Robots and IoT devices for assistive automation

Internship report

Institut de Robòtica i Informàtica Industrial

Kévin Bedin - SPID - Robotic

10 june 2019 - 30 august 2019

Acknowledgements

Thanks to Guillem Alenyà Ribas and Sergi Foix Salmerón for their welcome, support and trust throughout this project.

Résumé

Depuis quelques années, nous vivons chaque jour dans un monde toujours plus connecté. De ce phénomène se sont développées les technologies de l'Internet of Things (IoT). Parallèlement, les progrès dans la robotique permettent aujourd'hui d'avoir des robots de services autonomes dans leur mobilité et dans l'accomplissement de leurs missions. La question qui se pose donc, à ce jour, est de savoir s'il pourrait être utile d'unir ces deux domaines.

Tout comme l'être humain le fait, comment permettre à ces robots de service d'intégrer les objets de l' IoT à leur environnement afin de faciliter l'accomplissement de leur mission ?

Ce présent rapport s'intéresse donc à la manière dont il est possible d'intégrer les objets de l'IoT dans l'environnement d'un robot de l'entreprise PAL Robotics, TIAGo. Il est le fruit de douze semaines de stage au sein du laboratoire de Perception et de Manipulation de l'Institut de Robòtica i Informàtica Industrial de Barcelone. Il est ainsi étudié dans ce rapport la faisabilité, l'implémentation et l'utilisation d'une telle intégration. Le laboratoire dispose d'un appartement dans lequel deux robots TIAGo évoluent. Ainsi l'objectif principal de mon stage était de domotiser cet appartement et de rendre accessible les différents appareils de l'IoT aux robots. Pour ce faire, j'ai dans un premier temps effectué un état de l'art en ce qui concerne les logiciels de domotiques opensource. Mon choix s'est porté sur un logiciel très populaire dans le monde de l'IoT: OpenHAB (OH). Ce logiciel agit comme un hub, pouvant utiliser la plupart des protocoles domotiques actuellement utilisés. J'y ai ainsi intégré différents capteurs et actionneurs du monde de la domotique ainsi qu'une caméra IP. Par la suite je me suis penché sur le moyen de relier l'environnement du robot à OpenHAB. C'est au travers du middleware ROS et du package *iot_bridge*, que j'ai amélioré pour cette implémentation, que cette liaison a été mise en place. Enfin j'ai développé des algorithmes pour la camra, sous la forme d'un package ROS. Le principal algorithme permet de scanner l'appartement à la recherche d'une image binaire (Aruco) et de fournir sa localisation dans l'appartement. Enfin, une mission consistant à trouver un Aruco a été implémentée dans le robot. Pour ce faire il peut utiliser à la fois ses propres caméras mais aussi utiliser la caméra IP grâce au package développé.
L'accomplissement de cette mission a permis de montrer qu'il est actuellement possible d'intégrer le monde de l'IoT au sein de l'environnement d'un robot de service mais aussi d'en montrer l'utilité.

L'ensemble de mon travail est également disponible sous la forme d'un rapport technique (1) qui a donné lieu à une publication sous la référence IRI-TR-19-03, publié par le CSIC-UPC sur le site de l'institut.

Abstract

In recent years, we live every day in an ever more connected world. This phenomenon has developed the technologies of the Internet of Things (IoT). At the same time, advances in robotics now make it possible to have autonomous service robots in their mobility and in the accomplishment of their missions. So far the question is whether it might be useful to unite these two areas.

Just as humans do, how do these service robots integrate the objects of IoT into their environment to facilitate the fulfillment of their mission?

This report focuses on how it is possible to integrate IoT objects into the environment of a PAL Robotics robot, TIAGo. It is the result of twelve weeks of internship within the laboratory of Perception and Manipulation of the Institut de Robòtica i Informàtica Industrial of Barcelona. It is thus studied in this report the feasibility, the implementation and the use of such an integration. The laboratory has an apartment in which two TIAGo robots evolve. So the main objective of my internship was to domotise this apartment and make available to robots the different devices of the IoT. To do this I first made a state of the art regarding opensource home automation software. My choice fell on a very popular software in the world of IoT: OpenHAB (OH). This software acts as a hub, being able to use most of the domotic protocols currently used. I have integrated various sensors and actuators from the world of home automation as well as an IP camera. Subsequently I looked at how to connect the environment of the robot to OpenHAB. It is through the ROS middleware and the *iot_bridge* package, that I have upgraded for this implementation, that this link was set up. Then I developed algorithms, in the form of a ROS package, for the camera. The main algorithm allows to scan the apartment in search of a binary image (Aruco) and provide its location in the apartment. Finally a mission to find an Aruco has been implemented in the robot. To do this it can use both his own cameras but also use the IP camera with the developed package.
The accomplishment of this mission has shown that it is currently possible to integrate the world of IoT within the environment of a service robot but also to show the utility.

All of my work is also available in the form of a technical report (1) which resulted in a publication under the reference IRI-TR-19-03 published by the CSIC-UPC on the site of the institute.

Contents

1 Introduction **1**

2 Context **2**
2.1 Laboratory presentation . . . 2
2.2 Job description . . . 2
2.3 Available material . . . 2

3 Problem definition **3**
3.1 Problematisation . . . 3
3.2 State of the art . . . 3
3.2.1 OpenHAB - Centralize the IoT devices . . . 3
3.2.2 ROS - Communicate with TIAGo . . . 4
3.2.3 iot_bridge - Link ROS and OpenHAB . . . 5
3.2.4 Choosing an IP Camera . . . 5
3.2.5 Planned strategies . . . 5

4 Development **6**
4.1 Rapsberry Pi . . . 6
4.2 OpenHAB . . . 6
4.2.1 Sensors and actuators integration . . . 6
4.2.2 Data saving . . . 6
4.2.3 Voice integration . . . 7
4.2.4 iot_bridge . . . 7
4.3 IP Camera . . . 8
4.3.1 Redefine the strategy . . . 8
4.3.2 Manage the camera . . . 8
4.3.3 ROS integration . . . 9
4.3.4 Meet the missions . . . 10
4.3.4.1 PTZ controller . . . 10
4.3.4.2 Place saving . . . 11
4.3.4.3 Calibration . . . 12
4.3.4.4 Aruco detection . . . 13
4.3.4.5 Scan algorithm . . . 14
4.3.5 OpenHAB integration . . . 16
4.4 Global view . . . 16

5 Tests and results **18**
5.1 Simulations . . . 18
5.1.1 Pan controller simulator . . . 18
5.1.2 Location simulator . . . 18
5.2 Integration tests . . . 19
5.2.1 Wallplug test . . . 19
5.2.2 Scan routine . . . 20
5.3 Mission of Aruco detection . . . 22

6 Conclusion **24**
6.1 About the project . . . 24
6.2 Personal . . . 24

Bibliography **25**

List of figures **26**

Chapter 1

Introduction

Today in 2019, the world of the Internet of Things (IoT) sees itself as the technological industry of tomorrow. The number of connected objects increased from 3.8 billion in 2015 to 8.3 billion in 2019, with a prediction of more than 20 billion within 6 years, in 2025 (2). This is understandable by the progress in the speed of communication networks allowing ever larger data flows in ever shorter times.
Along with this development, the robotic industry is growing. In 2017 the service robotics market was shared by 700 companies, including 29% of start-ups. It is estimated that the period 2018-2020 will grow by 20 to 25% in the professional service robotics market and that the dedicated consumer market will reach a value of approximately $ 11 billion over the same period (3).
So these two sectors being on the rise, would not it be the right moment to unite them? Could we not allow service robots to use IoT objects so that it can fulfill its missions more easily, as a human being does? Would not that make them more efficient?
This report is limited to the implementation of IoT objects in the environment of a TIAGo service robot. The goal is to know how to integrate its objects within the robot as well as to show through a mission the usefulness of such an implementation. Finally, being in a domain dedicated to individuals, the proposed solutions as well as those used must be open source. It is the result of twelve weeks of internship within the labory of perception and manipulation of the Institut de Robòtica i Informàtica Industrial of Barcelona.

