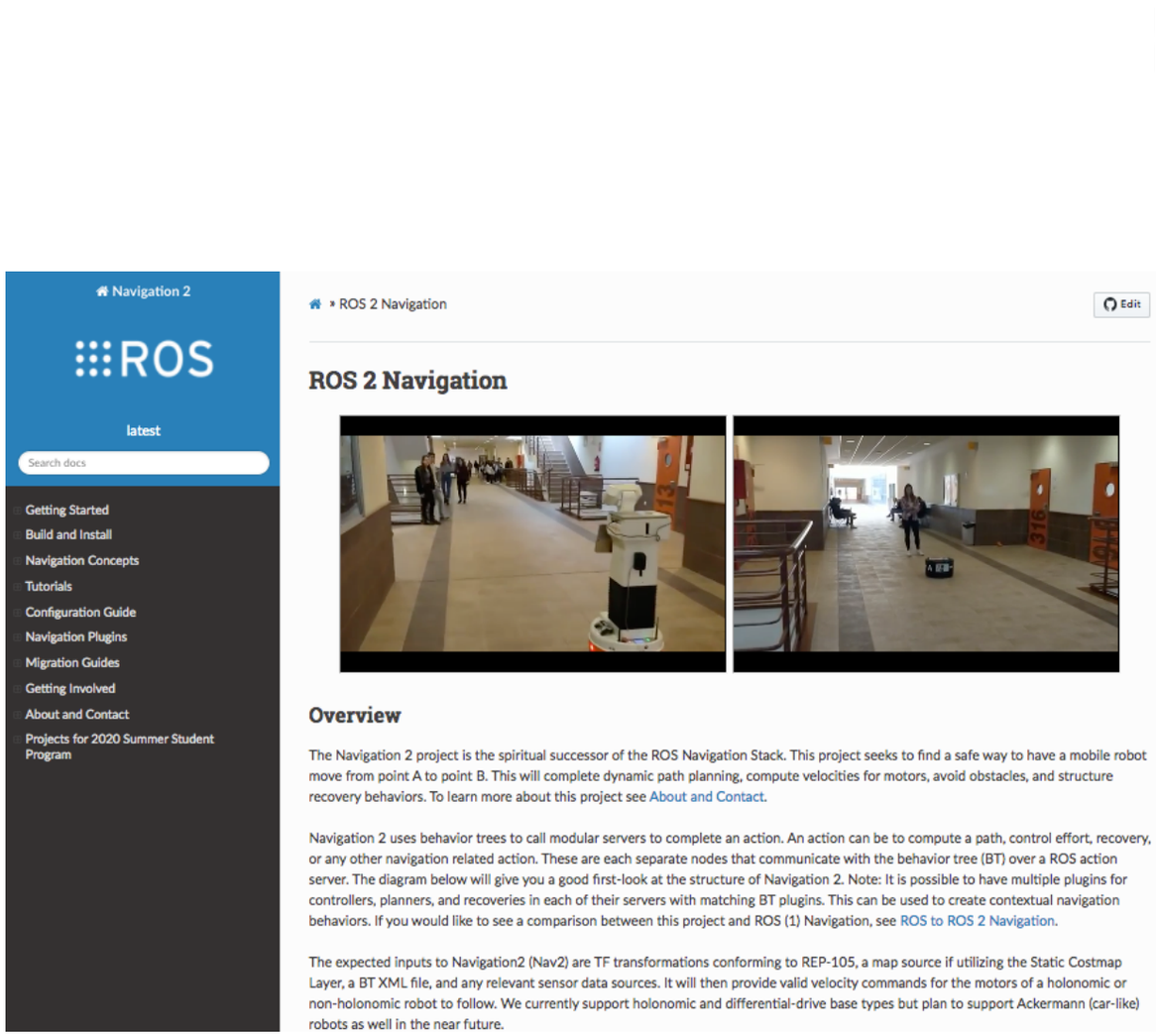
PlanSys2

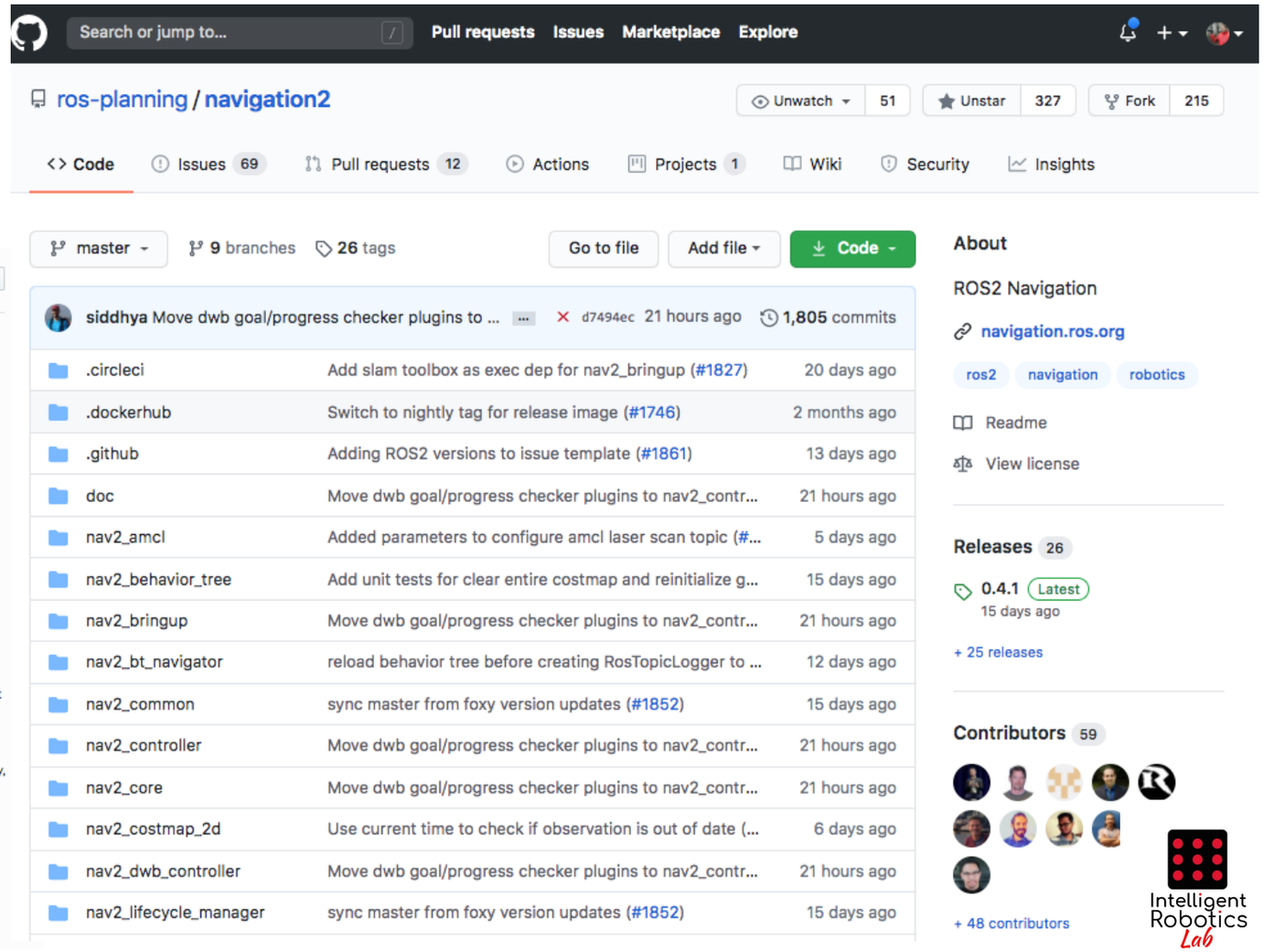


NAV2

Overview

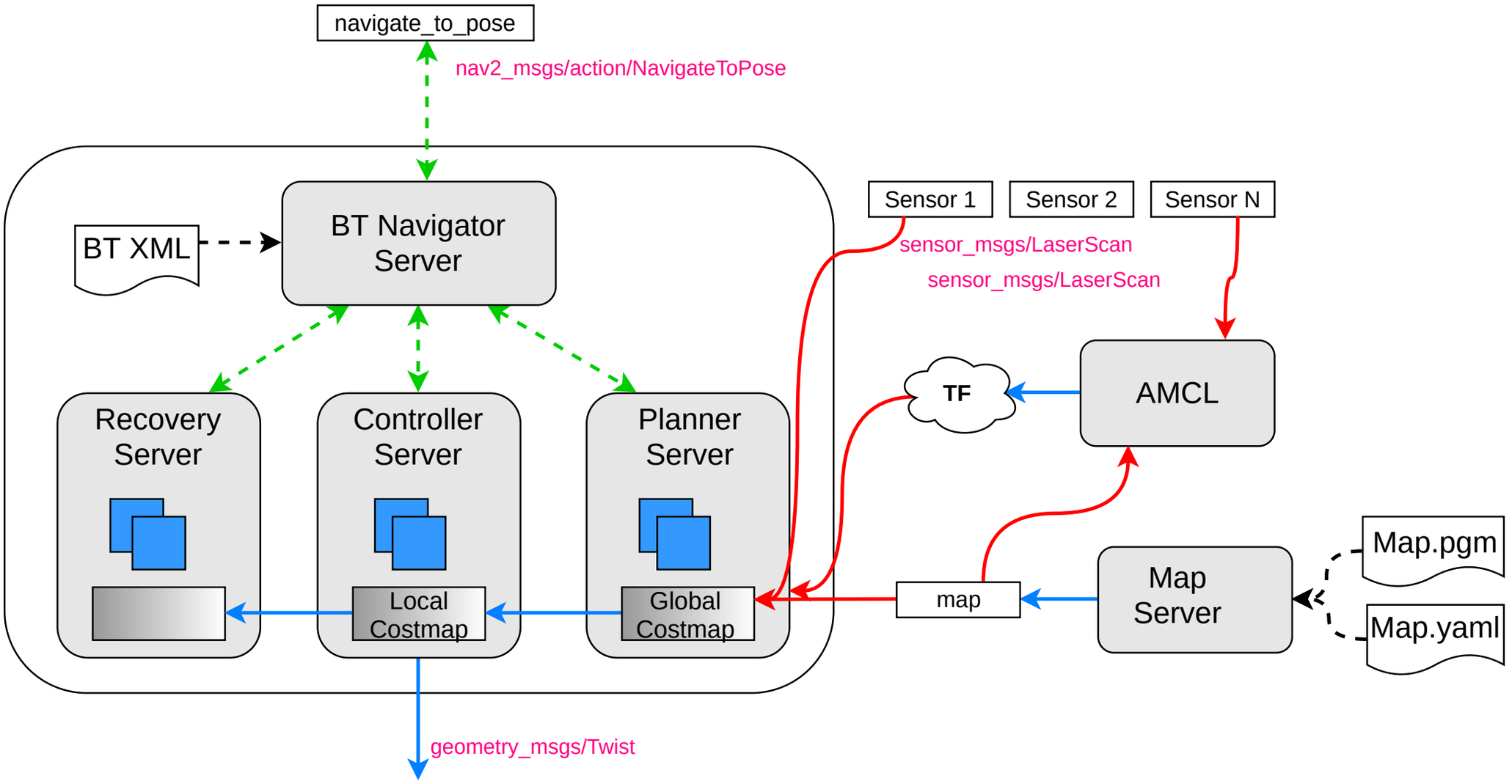
- Main author: **Steven Macenski**
- Reimplementation of Navigation stack
- Example of good practices in ROS2
- More algorithms, plugins, components…
- Contributions: 59 contributors, Intel, Samsung



