Research Track II
2° Assignment

Carmine Tommaso Recchiuto

Jupyter Notebook

Remark. Please consider that all example shown during previous classes were executed on a new container, whose .bashrc script sources ROS. In case you try to run them on your living container, where you have removed the "source /opt/ros/noetic/setup.bash" line, you need to source ROS manually before launching the Jupyter terminal, in case you intend using ROS in your Jupyter Notebook.

Also, consider that there is also a specific package for using ROS2 in jupyter: zmk5/jupyter-ros2: Jupyter widget helpers for ros2, the Next-Generation of the Robot Operating System (github.com) (It's actually a fork of the original jupyros). You can also still integrate classic ROS2 python functionalities in Jupyter.

Jupyter Notebook

Exercise -> CarmineD8/jupyter (github.com)

In the repository, you will find 3 jupyter notebooks (you may launch roslaunch robot_description sim2.launch in a terminal on the server to test them):

- the first one uses the jupyros extension for sending a linear and angular velocity to the robot and matplotlib to display the x and y position of the robot
- the second one uses jupyter widgets and classic ROS functionalities to pilot the robot
- the third one adds the 3d visualization of the robot.

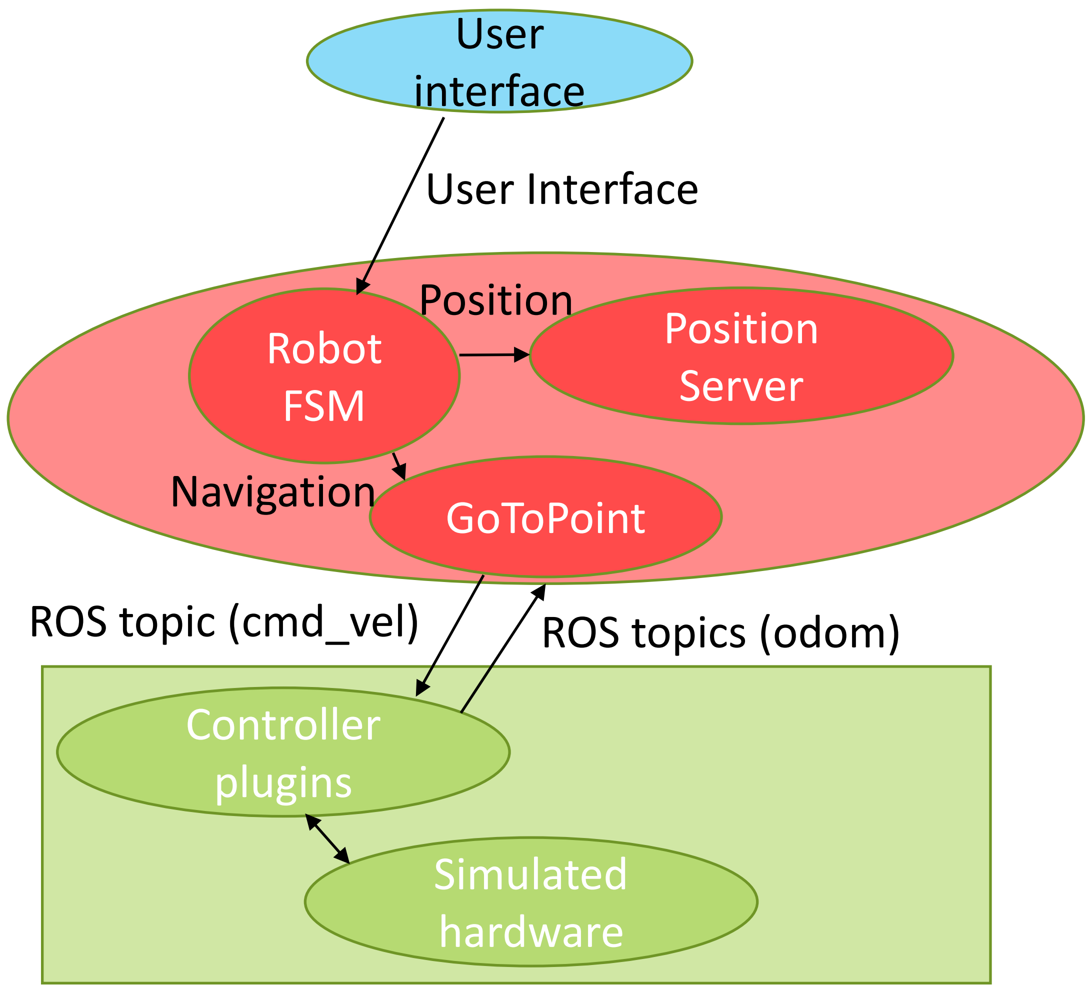
2° Assignment

We start from the 1st assignment of the course. You can still find it here: CarmineD8/rt2_assignment1: Package for the first assignment of the Research Track 2 course (github.com) and after building the workspace, you can run the launch file with:

- *roslaunch rt2_assignment1 sim.launch*

In particular, you can start from the branch *action* that you have created for the 1st assignment.