Chapter 2

Context

2.1 Laboratory presentation

I did my internship in the laboratory of Barcelona: Institut de Robòtica i Informàtica Industrial. It is a laboratory founded in 1995 under the alliance of Technical University of Catalonia (UPC: Universitat Politecnica de Catalunya) and Consejo Superior de Investigaciones Científicas (CSIS) which is the equivalent of CNRS in France.

It is divided into four research areas: Automatic control, Kinematics and robot design, Mobile robotics and Intelligent Systems, Perception and Manipulation. It is within this last pole that I did my internship. The team is made up of around forty people, the majority of whom are students in the final year of study or doing their thesis. In order to carry out their mission are available: two TIAGo service robots from the company PAL Robotics, various robotic arms and an experimental apartment. The most important area of research is the manipulation of textiles by robots.

2.2 Job description

To better understand the context of the work that was asked to me, here is the job description of the internship for which I applied and selected, entitled "Robots and IoT devices for assistive automation":

> "The Perception and Manipulation Group investigates in service robotics and has two TIAGo mobile manipulation robots and a laboratory simulating an apartment. This project wants to investigate the benefits of adding smart devices to this environment and making them available to the robot. In recent years, a wide variety of IoT (Internet of Things) domestic devices have appeared that can monitor environmental variables and control active parts, as well as several initiatives that try to standardize protocols and access methods to these devices. On the other hand, service robotics has advanced enormously thanks to the existence of robust solutions for critical parties, such as voice recognition or navigation avoiding obstacles. Now is the propitious moment to unite these two worlds and interconnect the robot with the environment. The robot would not only multiply its action capabilities but also increase its data acquisition capacity. For example, if you do not have enough information with your own sensors and cameras or it is ambiguous, you could connect to the surrounding network of cameras and obtain images from different points of view and places."(4)

So the purpose of my internship was to know if it was possible to integrate in the Lab's apartment smartdevices using various protocols within the environment of a TIAGo robot so that it can use it to enrich its range of actions.

2.3 Available material

In order to carry out the project I had access to:

- 2 Wallplugs from Fibaro company (IoT device)
- 1 Multisensor from Fibaro company (IoT device)
- 1 Motion sensor from Aeotec company (IoT device)
- 1 Z-wave plus stick from Aeotec company (IoT device)
- 1 Raspberry pi 3 B+
- 1 Amazon echo dot
- 2 robots TIAGo
- 1 IP Camera to choose
- A Gitlab workspace

Chapter 3

Problem definition

3.1 Problematisation

My first week of internship was to define the subject. It was necessary that I highlight various problems in order to better understand the expectations of the project as well as the exploitable technical solutions. The first problem was:

- How to use these sensors ?

Indeed before being able to connect it to the robots, you must know how to extract the information. The question that arises if therefore to know if you can use a single tool that can integrate a wide variety of IoT protocols, ie the existence of a hub for connected objects.

- Is it possible to standardize these protocols ?

After that it would be a question of how the robot interacts with its environment:

- What environment does the robot use ?

And then of course it would be necessary to know how to connect the world of IoT via a potential hub and the environment interpretable by the robot:

- How to integrate these sensors into the robot environment ?

Finally, in order not to lose any functionalities as well on the side of the robot as on the side of the connected objects, the last question is to know if these two systems can evolve independently while having a capacity of communication:

- Is it possible to decouple the IoT part and the robot while having an activatable link if necessary ?

So, after having arisen this set of issues, I was able to focus on researching existing technical solutions by making a state of the art.

3.2 State of the art

3.2.1 OpenHAB - Centralize the IoT devices

This part of the state of the art aims to answer the first two issues:

- How to use these sensors ?
- Is it possible to standardize their protocol ?

The world of IoT being born by definition on the Internet, it is relatively easy to find many software allowing to use various protocols of the IoT as well as having a user-friendly Human Machine Interface (HMI) (5). The main criterion is of course the fact that the software is free and open source. Here is the list of some of these software on which I learned:

Name	Origin	Main language
Domoticz	Netherlands	C++
Home assistant	USA	Python3
OpenHAB	Germany	Java
Jeedom	France	PHP

Table 3.1: Home automation popular software

All these software are free, open source and allow to use the different protocols that the laboratory could use (Zwave, MQTT, Zigbee, HTTP, WIFI, MI, Bluetooth, ...). However, there are two important criteria in choosing one for a professional implementation: viability and reliability. These two criteria can be measured by the community supporting the software:

- **Home Assistant**: The project with the largest community. Integrations are added weekly to the software. The daily activity in the official forum is very high, giving great support to the platform. Questions are solved, solutions are provided to the errors that arise and their own projects are shared. In addition, the updates are constant, every little time (one or two weeks) new trips come out with improvements, corrections and new component integrations.
- **Domoticz**: Characterized by consuming very few system resources, it make a very interesting solution if you want to combine it with a Low Cost controller such as a Raspberry Pi. The different modules that compose it are little decoupled from the "core" of the system, that has the main impact of a complicated development of new modules and functionalities. Moreover its documentation in C++ is not very detailed. This is now causing a small loss in the community that redirects to newer and more active solutions, but continues to have a good number of integrations.
- **Jeedom**: Is a much more modern French software in terms of its approaches and that has became a relatively well-known solution within the European home automation market. Very popular in France, its international community is still limited. Although of a more modern appearance and with a much more modular internal architecture, its youth means that we are not faced with a solution that is robust enough to be implemented in a system for professional use.
- **OpenHAB**: The visual design is relatively modern and the documentation is the most complete of those consulted, both at the Core level and the accessory modules. It has a responsive design that adapts the layout of the elements of the control panels to the size of the screen, which facilitates its consumption and cross-platform visualization. It also has official applications for iOS and Android. Finally, its community is as important as that of Home Assistant.

The natural choice is therefore between OpenHAB and Home Assistant. I preferred to rely on OpenHAB because it offers better stability than Home Assistant. Indeed, OpenHAB plugins are subjected to heavier protocols that ensure their smooth operation once made available on the official platform. However, OpenHAB requires more hardware resources than Home Assistant and integrations of new protocols take longer to implement. Being in a context of a semi professional use, the choice of stability prevailed.

3.2.2 ROS - Communicate with TIAGo

This part of the state of the art consists in answering the following problematic:

- What environment does the robot use ?

PAL Robotics has implemented the ROS middleware in their robot. As its name suggests, ROS (Robot Operating System) is an operating system for robots (6). It provides services close to an operating system (hardware abstraction, management of competition, processes ...) but also high-level functionalities (asynchronous calls, synchronous calls, centralized data database, parameterization system robot etc). ROS has 5 features that make it a must for the robotics of tomorrow:

- **Peer to Peer**: Allows different computing units to communicate on the same Ethernet network, synchronously or asynchronously as needed.
- **Multi languages**: ROS is neutral from a language point of view and can be programmed in different languages. The ROS specification intervenes at the message level. Peer to peer connections are negotiated in XML-RPC that exists in a large number of languages.
- **Tool-based**: Every command is actually an executable. The advantage of this solution is that a problem on an executable does not affect the others, making the system more robust and scalable than a system based on a centralized runtime.
- **Lightweight**: In order to fight against the development of algorithms more or less related to the robotic OS and therefore difficult to reuse, the developers of ROS want the drivers and other algorithms to be contained in independent executables.
- **Free and open source**: ROS has a large community that regularly adds new packages.

3.2.3 iot_bridge - Link ROS and OpenHAB

The problem underlying this part is:

- How to integrate the hub sensors into the robot environment ?

