IP.3510 End-of-Track project

Report

PAL'S ARI

Guidance and controlled by

Dr. Patric Wang and Zakia kazi-aoul Associate professor at LISITE (ISEP)

.

Submitted by,

Max-André LOISEL – 9927

Ajay kumar. Chittimilla–60482

Simon TALPE – 9953

INDEX page no

I. Introduction

Present ourselves:

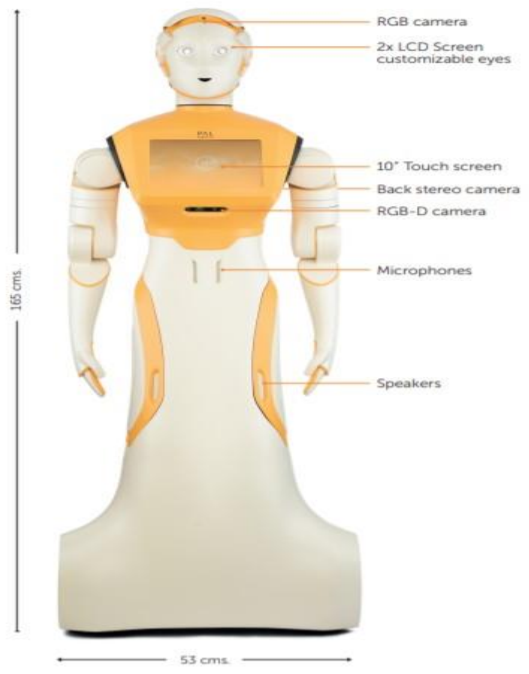
Max-André LOISEL : I was interested by this subject because, as a student in the digital health cursus at ISEP, I wanted to learn about robotic and especially about human – robot interaction. I wanted to develop my skill set in this domain because I hope one day to work on this kind of project applied to the domain of health. It was even more interesting for me because I had never previously had the opportunity to work with robots.

Simon TALPE : As a student in Computer Science, I had the opportunity to work last year on C++ and especially OpenCV. I wanted to deepen my knowledge on this subject: the prerequisites of the project seemed quite adapted to my goal. Moreover, after doing some research on the ARI robot, I found very exciting the prospect of working on it and more particularly on concrete cases like those described in the presentation.

Ajay Kumar CHITTIMILLA : pursing course in embedded system, I have chosen this project as my end track because it helps me to put all my experience that I gained during my course at ISEP. Moreover, this project helps me excel my knowledge in Linux OS and ROS which plays a key role in this developing world.

II. Presentation of Pal's ARI

ARI is a humanoid robot from Pal Robotics designed for Human-Robot interaction, speech recognition, perception, navigation and even manipulation. It is the latest creation from the company and was released in 2020. The model we worked with is one of the very first existing in the world. Its processing power allows to integrate diverse AI algorithms. The possibilities of ARI are almost endless, but its star features are to being able to execute coordinated dual-arm actions, to autonomously navigate in crowded environments and to provide dynamic information. It is possible to use ARI in simple ways with its web interface or to dive deeper and use the extensive **ROS** API.



ROS (Robot Operating System) is an open-source robotics middleware suite that helps you to build your own robot applications or to use already existing ones. ROS serves as the interface for the robot and allows you to use other tools integrated in it like **Gazebo**.

Gazebo is a 3D simulator which allows you to create a 3D scenario with objects, obstacle, different paths and many other possibilities. It is a very powerful tool because it allows you to work remotely on the robot and to test your code before using it on the actual robot.

III. Technical development

This section of the report will be the one that will serve as a “how to use” guide for the futures groups who will get to work on ARI. In it, we will go over the technical task the user will need to do if he wants to work with the robot.

All that is described in this section is the fruit of our experimenting with the robot. Unfortunately, we were not able to do as much as we originally wanted, and that is due in most part to our lack of experience working with such robots, but also that since there were not many people who got to work on this robot, there also was not any answers about potential errors that could be found online. All we had to go with was a very complete but also complex documentation provided by Pal themselves, and as we followed the steps described in it, many errors occurred and we had to figure out a way to fix or work around them without any solution online.