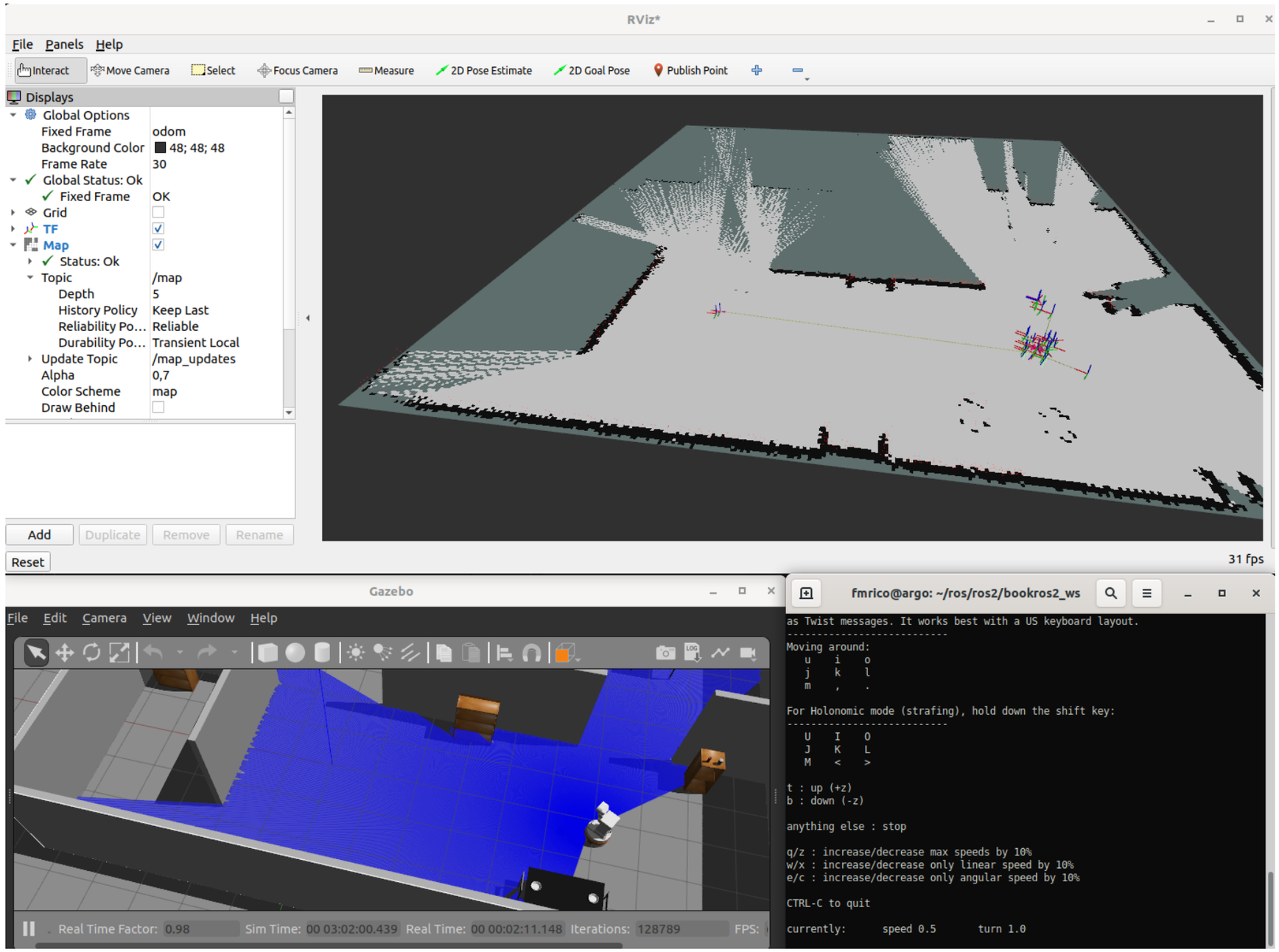


Intelligent Robotics Lab

Architecture

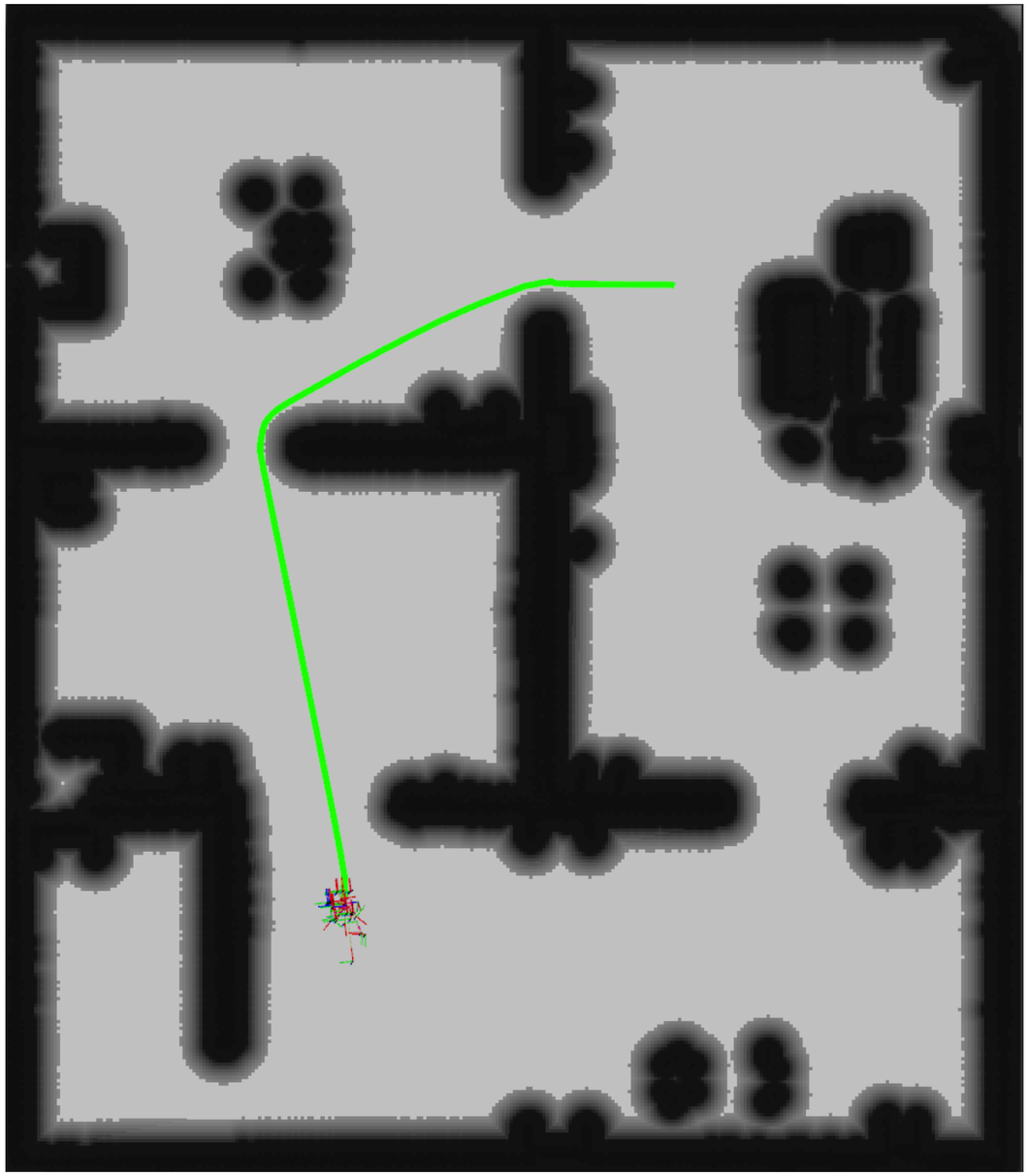


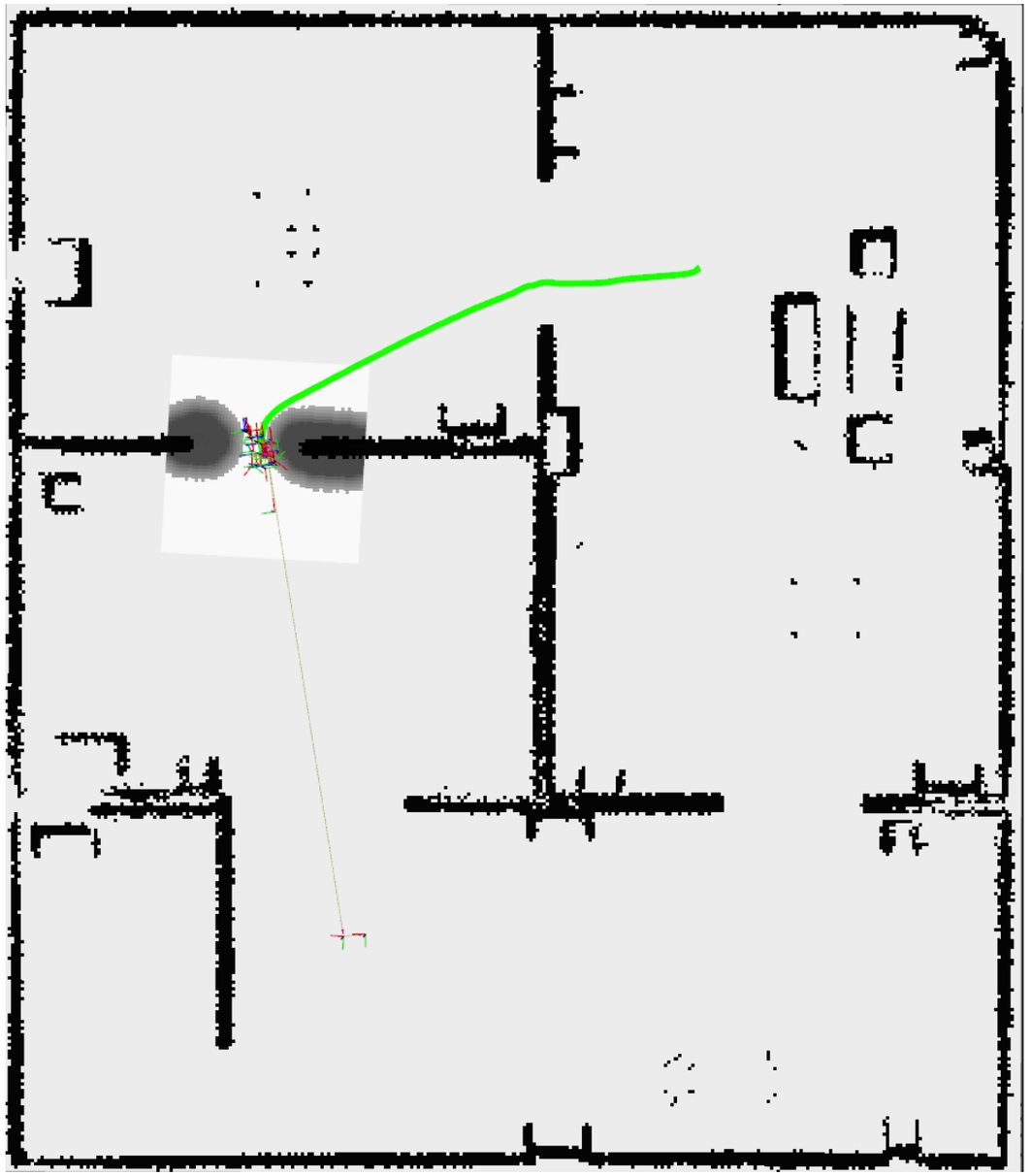
SLAM



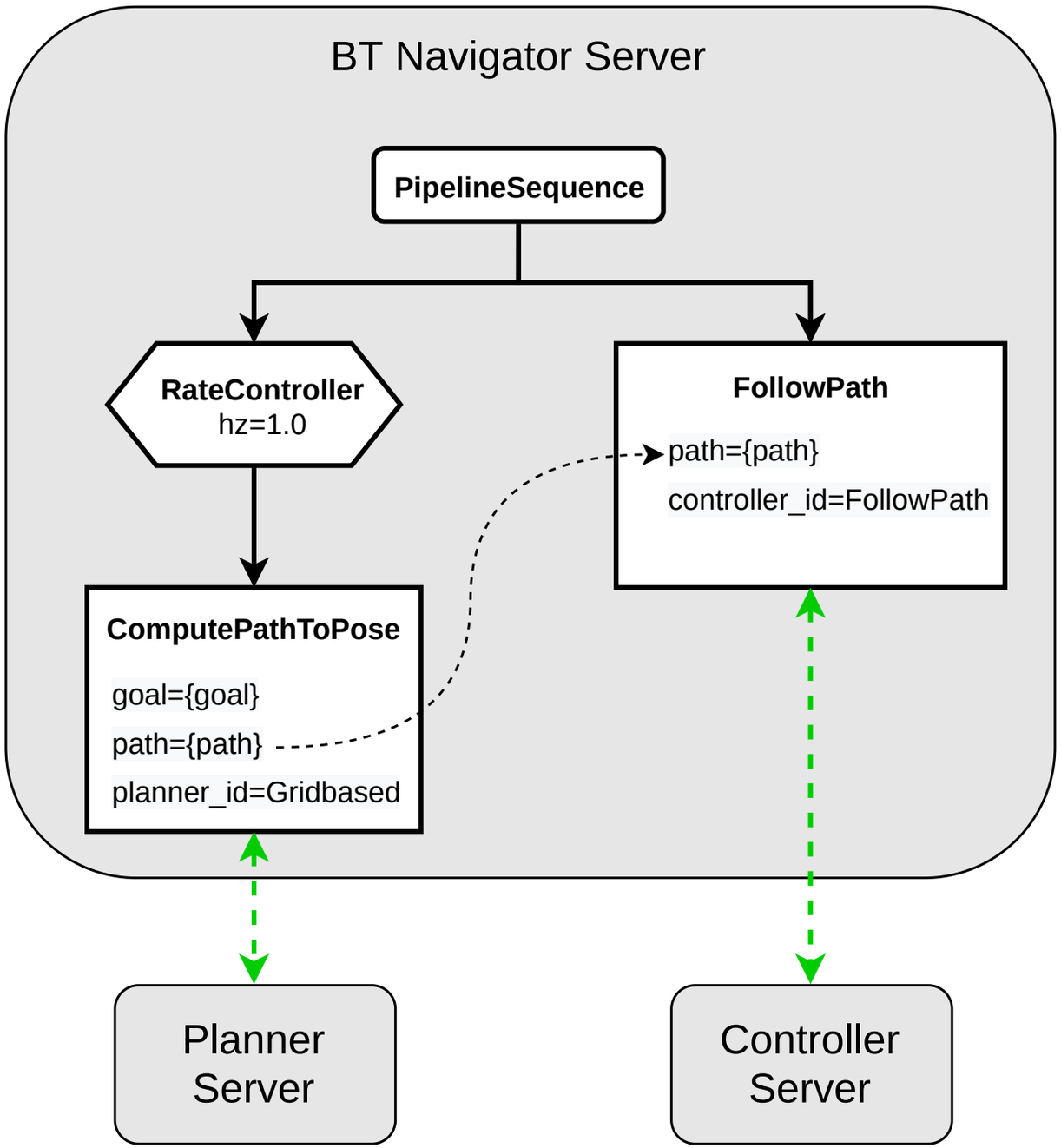