In other words, it is about how to link ROS and OpenHAB. After some research, I could find a ROS package doing this task: *iot_bridge*. This package written in python uses the OpenHAB JSON database accessible on the local network to read and write the data of the various integrated devices. This data is then distributed via topics on the ROS middleware.

3.2.4 Choosing an IP Camera

Having chosen OpenHAB as an IoT controller, the choice of the camera was made in the interests of software integration. An plugin, in development but usable, allows to connect some IP camera to OpenHAB2. Here is the list of cameras tested and approved by this plugin:

Brand	Amcrest	Dahua	Foscam	Hikvision	Instar
Model	IP2M-841EW	IPC-HDBW4433R-AS	FI9831W	DS-2CD2385FWD-I	IN-8015 Full HD
	IP2M-841B	IIPC-HFW4431R-Z	FI9821P	DS-2CD2042WD-I	
	IP2M-844	IPC-HDW4421E-AS	FI9900P	DS-2CD2142FWD-IWS	
	IP3M-943B	IPC-HDW2431R-ZS	Fosbaby P1	DS-7208HUI-K2	
	IP8M-2493EW	DH-SD22404T-GN PTZ	C1 Lite	DS-7208HQHI-F1 / N	
			C1	DS-2CD2383G0-I	
			C2	DS-7616NI-K2 / 16P	
				DS-2DE3304W-DE	

Table 3.2: IP Camera used by the OpenHAB IP Camera plugin

Having not found a more robust solution already existing, I chose a cheap camera in terms of price / quality ratio to perform integration tests. The camera we bought is an Amcrest IP3M-941W IP camera at a cost of $ 75. The features of the camera are available in the annex.

The part of the state of the art upstream being completed a technical solution can then be considered.

3.2.5 Planned strategies

In order to keep the IoT part and the robot independent, the planned technical solution is to install OpenHAB2 and ROS on the Raspberry Pi available, that will connect to a network shared with the robot.

The deal is now to find which image should be used for Rapsberry. The choice was to turn to an image of Ubuntu 16.04.6 LTS (GNU / Linux 4.14.98-v7 + armv7l) with already integrated ROS: Ubiquity Robotics Raspberry Pi Image, image which I already worked with. OpenHAB would then be installed as software and not via the image for Rapsberry: OpenHABian.

All IoT devices (sensors, actuators, Alexa, IP camera, ...) would be connected to the Raspberry via OpenHAB. The data will then be communicated to ROS internally via *iot_bridge* and transmitted to the Robot, on the local network, if necessary via ROS either by exporting the ROS `MASTER` or in `MULTIMASTER`.

Chapter 4

Development

4.1 Rapsberry Pi

As planned during the planning of the solutions, the installation of the Ubiquity image and the OpenHAB version 2.4 software was done well on the Raspberry Pi 3 B +.
In order to increase its IoT protocol use capabilities, a Z-wave plus stick was integrated and the Raspberry was connected to a local network via a router.

4.2 OpenHAB

4.2.1 Sensors and actuators integration

The first step was to integrate the Zwave sensor available (the two wallplugs, the motion sensor and the multi sensors) in OH2. Using `Paper UI` and with the Z-wave stick connected, the detection and the integration of the `Things` is almost automatic. You only need to put the devices in paired mode and let the `Inbox` menu discover it. Once added, you can manage the associated `Item` and display it then manage it.

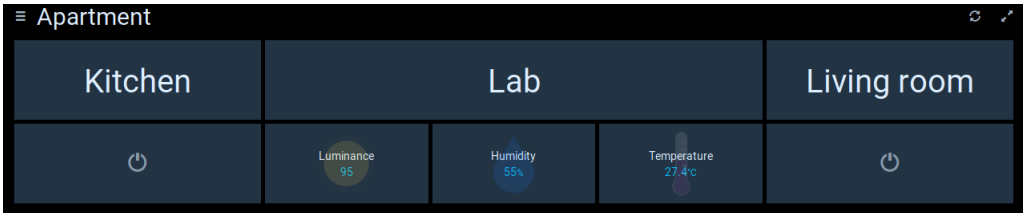


Figure 4.1: HAB Panel screenshot

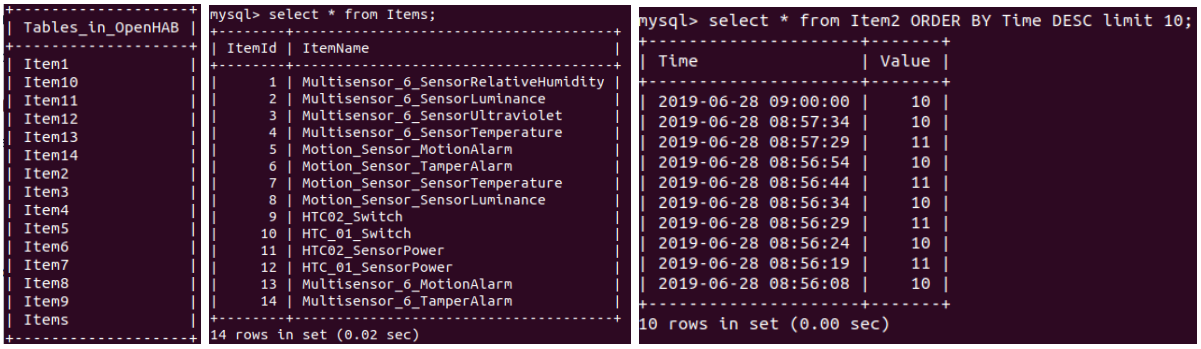
The second step was to add other IoT devices provided by Xiaomi, connected to the Xiaomi MI Gateway that is connected to a WiFi network. Recently the company disabled the port `9898` that allowed the Xiaomi MI Gateway devices to communicate with external devices. So, I had to hack the device to enable it. The OH community being important, it was easy to find a solution on the main forum.
The solution was to solder an `UART` device to the Xiaomi device and send a command to activate the Gateway. The pins on which the `UART` was soldered are pins allowing the manufacturer to configure the micro controller. The correct baudrate for the communication is `115200` and the command is:

```
1     psm-set network.open\_pf 3
```

Once the manipulation done, you can manage the Xiaomi devices through OpenHAB2 using the corresponding binding.

4.2.2 Data saving

The sensors and actuators now integrated into the OpenHAB2 software, it seems so worthwhile to save the sensor data so that they can be used by the robot if necessary. To do this I created a project on the gitlab of the laboratory: **OpenHab-MySQL**.
The MySQL persistence is an OpenHAB plugin that allows you to save data from Items. So I created a configuration that records all sensor and actuator data associated with the ROS group at each new data and every X minutes.



```
+-------------------+
| Tables_in_OpenHAB |
+-------------------+
| Item1             |
| Item10            |
| Item11            |
| Item12            |
| Item13            |
| Item14            |
| Item2             |
| Item3             |
| Item4             |
| Item5             |
| Item6             |
| Item7             |
| Item8             |
| Item9             |
| Items             |
+-------------------+
```

```
mysql> select * from Items;
+--------+-------------------------------------+
| ItemId | ItemName                            |
+--------+-------------------------------------+
|      1 | Multisensor_6_SensorRelativeHumidity |
|      2 | Multisensor_6_SensorLuminance       |
|      3 | Multisensor_6_SensorUltraviolet     |
|      4 | Multisensor_6_SensorTemperature     |
|      5 | Motion_Sensor_MotionAlarm           |
|      6 | Motion_Sensor_TamperAlarm           |
|      7 | Motion_Sensor_SensorTemperature     |
|      8 | Motion_Sensor_SensorLuminance       |
|      9 | HTC02_Switch                        |
|     10 | HTC_01_Switch                       |
|     11 | HTC02_SensorPower                   |
|     12 | HTC_01_SensorPower                  |
|     13 | Multisensor_6_MotionAlarm           |
|     14 | Multisensor_6_TamperAlarm           |
+--------+-------------------------------------+
14 rows in set (0.02 sec)
```

```
mysql> select * from Item2 ORDER BY Time DESC limit 10;
+---------------------+-------+
| Time                | Value |
+---------------------+-------+
| 2019-06-28 09:00:00 |    10 |
| 2019-06-28 08:57:34 |    10 |
| 2019-06-28 08:57:29 |    11 |
| 2019-06-28 08:56:54 |    10 |
| 2019-06-28 08:56:44 |    11 |
| 2019-06-28 08:56:34 |    10 |
| 2019-06-28 08:56:29 |    11 |
| 2019-06-28 08:56:24 |    10 |
| 2019-06-28 08:56:19 |    11 |
| 2019-06-28 08:56:08 |    10 |
+---------------------+-------+
10 rows in set (0.00 sec)
```