Nevertheless, we were able to figure out some processes and we are going to describe them in this part, in hope that the future groups won’t have to struggle with them and will be able to make it further then us.

To point out the components in short:

- **Flash drive**

  The flash drive consists of robot software it is usable for boot or reinstall the entire software of the pal robot.
  Precautions, once you boot the robot the setting will be reset to back like WIFI and its Password. And for software update, once you connect to the internet via ethernet cable everything will be automatically updated.

- **Joystick**

  A joystick is provided with pal robot which is mainly focused for mapping and navigation. There is no need of any additional setting or configuration for usage of joystick. Once you turn on the robot it should automatically work, for example you can press any keys on right side for head moment.

A) Environment setup

UBUNTU:

Ubuntu Linux is the most popular open source operating system. There are many reasons to use Ubuntu Linux that make it a worthy Linux distro. Apart from being free and open source, it’s highly customizable and has a Software Center full of apps. There are numerous Linux distributions designed to serve different needs.

Instructions to Install Ubuntu Linux 18.04 (LTS) along with Windows :

Back Up Your Existing Data!

This is highly recommended that you should take backup of your entire data before start with the installation process.

Obtaining System Installation Media

Download latest Desktop version of Ubuntu from this link:

http://old-releases.ubuntu.com/releases/18.04.2/ .

Booting the Installation System

There are several ways to boot the installation system. Some of the very popular ways are , booting from a CD ROM, Booting from a USB memory stick, and Booting from TFTP.

Here we will learn how to boot installation system using a USB drive. In order to do that, we first need to have a bootable USB drive with the right version of ubuntu deployed on it. To do that there are a few easily downloadable tools that can do that quickly. To do it, we used RUFUS (https://rufus.ie/). This process takes a few minutes, once it was done, we could start the installation process.

Because we had already worked on our window environment, we had to modify the booting sequence in the window restart option menu, accessible by pressing shift + restart. If all is well, you should see the option to reboot the computer from your flash drive. With that done, the process could begin.

Changing the Boot Order of a Computers

1. As your computer starts, press the **DEL**, **ESC**, **F1**, **F2**, **F8** or **F10** during the initial startup screen. Depending on the BIOS manufacturer, a menu may appear. However, consult the hardware documentation for the exact keystrokes.

2. Find the Boot option in the setup utility. Its location depends on your BIOS. Select the Boot option from the menu, you can now see the options Hard Drive, CD-ROM Drive, removable Devices Disk etc.

3. Change the boot sequence setting so that your bootable flash drive is in first position in the list. You can also delete the windows boot option if you don't think it is needed anymore.

4. save your changes. Instructions on the screen tell you how to save the changes on your computer. The computer will restart with the changed settings.

5. If all is well, the computer should restart with unbuntu as its main OS.

6. We have now successfully installed Ubuntu 18.04 LTS. All that is left to do is to follow the first boot steps. While going through this process, we encountered different errors

that were manly caused by the fact that the USB drive we used didn't have enough space on it. For the process to work, the user needs to have at least 25 Go on his USB drive.

Once the ubuntu works well, we are ready to get set the system to accept the software from third party and setup the ROS for further process.

Configure your Ubuntu repositories

Configure your Ubuntu repositories to allow "restricted," "universe," and "multiverse." You can follow the Ubuntu guide for instructions on doing this.