Intelligent Robotics Lab

Costmaps





BT Navigator



The marathon

THE MARATHON 2

A NAVIGATION SYSTEM

STEVE MACENSKI, FRANCISCO MARTIN, RUFFIN WHITE,
JONATHAN GINES

IROS 2020

Intelligent Robotics Lab

Results

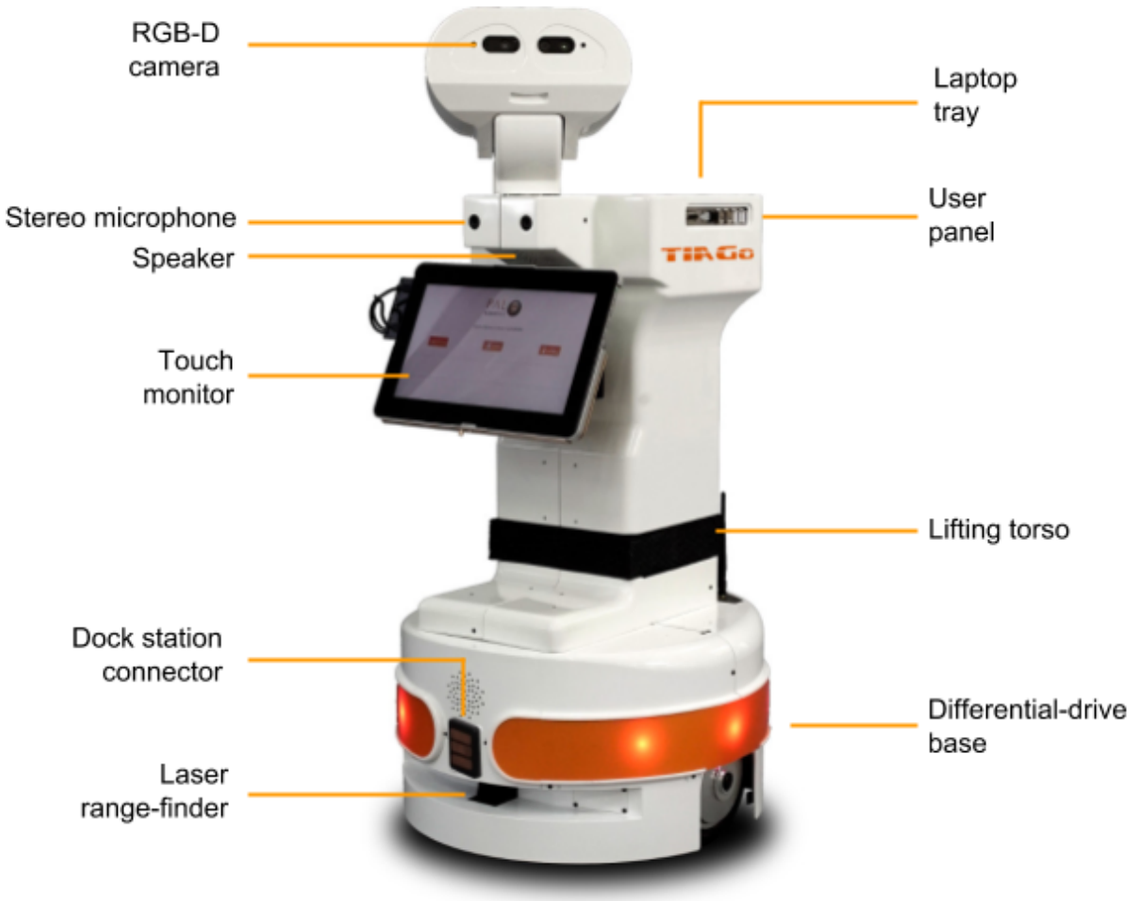
- Ultra Marathon: **37 miles**
- No collisions
- No manual restart
- No navigations aborted
- No students were damaged!!!