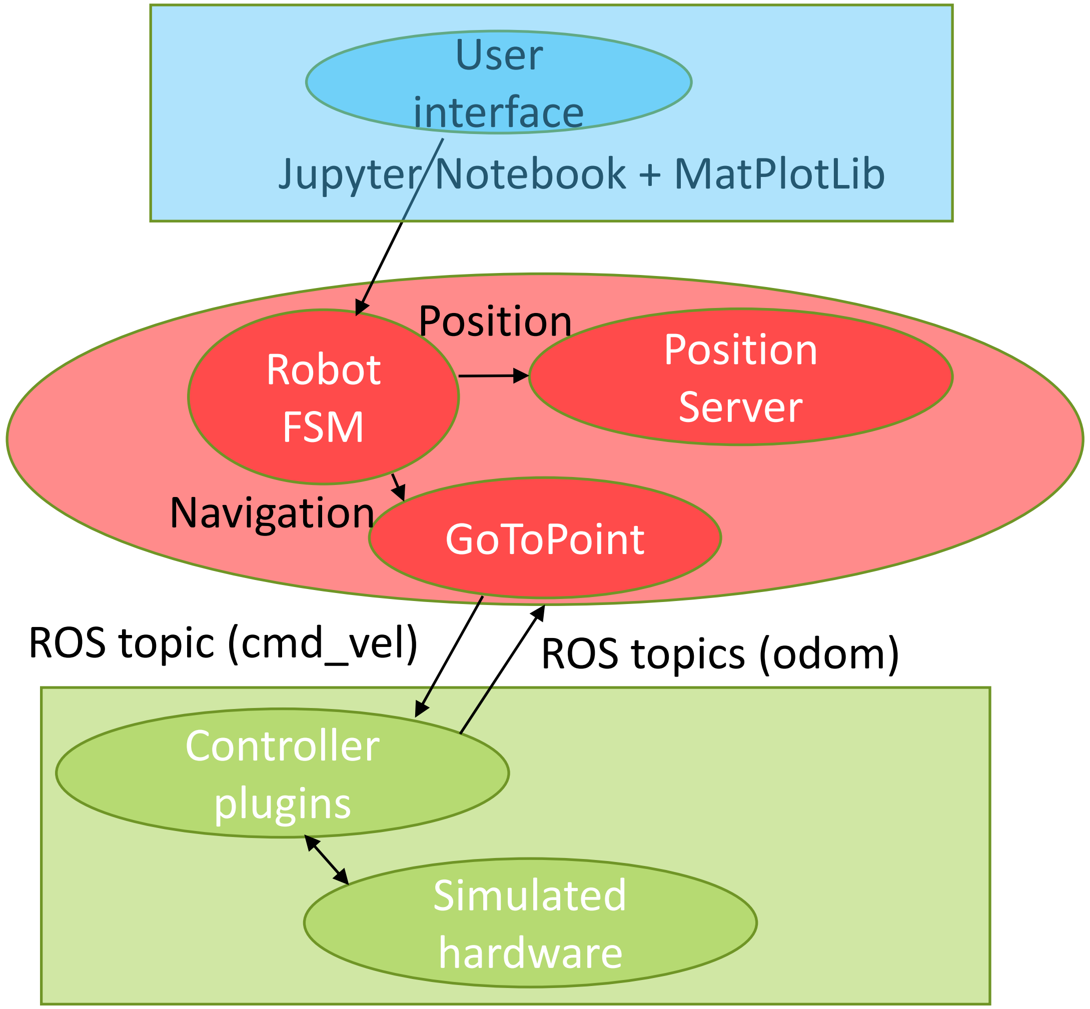
2° Assignment



- The initial structure of the assignment was a modular structure with two services (Position Server and GoToPoint). In the branch action, you have written the GoToPoint module as an action server.