Setup your sources.list

Setup your computer to accept software from packages.ros.org.

```
$ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'
```

Set up your keys

```
$ sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

Installation

First, make sure your Debian package index is up-to-date:

```
$ sudo apt update
```

install ROS packages individually.

Desktop-Full Install: (Recommended) : ROS, rqt, rviz, robot-generic libraries, 2D/3D simulators and 2D/3D perception

```
$ sudo apt install ros-melodic-desktop-full
```

e.g. for additional

```
$ sudo apt install ros-melodic-slam-gmapping
```

To find available packages, use:

```
$ apt search ros-melodic
```

Environment setup

It's convenient if the ROS environment variables are automatically added to your bash session every time a new shell is launched:

```
$ echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc
```

```
$ source ~/.bashrc
```

If you have more than one ROS distribution installed, ~/.bashrc must only source the setup.bash for the version you are currently using.

If you just want to change the environment of your current shell, instead of the above you can type:

```
$ source /opt/ros/melodic/setup.bash
```

If you use zsh instead of bash you need to run the following commands to set up your shell:

```
$ echo "source /opt/ros/melodic/setup.zsh" >> ~/.zshrc
$ source ~/.zshrc
```

Initialize rosdep

Before you can use many ROS tools, you will need to initialize rosdep. rosdep enables you to easily install system dependencies for source you want to compile and is required to run some core compootprint dynamixel_cpp tf_lookup opencv3 librealsense2-dev librealsense2-dkms hey5_transmissions" -y

Then, you may proceed building the workspace:

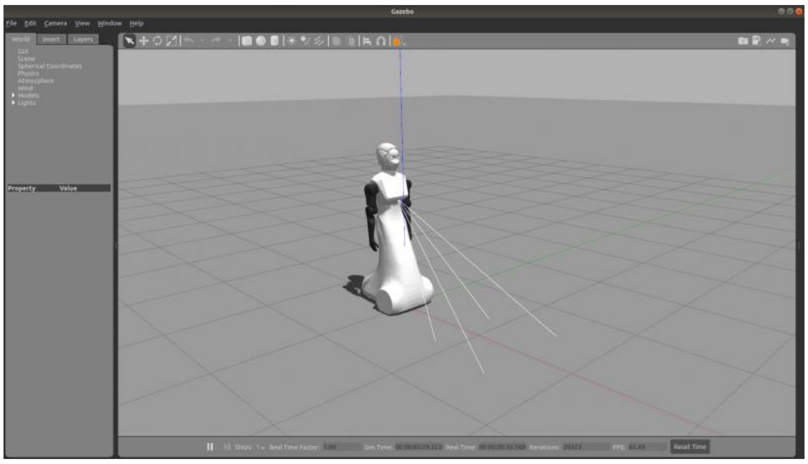
```
$ source /opt/ros/melodic/setup.bash
$ catkin build -DCATKIN_ENABLE_TESTING=0
```

Once you compiled all packages and sourced the environment (source devel/setup.bash) it's all ready to go.

Here you can check them in two ways one

GAZIBO: this is where you can build your own social Virtual AI robot with your system ROS.