Figure 4.2: Database from MySQL Persistence

This database being local, I made it accessible to all users of the local network by using the following command lines:

```
sudo mysql -u root -p
mysql> CREATE USER 'new_user' @ 'IP_address' IDENTIFIED BY 'password';
mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON OpenHAB. * TO 'new_user' @ 'IP_address';
mysql> FLUSH PRIVILEGES;
```

At the end of this first project, the data acquired via OpenHAB are now saved in the OpenHAB local MySQL database and accessible from all users of the network subject to having the identifier. By this way, the robot increased its sensor data and it can acquire information from devices that are present in the apartment.

4.2.3 Voice integration

An Echo Dot being installed on the head of one of the TIAGo, it seemed interesting to us to integrate it into the developing IoT system. This desire resulted in a second project - **alexa_openhab** - available on the gitlab of the laboratory.

The integration of the Alexa voice to IoT objects linked to OpenHAB2 is done in four steps:

- Connecting the local OpenHAB server to the online server `myopenhab.org`
- Login amazon account associated with the echo dot to the online server `myopenhab.org`
- Labeling of OpenHAB Items to be recognized by Alexa
- Detection of labeled items via the amazon account

Once these four steps are done, it is possible to interact with the objects connected via Alexa. Here are some sentences that can be used:

- `Alexa, switch on the Kitchen.`
- `Alexa, what is the temperature in the Lab ?`

4.2.4 iot_bridge

Although functional, this package is based on version 1.0 of OpenHAB - based only on the concept of `Item` (an actuator or sensor) - that became almost obsolete in favor of version 2.0 - that incorporates the concept of `Thing` (a device) on which several `Items` are connected.

In order to better meet the needs of the laboratory and to be consistent with the version of OpenHAB used (2.4), I therefore upgraded this package creating a new project: **iot_bridge_upgrade**.
I integrated the notion of `Thing` and a diagnostic topic to monitor all devices via *rqt_robot_monitor*.

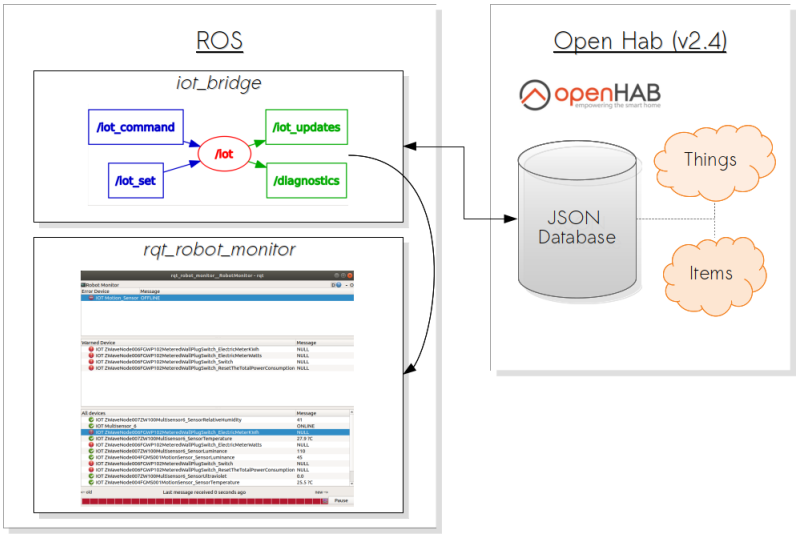


Figure 4.3: Overview diagram of the **iot_bridge_upgrade** package

- */iot_command* (diagnostic_msgs/`KeyValue`)
  When iot_bridge receives a name/value pair from the ROS iot_command topic, it publishes those to OpenHAB and OpenHAB sends that command to the device specified.

- */iot_set* (diagnostic_msgs/`KeyValue`)
  When the iot_bridge receives a name/value pair from the ROS iot_set topic, it publishes those to OpenHAB and OpenHAB updates the status for the item specified (e.g. indicates that a switch is now ON).

- */iot_updates* (diagnostic_msgs/`KeyValue`)
  The IoT bridge receives updates from OpenHAB and publishes those as name/value pairs to the iot_updates ROS topic.

- */diagnostics* (diagnostic_msgs/`DiagnosticArray`)
  The IoT bridge receives updates from OpenHAB and publishes those under a DiagnosticArray message to the /diagnostic_agg ROS topic to be monitored by rqt_robot_monitor.

4.3 IP Camera

My biggest project was to integrate an IP Camera in the environment of the robot so that the robot can fulfill missions such as locating an object in the apartment using its own cameras and the IP camera in place. The IP camera was positioned on the ceiling above the apartment.

4.3.1 Redefine the strategy

When choosing the adopted strategies and the camera, the connection between the camera and the robot should be via *iot_bridge*. The camera had to be linked to OpenHAB and communicates via ROS through the bridge. However, after having configured the associated blinding, the speed and the possible action panel were very bad. This did not allow to use the camera dynamically by the robot. So I had to adapt the strategy regarding the camera.

Thus, I developed a set of programs so that the chosen IP camera is both usable by the robot via ROS, but also usable directly from the network, while being integrated with OpenHAB.

4.3.2 Manage the camera

The advantage of the chosen camera is that it has an integrated API in the form of HTTP GET requests. This API allows you to configure videos, control the movement of the PTZ (Pan, Tilt, Zoom) camera, take snapshots, etc. The question then arose as to whether the programming of a "driver" using this API should do this in Python or C++, being the two languages that can be easily integrated with ROS.
After some research, a Python module - *python-amcrest* - already existing, I turned to this language.

The module does not include all the integrated features. Thereby, I upgraded it so that it can be integrated into ROS with the necessary features. Two features have been added:

- Possibility of audio recording for a given duration
- Possibility of saving a PTZ position (preset point)

The version incorporating these changes is available in a Git repository on my account : **python-amcrest ros_used branch** (`https://github.com/KevinBdn/python-amcrest`). The second feature has resulted in a merge request with the official module.

4.3.3 ROS integration

The *python-amcrest* module, being updated, I could create the following ROS package: **amcrest_ip_camera**. The package aims to interact with the HTTP API of the camera through different topics. It is therefore a question of making a second API usable via ROS.

You can check the class diagram of the package developed in the annex. Here is how you can interact with the camera via ROS:

Figure 4.4: Graph node of the **amcrest_ip_camera** package

Two main topics are used as Subscriber

- */ip_camera_position*: In the form of `Point` message, moves the camera according to PTZ configuration.
- */ip_camera_order*: in the form of `KeyValue` message, allows a whole range of action of the camera described in the following table.