	RB1	TIAGo	Total
Time (hrs)	9.4	13.4	**22.8**
Distance (miles)	15.6	21.8	**37.4**
Recoveries	52	116	**168**
Recoveries per mile	3.3	5.3	**4.3**
Avg. speed (m/s)	0.39	0.35	**0.37**
Max speed (m/s)	0.45	0.45	**0.45**
Num. of Collisions	0	0	**0**
Num. of Emergency stops	0	0	**0**

The robots

Tiago

Pal Robotics (Barcelona, Spain)



- Base + Torso(0.54 m x 1.45 m)
- Max speed: 1.0 ms/s
- Battery: 2 x 36V 20Ah
- Intel i7 - 7700, 8GB RAM, 250 GB SSD
- Ubuntu 16.06 LTS (Kinetic)
- Camera RGBD Orbbec Astra S
- Laser Sick TIM561
- Navigation2 executed externally

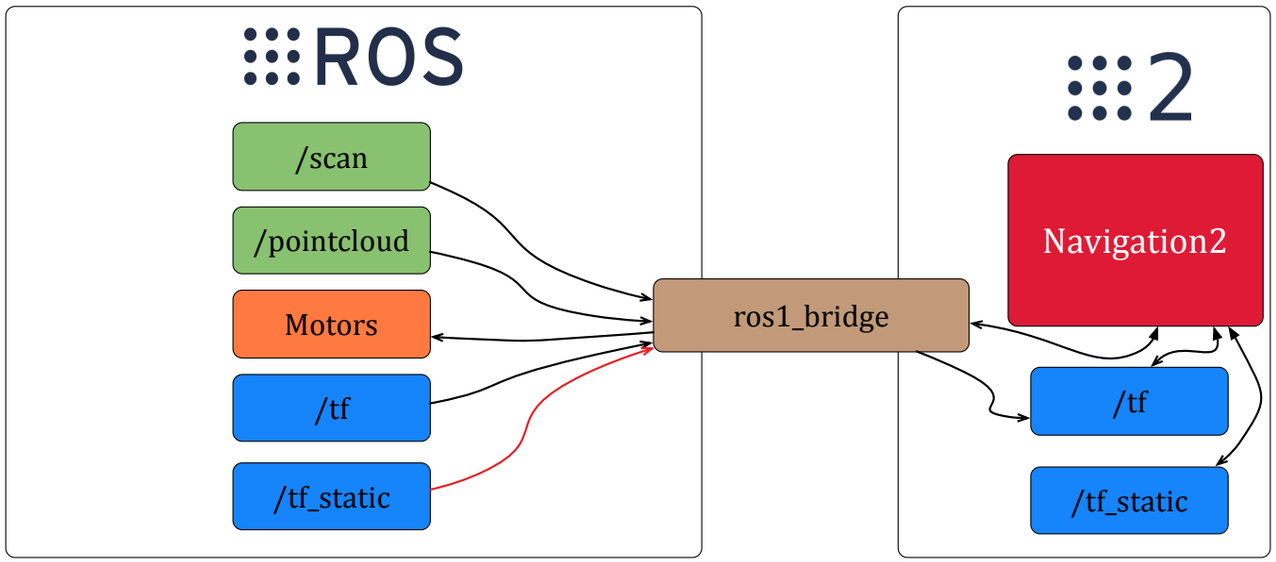
RB1 base

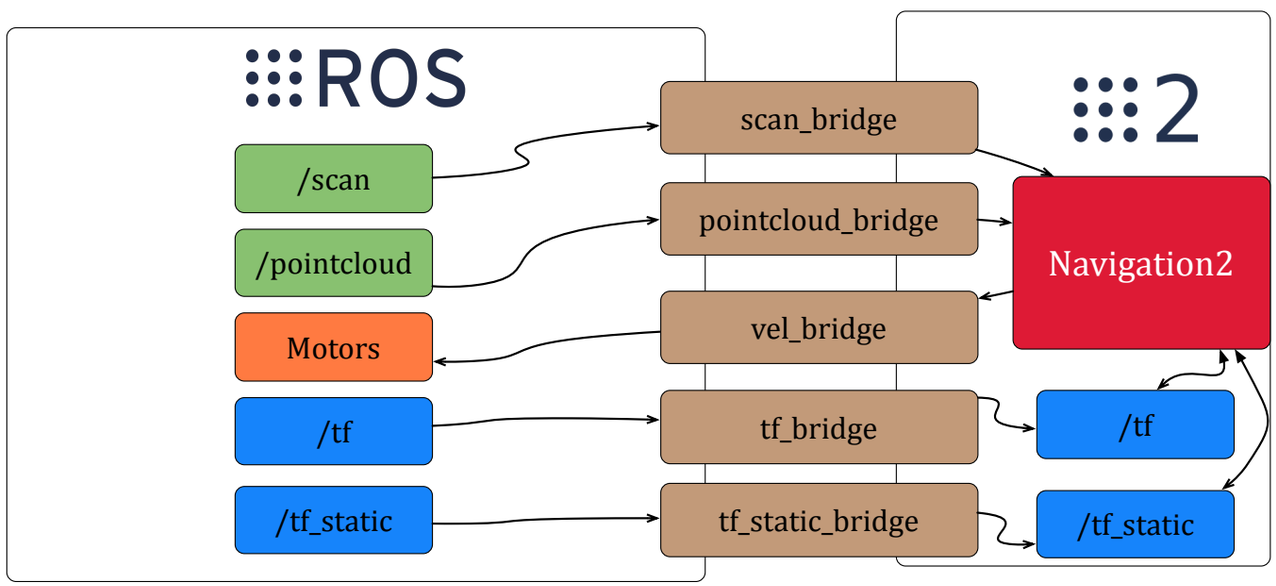
Robotnik (Valencia, Spain)