```
$ source ./devel/setup.bash
$ roslaunch ari_gazebo ari_gazebo.launch public_sim:=true
```



B) Package creation

Package creation is very useful when working with ARI, it is how the user can make the robot learn new methods and implement new scripts. After creating a package with catkin, it is required to have it deployed to the robot and it can then be used with the terminal command "rosrun". In the part, we will go through those three processes.

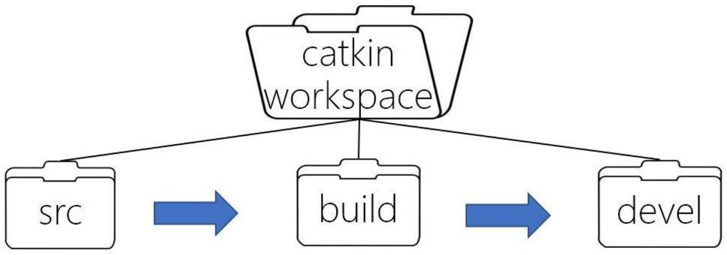
The first step in the creation process is to setup up your workstation while being connected to the robot via ssh. In the documentation, it is said to create the package on your machine and deploy it, but when we tried this method it didn't work. The process that lead to a positive result is the following.

Once the user is connected to the robot, if he hasn't had the opportunity to do it before, he needs to create his workspace and access it using the following commands:

$ mkdir your_ workspace _name_ws

$ cd your_ workspace _name_ws

Once inside the workspace, the user needs to create a directory called "src" using the same command to create the workstation (*$ mkdir src*) and set it up with the command *$ source /opt/pal/melodic/setup.bash* . Once the setup is done, the user needs to access his src directory ($ cd src) and initialize his workspace using catkin: *$ catkin_init_workspace*



Now, we have an initialized workspace on the robot, it will be there that we will create our packages. Up to this point, the user only needs to do this once! The next steps are for the actual package creation and deployment to the machine.

Inside of the src directory, the user needs to create the structure for his package using a method proposed by catkin : *$ catkin_create_pkg your_package_name.* This will create a directory with the name given in the previous command, it contains all the files necessary to the package creation, but the user still needs to create a directory called scripts *($ mkdir scripts)*. It is in this directory that the actual codes of the package will be stored. From the package directory, the user needs to access the scripts directory ($ cd scripts) and create the code file using this command:

if in python : *$ touch script_name.py*

At this point, the user has an initialized workstation and a properly structured package. All that is left is to write the scripts, do a final setup and finish the creation.

To edit the script, the user can either try to find the text file on his computer, or he can use vim to edit it via the terminal (*$ vim script_name.py*) . Vim is a very useful tool to edit text files, but it has its own special commands, so the user needs to make sure he knows the right commands to navigate properly. Here are some vim quick tips: press ":" to enter a command, ":q!" to leave without saving and ":x" to save and leave.

As we were saying, it here that the user can code his script. We couldn't create our own package but there are a lot of them that the user can use online. For example, we tested this process with the script say.py from the package ari_say_something that can be found with this link : https://github.com/pal-robotics/ari_tutorials. We copied the script of say.py and pasted it in the say.py file we had in our package.

After having edited the script, there just a few steps left. The first one is to change the access permissions of the script file using the following command:

$ chmod +x script_name.py (Do not forget the "+x" or the process will not work!)

The package is now ready to go through the last step of the process, the actual making into a usable package. First the user needs to return to the workspace directory. Once in there, the finish the creation, the user needs to use the command:

$ catkin_make

Now, one finale setup is necessary : *$ source ~/your_ws/devel/setup.bash.* And now, all is ready for the user to use his package. The command to call the script is the following:

$ rosrun your_package_name script_name.py your arguments if necessary

A we said, this process is for the creation of a new package. During our experimenting this is the method we found to be the most functional, but like previously mentioned it is not the one specified in the Pal documentation. This leads us to believe that there could be another one, but by using this one, the user is guaranteed to get the result he wants.

a) Rviz

Rviz is the 3d visualization tool used by ROS. In our case, we need Rviz to be able to visualize ARI in its environment as well as to monitor any action that has to do with the different cameras on ARI. Rviz can also be used with gazebo by simulating the different data required.

Since Rviz is a mor commonly used tool, we were able to find more solutions online regarding how to download it. In order to have a functioning version of Rviz, we followed the following steps:

To download rviz, since we are using ROS melodic, to command line is:

$ sudo apt-get install ros-melodic-rviz

And that is it, much faster than the package creation process. Now to start it up while having it linked to our robot, we followed those steps:

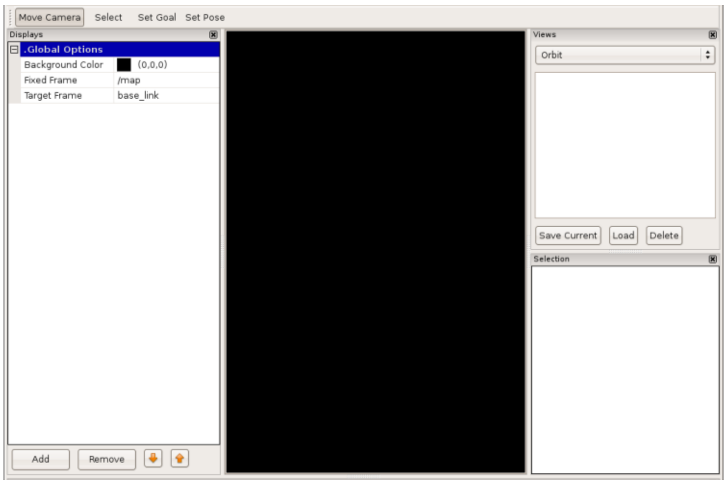
First, we need to source it:

```
$ source /opt/ros/melodic/setup.bash
```

Then to run it:

```
$ roscore &
$ rosrun rviz rviz
```

Then, this should start up the Rviz interface:



.Global Options

Background Color	■ (0,0,0)
Fixed Frame	/map
Target Frame	base_link

In the next paragraph, we will see how to link Rviz to our robot.

b) Mapping

Mapping is one of Ari's most interesting functionality. It is the one that we wanted to do since the beginning, and we managed to make it a few steps down the road. In this section, we will be able to give the reader an example of the kind of struggle we faced with every task we tried to do on this project.

To start the mapping procedure, we needed to launch Rviz while it was linked to the robot. In order to do that, it is once again with the help of our trusty terminal. The steps are the following:

First, connect to the ssh with this command line :

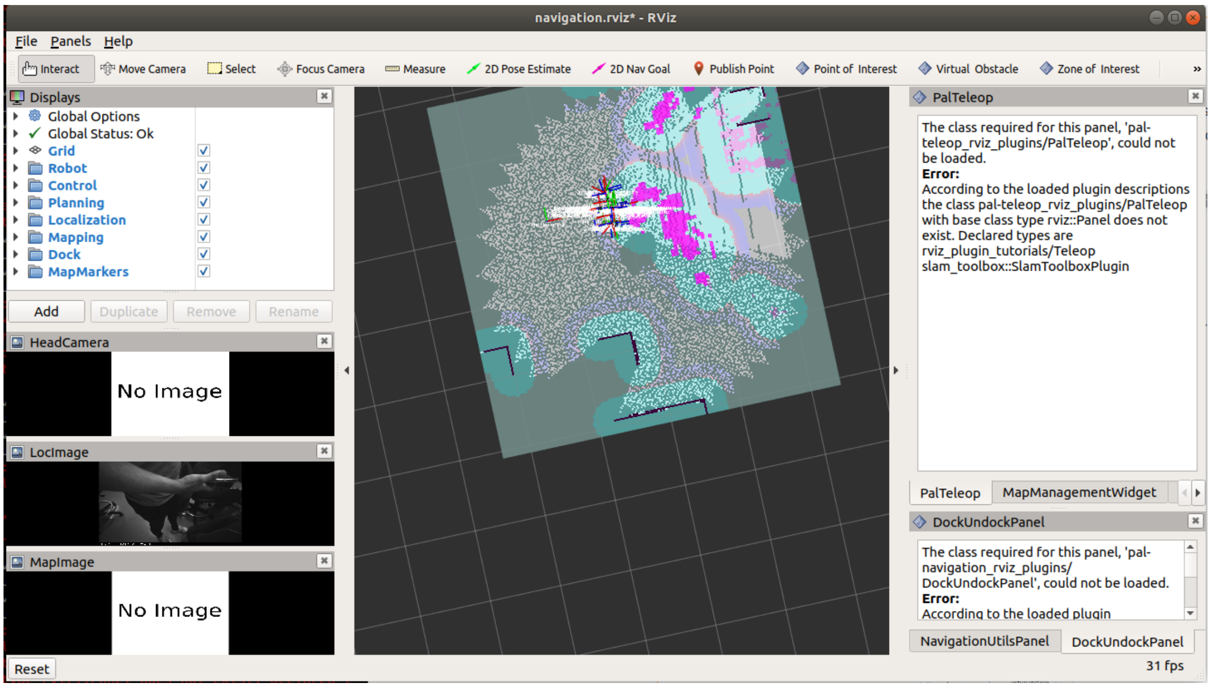
```
$ ssh -X pal@10.68.0.1 or $ ssh -X pal@ari-4c
```

The "-X" is very important. Without it, Rviz will not be able to start and the user will get an "X display" error.