Key	Description
Goto	Move the camera to a saved Place
SaveAs	Save the current PTZ configuration as a Place
Remove	Remove a saved Place
RTSP	Enable/Disable the RTSP stream
AudioRec	Record audio for a predefined time
AudioPlay	Play an audio file
Mirror	Configure the Image as the mirror of the current Image
Flip	Flip the Image
Scan	Scan the apartment looking for an Aruco
ScanImgPub	Enbale/Disable the analyzed images during the scan routine
SetTarget	Define the Aruco targeted during the scan routine
SetRoutine	Define the place order during the scan routine
AddTarget	Add an Aruco to the targeted Aruco list
RemoveTarget	Remove an Aruco from the targeted Aruco list
Reboot	Reboot the camera
VideoMode	Change the video configuration
Move	Change the PTZ configuration - Up/Down/Left/Right/etc
Zoom	Zoom In or Out

Table 4.1: Available Key order in the **amcrest_ip_camera** package

In addition, the current status of the camera is published in a `diagnostics` message that can monitor whether or not it works via *rqt_robot_monitor*.

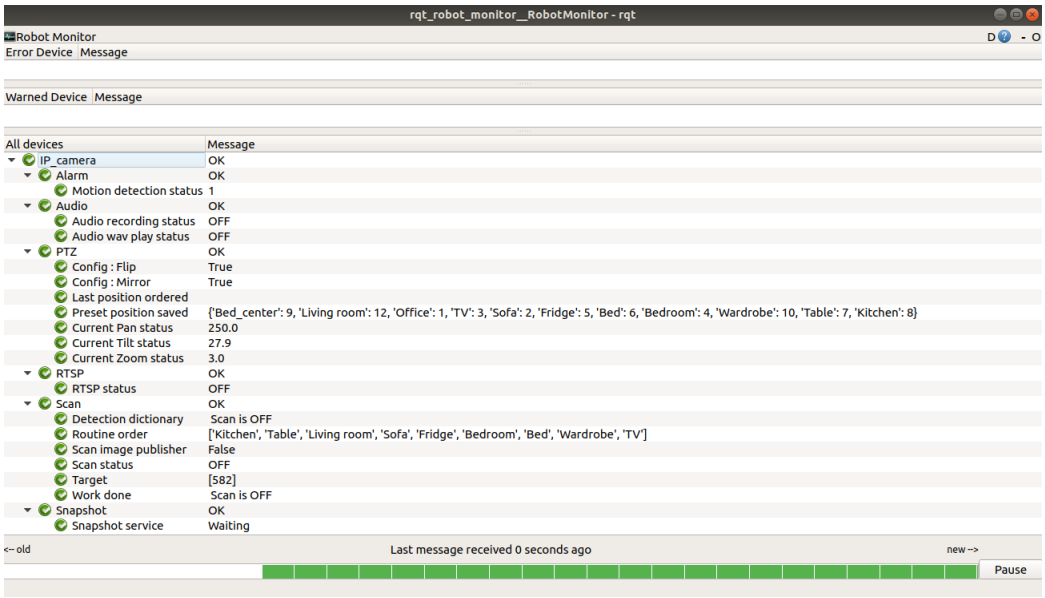


Figure 4.5: *rqt_robot_monitor* from **amcrest_ip_camera** package

4.3.4 Meet the missions

The camera now integrated with ROS, it is possible to meet the missions that will be entrusted to it. For that, I will go back in more details on some of the developed functionalities.

4.3.4.1 PTZ controller

A proportional controller has been integrated. The manufacturer's HTTP API allows to send a single `start` or `stop` command in one direction (left, right, up, down) with a speed ranging from 1 to 8. The same applies for the zoom. The API also provides the possibility of getting the current PTZ configuration.
However, even when sending a `start` then `stop` request with a minimum speed, the camera takes a while to process the HTTP requests - the minimum step is about 0.3°for the pan or the tilt. In addition you should know that the camera has a "no-go" area at the Pan angle between 180° and 185°. Regarding the zoom, there are 5 levels of

current state but the level between two levels is very wide so that for the same zoom value, the actual magnification can be different.
Thanks to these various elements, I was able to create a regulator allowing to position the camera in a desired PTZ configuration. In reality, there are three separated regulators because the camera can only handle one type of movement at a time.
Here is the simplified algorithm implemented for the Pan angle controller:

Algorithm 1 Simplified Pan angle controller

function SAWTOOTH(α) ▷ Implementation in degrees
    $\beta \leftarrow (\alpha + 180) \bmod (360) - 180$
    return β

function PAN_ERROR($\theta, \bar{\theta}$)
    $\epsilon \leftarrow Sawtooth(\theta - \bar{\theta})$
    if $\epsilon < 0$ and *no-go zone* is between θ and $\bar{\theta}$ **then**
        $\epsilon \leftarrow \epsilon + 360$
    else if $\epsilon > 0$ and *no-go zone* is between $\bar{\theta}$ and θ **then**
        $\epsilon \leftarrow \epsilon - 360$
    return ϵ

function PAN_CONTROLLER($\bar{\theta}, \Delta_\theta, \Delta_t, K$)
    if $\bar{\theta}$ is in the *no-go zone* **then** ▷ $\bar{\theta}$: Goal Pan angle
        $\bar{\theta} \leftarrow$ 180°or 185°according to the nearest angle
    $\theta \leftarrow$ Reading the current Pan angle
    $\epsilon \leftarrow Sawtooth(\theta - \bar{\theta})$
    while $|\epsilon| > \Delta_\theta$ **do** ▷ Δ_θ: maximum error
        $u \leftarrow \lfloor min(8, K.|\epsilon|) \rfloor$ ▷ K: proportionality coefficient
        if $\epsilon < 0$ **then**
            Move the camera to the right at speed set to u
        else
            Move the camera to the left at speed set to u
        Wait for Δ_t seconds ▷ Δ_t: time interval
        if $|\epsilon| < 7$ **then**
            Stop the camera ▷ Step by step when the error is low
        $\theta \leftarrow$ Reading the current Pan angle
        $\epsilon \leftarrow Sawtooth(\theta - \bar{\theta})$

Here is schematically the utility of the *Pan_error* function:

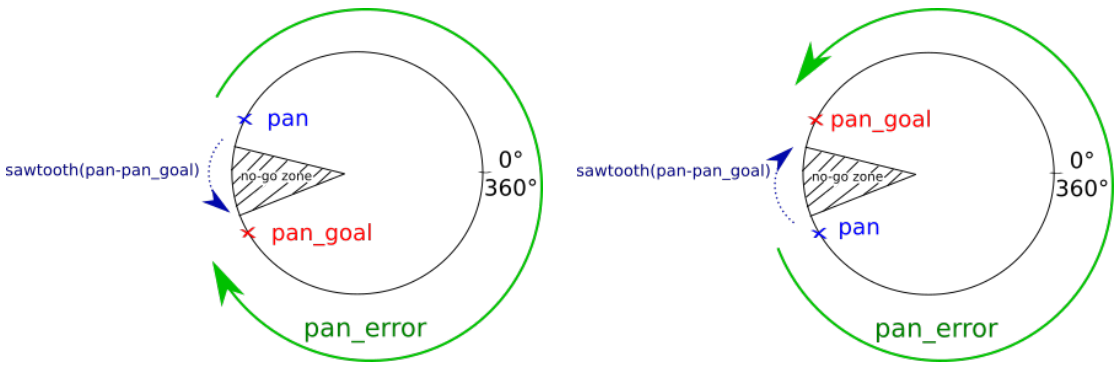


Figure 4.6: Illustration of the *Pan_error* function

4.3.4.2 Place saving

Now able to position the camera in a desired configuration thanks to the controller, I set up an association between a particular position and a place. Thus after configuration, using the `Order` topic with `Goto` as key, you can point

the camera on the Kitchen, Living room, Bedroom, TV, etc., of the apartment. This feature will be very useful later. The dictionary thus created is saved in a local `.yaml` file to keep in memory this association.

4.3.4.3 Calibration

What is calibration ?