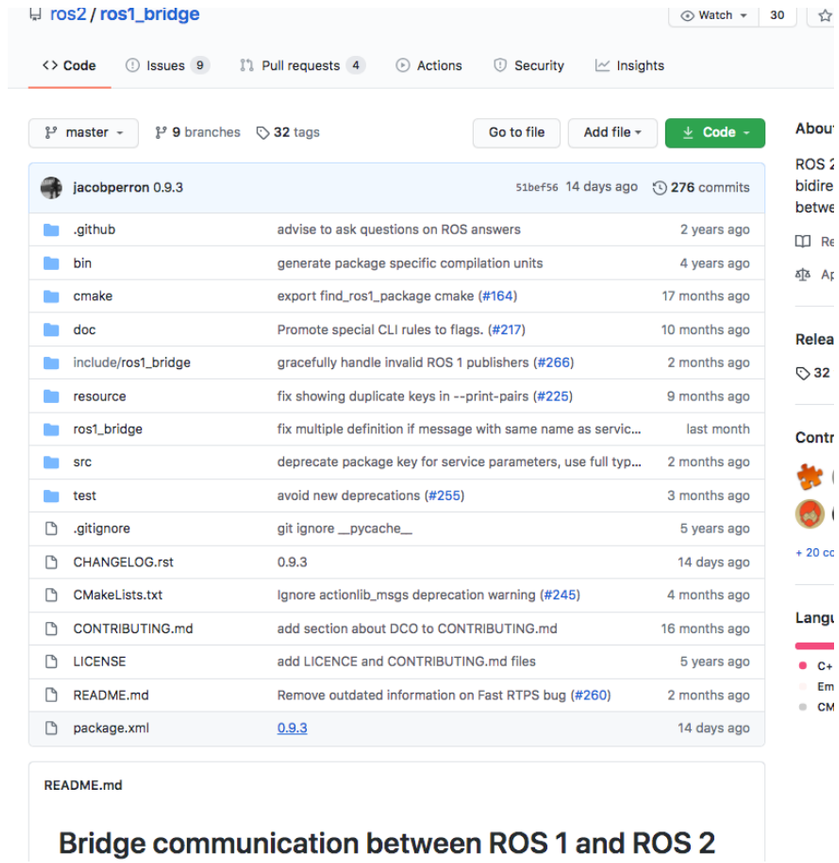
- Base (0.50 m x 0.251 m)
- Max speed: 1.5 ms/s
- Battery: 24V 30Ah
- Intel i7 - 7567, 8GB RAM, 120 GB SSD
- Ubuntu 18.06 LTS (Melodic)
- Camera RGBD Orbbec Astra
- Laser Sick TIM571
- Navigation2 executed onboard

The problems

- The drivers of the robot was for ROS1
- We used ros1_bridge
- Continuous flow problem