After that, we need to access tour workspace (*$ cd ~/your_ws*) and source it (*$ source /opt/ros/melodic/setup.bash* and *$ roscore &*) .Finally, we need to link the robot before launching Rviz. To do so, we enter the command lines :

> *$ export ROS_MASTER_URI=http://ari-4c:36587*
> *$ export ROS_IP=10.68.0.1´´ ``rosrun rviz rviz -d `rospack find ari-4c`/config/rviz/navigation.rviz*

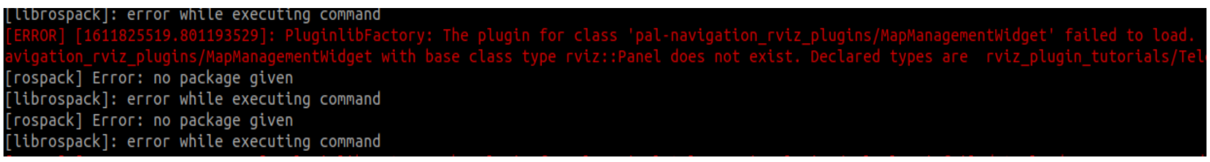
After doing that, Rviz launchs and is linked to our robot, victory is ours!



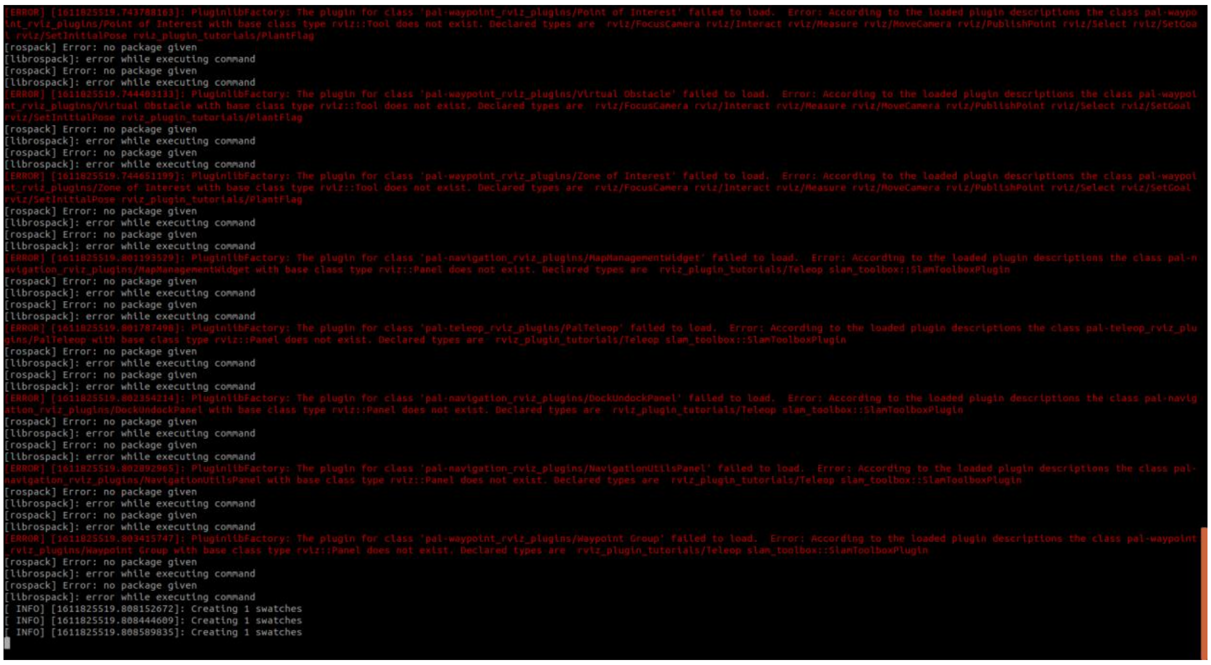
The class required for this panel, 'pal-teleop_rviz_plugins/PalTeleop', could not be loaded.
Error:
According to the loaded plugin descriptions the class pal-teleop_rviz_plugins/PalTeleop with base class type rviz::Panel does not exist. Declared types are rviz_plugin_tutorials/Teleop slam_toolbox::SlamToolboxPlugin

The class required for this panel, 'pal-navigation_rviz_plugins/DockUndockPanel', could not be loaded.
Error:
According to the loaded plugin

Actualy, no. Because, for the mapping process to be launched, we need to access the MapManagementWidget. Unfortunately, as can be seen in the picture above, the widget is inaccessible because we are missing the necessary plugin. When we look at the terminal, we get a better idea of the issue:



```
[librospack]: error while executing command
[ERROR] [1611825519.801193529]: PluginlibFactory: The plugin for class 'pal-navigation_rviz_plugins/MapManagementWidget' failed to load.
avigation_rviz_plugins/MapManagementWidget with base class type rviz::Panel does not exist. Declared types are  rviz_plugin_tutorials/Tel
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
```



```
[ERROR] [1611825519.743788163]: PluginlibFactory: The plugin for class 'pal-waypoint_rviz_plugins/Point of Interest' failed to load.  Error: According to the loaded plugin descriptions the class pal-waypoint_rviz_plugins/Point of Interest with base class type rviz::Tool does not exist. Declared types are  rviz/FocusCamera rviz/Interact rviz/Measure rviz/MoveCamera rviz/PublishPoint rviz/Select rviz/SetGoal rviz/SetInitialPose rviz_plugin_tutorials/PlantFlag
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
[ERROR] [1611825519.744403133]: PluginlibFactory: The plugin for class 'pal-waypoint_rviz_plugins/Virtual Obstacle' failed to load.  Error: According to the loaded plugin descriptions the class pal-waypoint_rviz_plugins/Virtual Obstacle with base class type rviz::Tool does not exist. Declared types are  rviz/FocusCamera rviz/Interact rviz/Measure rviz/MoveCamera rviz/PublishPoint rviz/Select rviz/SetGoal rviz/SetInitialPose rviz_plugin_tutorials/PlantFlag
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
[ERROR] [1611825519.744651199]: PluginlibFactory: The plugin for class 'pal-waypoint_rviz_plugins/Zone of Interest' failed to load.  Error: According to the loaded plugin descriptions the class pal-waypoint_rviz_plugins/Zone of Interest with base class type rviz::Tool does not exist. Declared types are  rviz/FocusCamera rviz/Interact rviz/Measure rviz/MoveCamera rviz/PublishPoint rviz/Select rviz/SetGoal rviz/SetInitialPose rviz_plugin_tutorials/PlantFlag
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
[ERROR] [1611825519.801193529]: PluginlibFactory: The plugin for class 'pal-navigation_rviz_plugins/MapManagementWidget' failed to load.  Error: According to the loaded plugin descriptions the class pal-navigation_rviz_plugins/MapManagementWidget with base class type rviz::Panel does not exist. Declared types are  rviz_plugin_tutorials/Teleop slam_toolbox::SlamToolboxPlugin
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
[ERROR] [1611825519.801787498]: PluginlibFactory: The plugin for class 'pal-teleop_rviz_plugins/PalTeleop' failed to load.  Error: According to the loaded plugin descriptions the class pal-teleop_rviz_plugins/PalTeleop with base class type rviz::Panel does not exist. Declared types are  rviz_plugin_tutorials/Teleop slam_toolbox::SlamToolboxPlugin
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
[ERROR] [1611825519.802354214]: PluginlibFactory: The plugin for class 'pal-navigation_rviz_plugins/DockUndockPanel' failed to load.  Error: According to the loaded plugin descriptions the class pal-navigation_rviz_plugins/DockUndockPanel with base class type rviz::Panel does not exist. Declared types are  rviz_plugin_tutorials/Teleop slam_toolbox::SlamToolboxPlugin
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
[ERROR] [1611825519.802892965]: PluginlibFactory: The plugin for class 'pal-navigation_rviz_plugins/NavigationUtilsPanel' failed to load.  Error: According to the loaded plugin descriptions the class pal-navigation_rviz_plugins/NavigationUtilsPanel with base class type rviz::Panel does not exist. Declared types are  rviz_plugin_tutorials/Teleop slam_toolbox::SlamToolboxPlugin
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