Calibrating a camera consists in determining the point transformation that passes from the 3D point expressed in an absolute coordinate system to its image.
It is thus necessary to model the optics of the camera (intrinsic parameters) and to determine the transformation absolute reference / camera reference (extrinsic parameters).
Without calibration we use a simplified model:

- Hypothesis of an image plan
- Pixels regularly spaced on the image plane are assumed
- No geometric distortion (the image of a line is a straight line)

This type of simplified model is used in most cameras marketed at low prices, including the chosen IP camera.

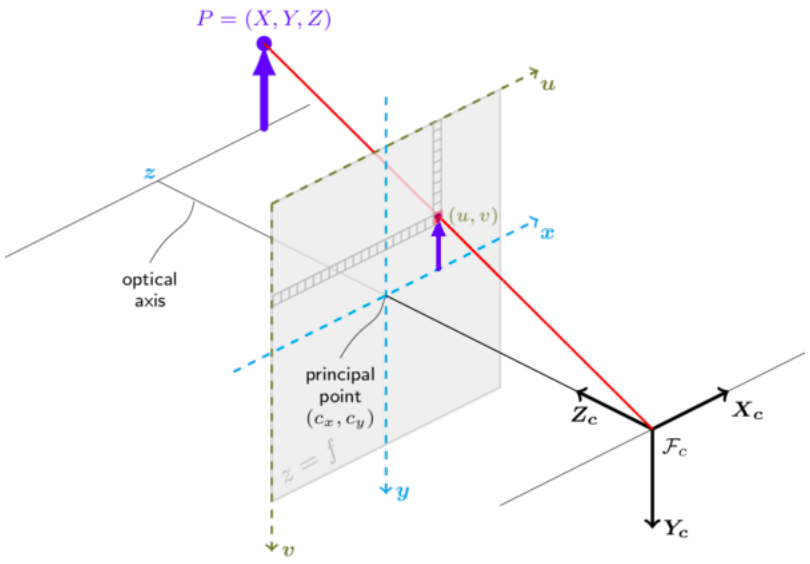


Figure 4.7: Illustration of the pinhole camera model (7)

$P(X,Y,Z) \in R_c(F_c, X_c, Y_c, Z_c)$
$Q(x,y) \in (O,x,y)$, (x,y) in *millimeters* and (u,v) in *pixels* with:

$\frac{1}{k_u}$ the horizontal dimension of the pixel in mm.
$\frac{1}{k_v}$ the vertical dimension of the pixel in mm.

and:

$$\begin{cases} \frac{x}{f} = \frac{X}{Z} \\ \frac{y}{f} = \frac{Y}{Z} \end{cases}$$

But:

$$\begin{cases} u_{/(O,x,y)} = k_u.x + u_0 \\ v_{/(O,x,y)} = k_v.v + v_0 \end{cases}$$
$$\Rightarrow \begin{cases} Z.u = k_u.f.X + u_0.Z \\ Z.v = k_v.f.Y + v_0.Z \end{cases}$$

It is about finding $\alpha_u = k_u.f$, $\alpha_v = k_v.f$, u_0 and v_0 (intrinsic parameters).
The matrix written in homogeneous coordinates is:

$$\begin{pmatrix} s.u \\ s.v \\ s \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}}_{K} . \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

The transformation absolute reference ($R_0(F_0, X_0, Y_0, Z_0)$) / camera reference ($R_c(F_c, X_c, Y_c, Z_c)$) is a displacement defined by a rotation R and a translation T:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R. \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} + T = [R\ T]. \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{pmatrix}$$

So:

$$\begin{pmatrix} s.u \\ s.v \\ s \end{pmatrix} = \underbrace{K.[R\ T]}_{P}. \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{pmatrix}$$

Calibrate a camera means finding its perspective projection matrix P.

How to calibrate ?

To do this, many processes exist. One of them is to calculate P with a plane object whose geometry is known. The parameters are calculated from the observation of several views of the object without needing to know their position (8). The method can be used via opencv and is described in the official documentation following robust methods (9).

Figure 4.8: Calibration process

I used a ROS package - *ros_calibration* - that allows to calculate the coefficients associated with the matrix P by using a checkerboard whose number of boxes and dimension of a square are entered in parameter. I had to publish the camera image and camera information under two topics and launch the *camera_calibration* package. In order to obtain the result, it is necessary to move the checkerboard in the image of the camera in X, Y and Z then according to the corresponding angles. The package then generates a `.yaml` file containing all the coefficients.

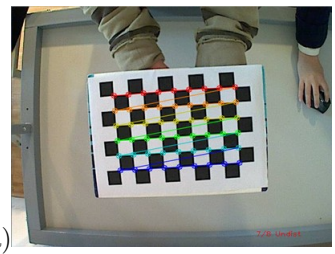
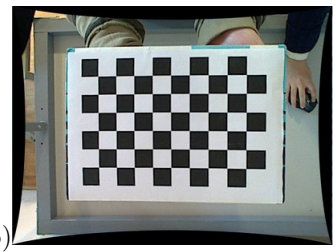
(a)  (b) 

Figure 4.9: Illustration of the correction: during (a) and after (b) the calibration

4.3.4.4 Aruco detection

Now that the IP camera is calibrated, it can be used to detect objects. The objects used are binary square fiducial markers, this form makes them particularly robust to error detection. The ones the lab uses are Aruco.

Two possibilities could be exploited:

- Use OpenCV which has an Aruco library, do the processing via this library directly in the package and publish the result
- Use the ROS package *aruco_detect* which must first have an already corrected image

Schematically here are the two solutions:

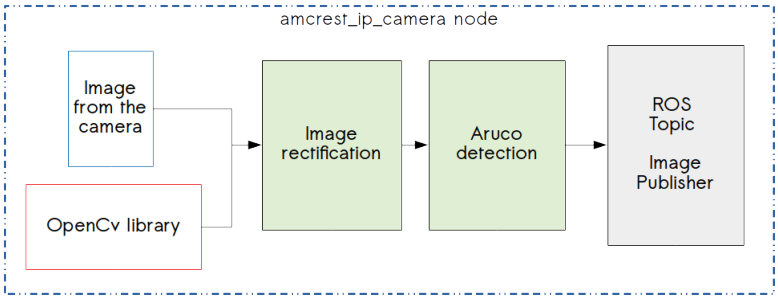


Figure 4.10: Aruco detection directly using OpenCV library

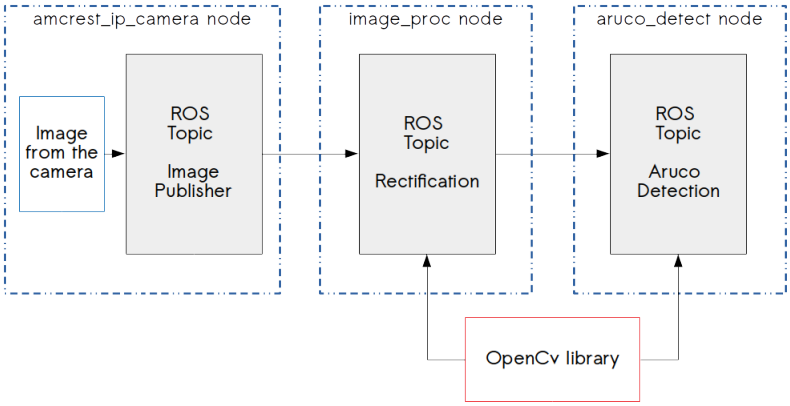


Figure 4.11: Aruco detection using *aruco_detect* ROS package

ROS consuming many resources, I chose to do the treatment internally in the package and not to use the existing package. After implementation we obtain such results:

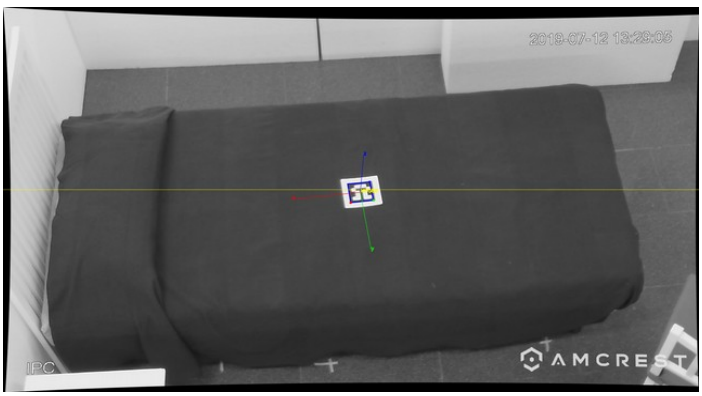


Figure 4.12: Result of an Aruco detection

Having the calibration - by image processing - the Aruco library of opencv makes possible to obtain the vectors of translation and rotation of the Aruco in the reference frame of the camera.