- Some limitations:
  - The User Interface is only able to start / stop the robot behaviour
  - The User Interface is not so nice
  - We don't have any clear feedback about what the robot is doing and how

2° Assignment



- A possible solution may involve the usage of the Jupyter Notebook for interacting with our simulation. We have seen some example related to the integration of a Jupyter Notebook with a robotic simulation.

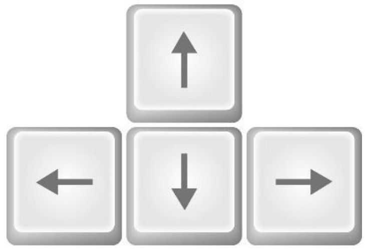
- Moreover, we can use the MatPlotLib library for have a graphical visualization for some data, so that the user may be aware of possible problems of the system

2° Assignment - Requirements

That’s exactly what you should do for your second assignment of this course. The task is to replace the User Interface of the first assignment with a nicer user interface developed with a Jupyter Notebook.

In particular we want the user interface to be able of:

- starting / stopping the robot “random position” behaviour by using two Buttons
- setting the linear and angular velocity by using two Sliders
- directly controlling the robot movements by using 5 Buttons, (forward, turn right, backward, turn left, stop), possibly placed in an intuitive fashion

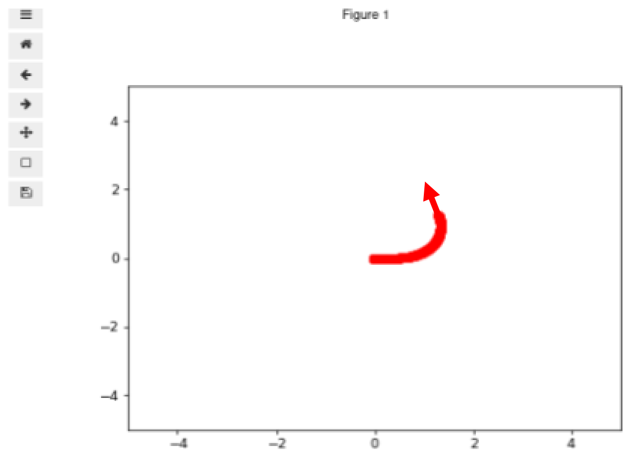


Please notice that if the robot is performing the random movement and the user wants to directly control the robot movements, the goal should be canceled, and the “random position” behaviour stopped. Also, the velocity set with the sliders should also be used for the “random position” behaviour

2° Assignment - Requirements

Also, we want to have a section of our jupyter Notebook for displaying some graphical information to the user. In particular we want to display:

- a line plot for visualizing cmd_vel vs. actual velocity (for linear and angular velocity)
- a bar plot displaying the number of reached targets and cancelled targets
- a hist plot showing the time required to reach targets
- an xy graph showing the robot's position and the orientation (**<u>orientation is optional</u>**)

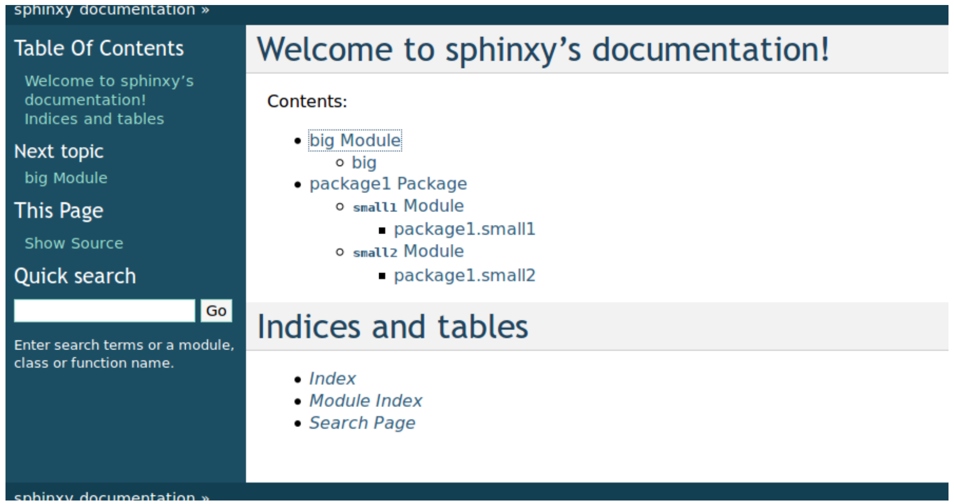


Concerning the last point, you are free to show the orientation in different ways. A possible solution could be to plot an arrow over the xy graph, oriented with the same angle of the robot.