[ERROR] [1611825519.803415747]: PluginlibFactory: The plugin for class 'pal-waypoint_rviz_plugins/Waypoint Group' failed to load.  Error: According to the loaded plugin descriptions the class pal-waypoint_rviz_plugins/Waypoint Group with base class type rviz::Panel does not exist. Declared types are  rviz_plugin_tutorials/Teleop slam_toolbox::SlamToolboxPlugin
[rospack] Error: no package given
[librospack]: error while executing command
[rospack] Error: no package given
[librospack]: error while executing command
[ INFO] [1611825519.808152672]: Creating 1 swatches
[ INFO] [1611825519.808444609]: Creating 1 swatches
[ INFO] [1611825519.808589835]: Creating 1 swatches
```

This is a good example of the errors we commonly faced during this project. This error seems to be caused by the inexistence of the plugins and when we looked online for a solution, all we found was one comment saying that those kind of package are reserved to buyers of pal products and not accessible to anyone. But we had the robot and therefore should have access to this content.

Sadly, this is as far as we could make it regarding the mapping. This was a huge personal disappointment because we really saw this functionality as the goal we had to reach during our time working with ARI, but that feeling has been present on every objectives we set ourselves, which was really hard.

In our opinion, a solution to this recuring issue could be to have a contact working at Pal that could help us trough the first steps before letting us work on our own, just to have a possibility to fix some errors and not have to give up and get to another objective after multiple hours of unfruitful work. For example, in the case of the mapping, we were sure that the issue was something very trivial, but after looking for a long time for a solution, we still couldn't do it, but maybe someone at Pal could have shown us where we got it wrong.

IV. Retrospective on our project experience

The beginning of the project has been quite exciting. We researched the robot's capabilities in more depth and saw it for real. We thought about what we could do with it in a semester and thought we could go far in exploration. In addition, the tutorials for setting up the environment seemed well done and easy to follow. However, we were very quickly disappointed: we realized that we were encountering problems at almost every step, from installing the right version of Ubuntu to the catkin packages, without forgetting the installation of ROS. We felt like we were not moving forward at all, and for every problem we solved, a bigger one was standing in front of us. Moreover, these problems were not at all motivating to deal with, since it was only Linux command line and environment installation, combined with the fact that there was very little documentation available. At the time, all three of us were very discouraged. When we realized that working on a virtual machine was causing bugs that we couldn't solve, we started from scratch and changed the computer system to Ubuntu directly (instead of Windows and VirtualBox). From then on, based on what we had already learned and thanks to the better performance of the system, we were able to move faster and make new advances (such as installing rviz). However, the semester was already coming to an end and we could not continue our good momentum. So we decided to prepare a kind of "ARI quick start guide" for the students who will take over the project next year and prevent them from being stuck by the same problems as us.

In conclusion we ended this project on a positive note, but not enough to erase the frustration of not having achieved much in the way of concrete results.