4.3.4.5 Scan algorithm

The camera is now able to:

- Point in a desired direction

- Detect an Aruco and locate it in the camera reference frame

We can now consider an algorithm allowing the camera to scan the apartment in search of an Aruco and providing its location in the reference frame of the apartment.
Here is the simplified algorithm in the form of a flowchart:

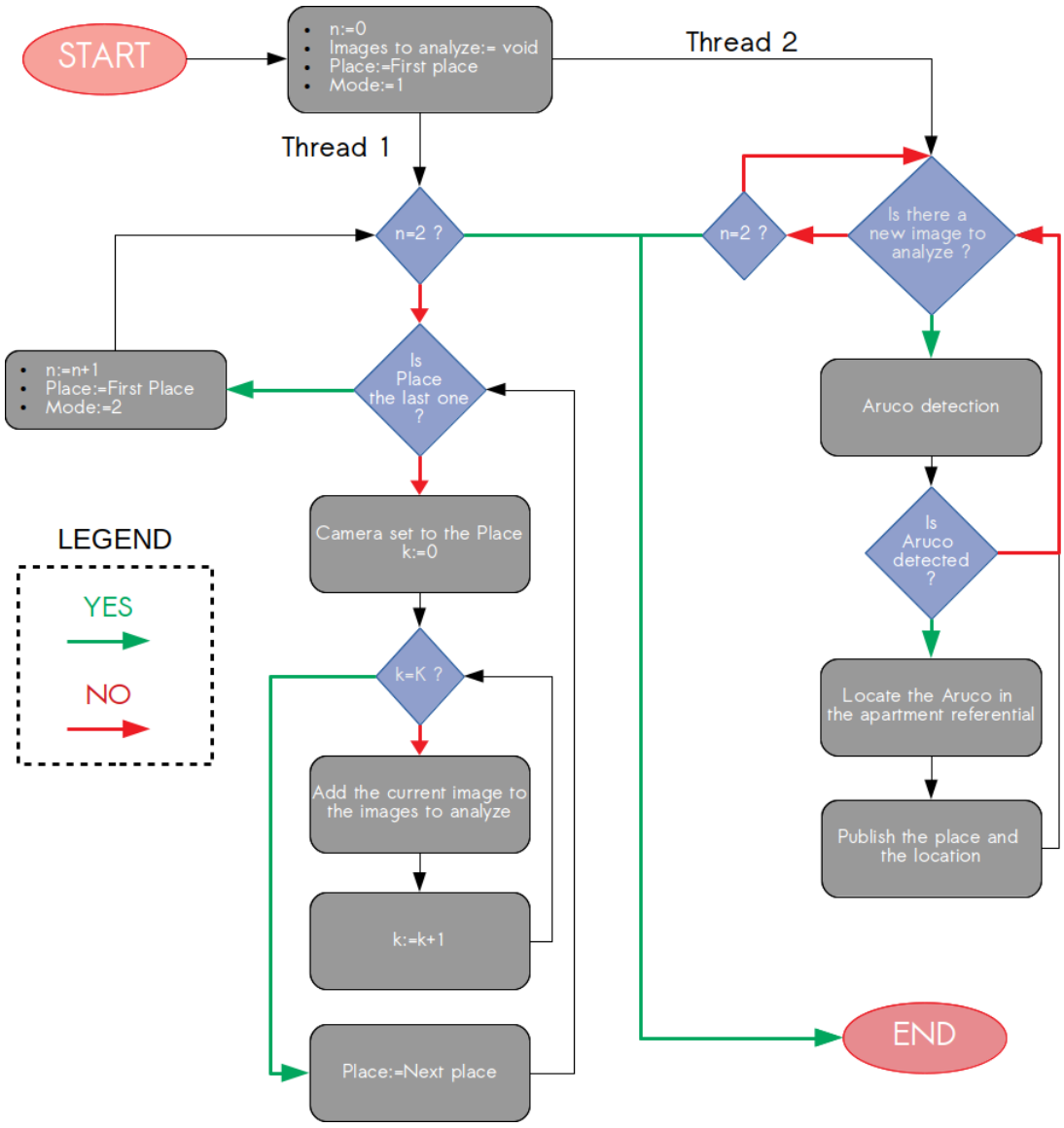


Figure 4.13: Flowchart of the simplified scan algorithm

It is therefore divided into two main threads:

- The first is to make two turns of the different places defined in arguments. For each place we capture K (integer in argument) photos (from the stream RTSP) to have slightly different images for the same place. These captures are stored in an image queue to be analyzed. During the first turn, the camera configuration is in a first mode in Black and White with a high contrast, which allows to see better the Aruco. During the second turn, the camera is in a color mode. Thus we finally capture 2.K images including K of each mode.
- The second is to scroll through the queue of images to be analyzed from the first thread. As soon as a new image appears, it is analyzed to detect an Aruco. If this is the case, then one calculates its position in the reference frame of the apartment having the configuration in Pan and Tilt of the camera, knowing its initial position and obtaining the position of the Aruco in the reference frame of the camera. We can then make a change of reference and publish the position of the calculated Aruco and the place in which it was detected.

The use of threads allows you to go faster by doing parallel operations. Thus, the analysis of the images can be done while the scan routine continues. For instance, the scan of the apartment in 9 places with K = 8, so 144 images, is in about 2 minutes.

4.3.5 OpenHAB integration

The camera can be managed from the online HMI provided by the manufacturer and now from ROS. It seemed interesting to me to control it from OpenHAB2, the initial idea. So I created a new Gitlab project: **ip_camera_openhab**. This project consists in generating the necessary scripts to integrate the camera, used by OpenHAB2 from two configuration files:

- The first being the necessary information for its use ie: IP address, user, password, port.
- The second is the dictionary of places and preset points that can be generated and modified via ROS (see above).

The principle is then to generate switches for each recorded place making it possible to point the camera according to the corresponding place. But also to have a visual via MJPG stream that can be enabled or disabled (to save the bandwidth of the network). It is by directly using the HTTP GET requests of the manufacturer's API that it is possible to integrate these elements to OpenHAB2.

Thus the generator program resulting from this project is based on Parser techniques in order to generate codes. I used Python3.6 and the Jinja2 module for their ease of use to generate scripts based on templates. In addition, this program allows you to add several different cameras using different configuration files, that can be useful if the laboratory decides to buy new IP cameras.

The generated scripts are:

- The interrupter - `.items` - associated with the camera: Multi-choice switch for the places and the switch for activating or deactivating the MJPG stream.
- An entity - `.sitemaps` regrouping these switches, as well as the video stream.