2° Assignment - Requirements

- Finally, you should add code documentation (for the three nodes implementing the robot's behaviour of the action branch) written in Sphinx and Doxygen, and presented in html.

- Consider that you have to deal both with code in cpp and in python. You may create 2 additional branches for the documentation (and the Jupiter Notebooks, which will be the same in both branches), starting from the action branch.



sphinxy documentation »

Table Of Contents

Welcome to sphinxy's documentation!
Indices and tables

Next topic

big Module

This Page

Show Source

Quick search

Go

Enter search terms or a module, class or function name.

Welcome to sphinxy's documentation!

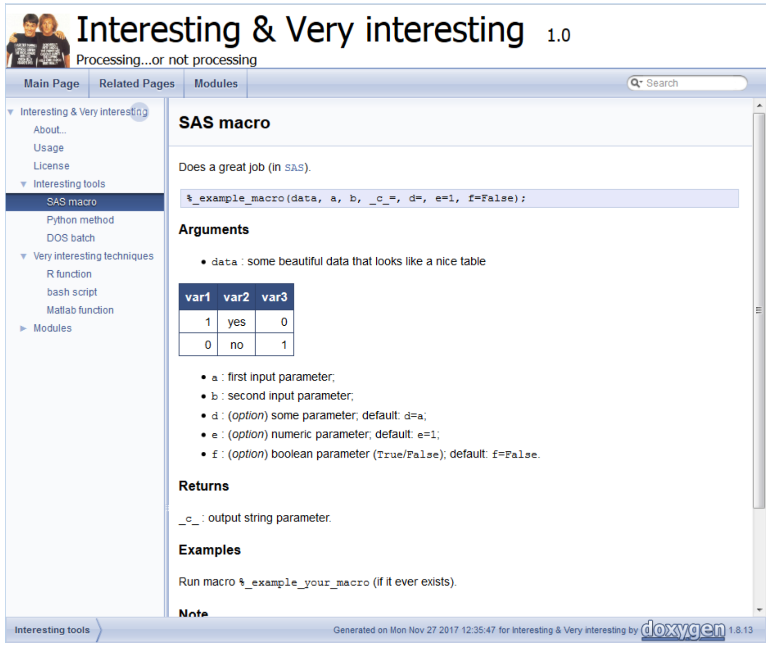
Contents:

- big Module
  - big
- package1 Package
  - small1 Module
    - package1.small1
  - small2 Module
    - package1.small2

Indices and tables

- *Index*
- *Module Index*
- *Search Page*

sphinxy documentation »



Interesting & Very interesting 1.0

Processing...or not processing

Main Page | Related Pages | Modules | Search

- Interesting & Very interesting
  - About...
  - Usage
  - License
  - Interesting tools
    - SAS macro
    - Python method
    - DOS batch
  - Very interesting techniques
    - R function
    - bash script
    - Matlab function
  - Modules

SAS macro

Does a great job (in SAS).

```
%_example_macro(data, a, b, _c_=, d=, e=1, f=False);
```

Arguments

- `data` : some beautiful data that looks like a nice table

var1	var2	var3
1	yes	0
0	no	1

- `a` : first input parameter;
- `b` : second input parameter;
- `d` : (*option*) some parameter; default: `d=a`;
- `e` : (*option*) numeric parameter; default: `e=1`;
- `f` : (*option*) boolean parameter (`True/False`); default: `f=False`.

Returns

`_c_` : output string parameter.

Examples

Run macro `%_example_your_macro` (if it ever exists).

Note

Interesting tools

Generated on Mon Nov 27 2017 12:35:47 for Interesting & Very interesting by doxygen 1.8.13

2° Assignment - Requirements

- The Jupyter notebook that you will upload in your repository will also consistute the report for the assignment. Use markdown cells to describe what you have done, particular instructions for interfacing with the robot, or for visualizing the graph, …

- The final discussion will focus on both assignments, thus general questions about the topics objects of the course (action servers, ROS2, CoppeliaSim, Matplotlib, Jupyter Notebook and software documentation) may be asked.

- There is no deadline for the second assignment: the second assignment (and the first, if you missed the deadline) should be submitted at least 5 days before the oral discussion.