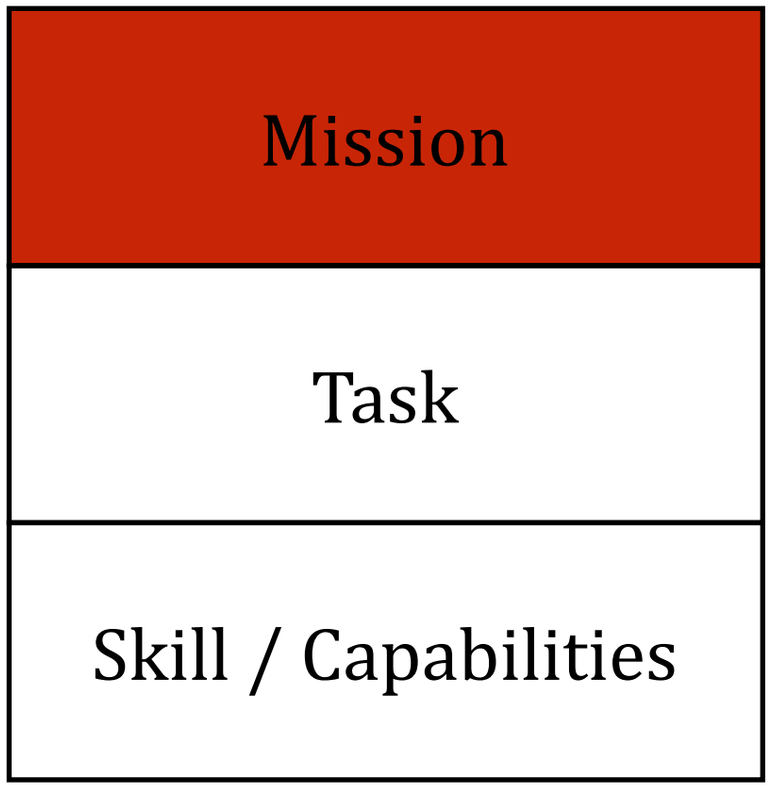




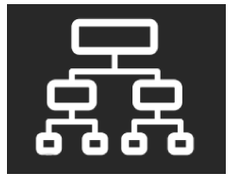


- QoS per topic

Software Architectures



PlanSys2



NAV2

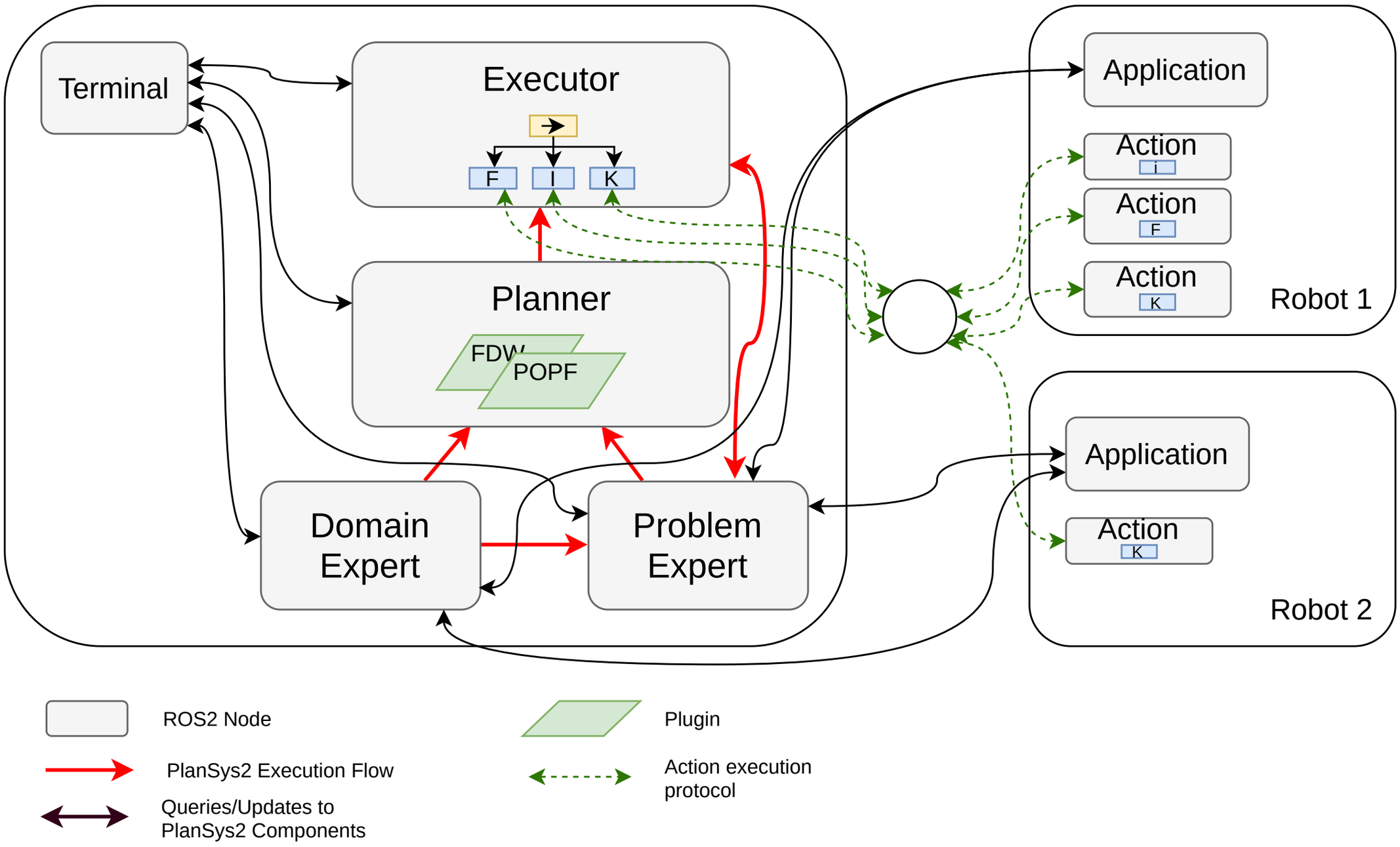
ROS 2 Planning System

- ROS2 Planning System (PlanSys2 in short) is a framework for symbolic planning that incorporates novel approaches for execution on robots working in demanding environments

- PlanSys2 aims to be the reference task planning framework in ROS2

- Among its main features, it can be highlighted the optimized execution, based on Behavior Trees, of plans through a new actions auction protocol and its multi-robot planning capabilities

Intelligent Robotics Lab


Terminal
Executor
F
I
K
Planner
FDW
POPF
Domain Expert
Problem Expert
Application
Action
i
Action
F
Action
K
Robot 1
Application
Action
K
Robot 2
ROS2 Node
PlanSys2 Execution Flow
Queries/Updates to PlanSys2 Components
Plugin
Action execution protocol

2 Planning System

PlanSys2 aims to be the reference Planning System in ROS2

- *Reliable, Secure and Predictable*
- *Multi-robot and Specialized Action Performers*
- *Modular y Extensible*
- *Efficient and Explainable*
- *Open Source*

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

Migrate!!

1. package.xml and CMakeLists.txt
   a. Look for dependencies in packages
   b. Look for dependencies in sources
   c. Migrate dependencies, if possible
2. **Direct migration:**
   a. New message and rclcpp headers
   b. Create a rclcpp::Node. Direct for Nodelets.
   c. Change every publisher / subscriber
   d. ROS_INFO to RCLCPP_INFO,…
   e. Services and actions are a little harder
   f. Launchers
3. **Improved migration:**
   a. Nodes to classes and instantiate in one executable
   b. Consider usage lifecycle nodes
   c. Rethink QoS
   d. Split in Nodes
   e. Multiple nodes with executors

MIGRATE TO ROS2 !!!!!

Intelligent Robotics Lab

ROS2

for

ROS Developers

Muchas gracias!!!

Prof. Dr. Francisco Martín Rico

Intelligent Robotics Lab

Universidad Rey Juan Carlos

Madrid 2-03-2022