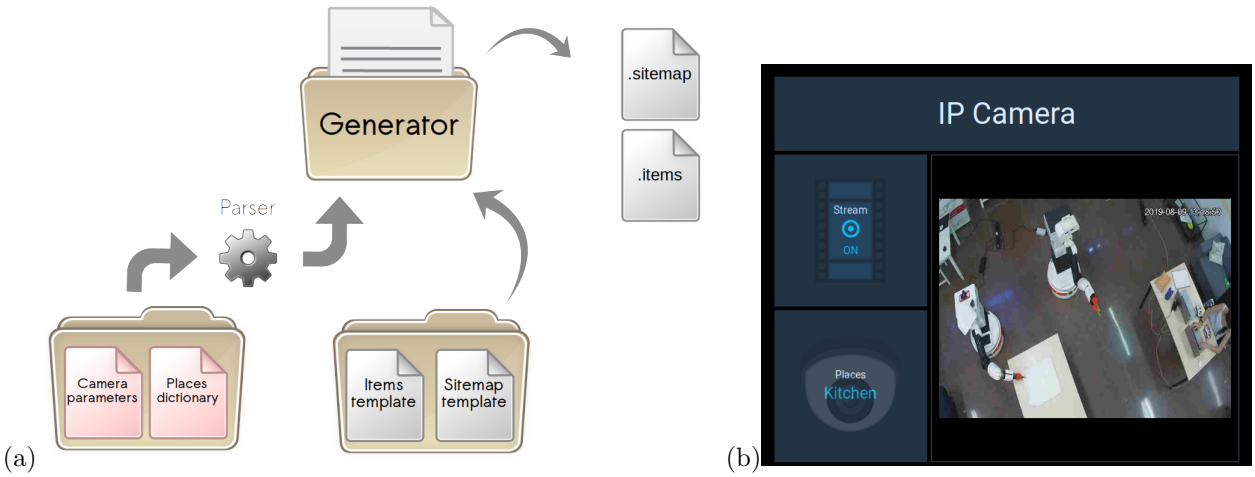


Figure 4.14: The script generator process (a) and its result in HABPanel (b)

After comparison, this method allows a much faster camera management via OpenHAB2 than when using the developing binding initially considered.

4.4 Global view

Here then in schematic form all the projects developed and previously presented allowing the implementation of the IoT part:

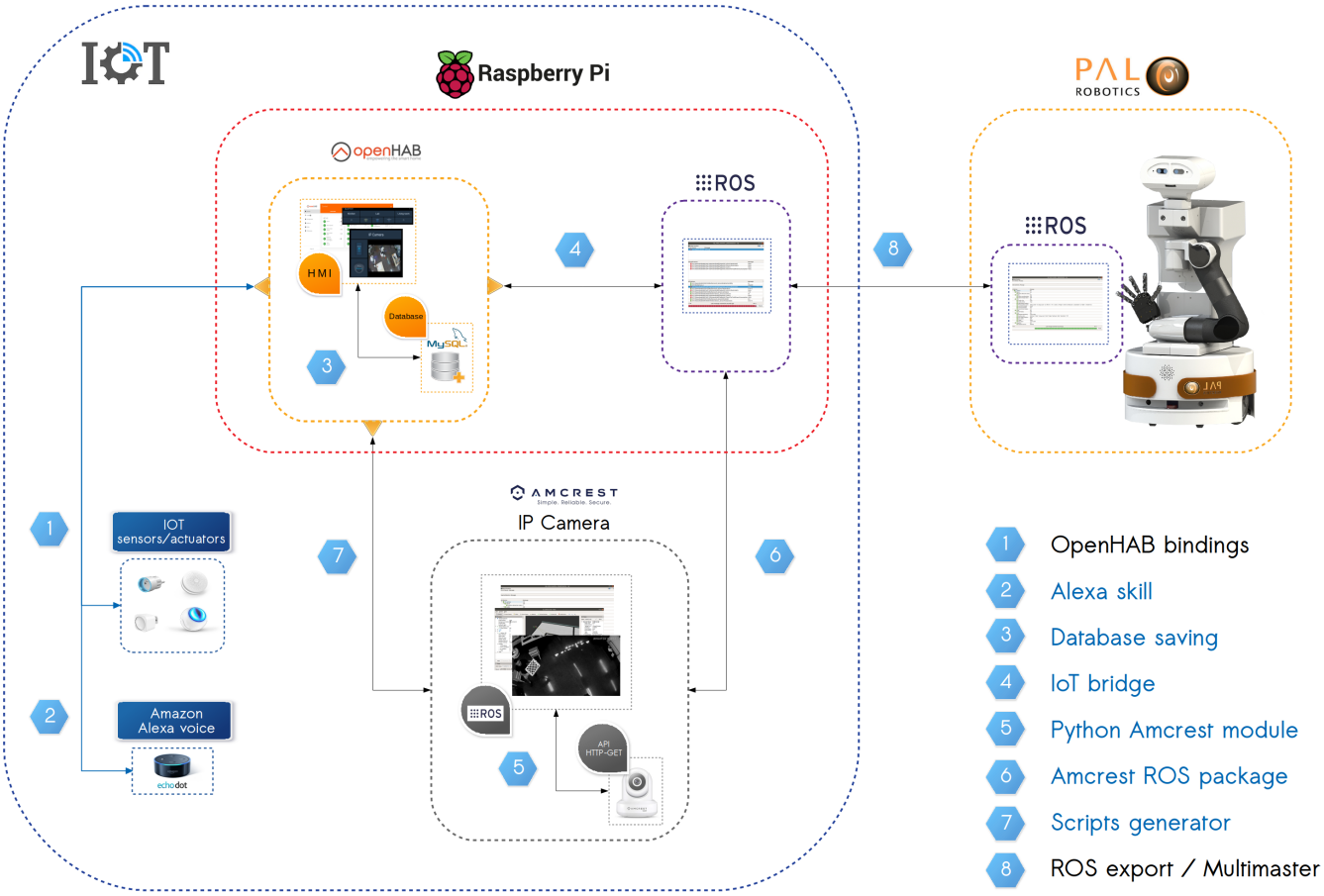


Figure 4.15: Global IOT architecture developed

Each blue description of the legend is a project or tutorial I have developed available on Gitlab and/or Github:

- Alexa skill: **alexa_openhab** (4.2.3)
- Database saving: **OpenHab-MySQL** (4.2.2)
- IoT bridge: **iot_bridge_upgrade** (4.2.4)
- Python Amcrest module: **python-amcrest ros_used branch** (4.3.2)
- Amcrest ROS package: **amcrest_ip_camera** (4.3.3)
- Scripts generator: **ip_camera_openhab** (4.3.5)

Chapter 5

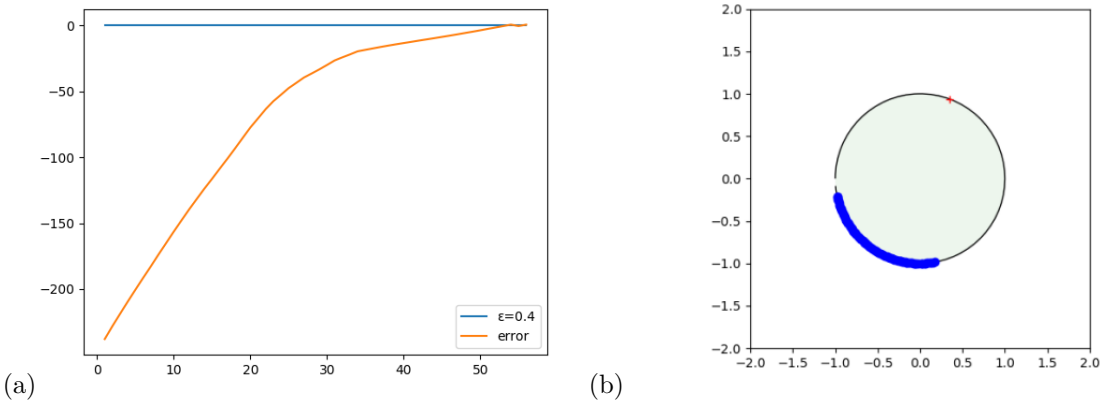
Tests and results

5.1 Simulations

In order to implement the various developed elements of the camera, I went through a simulation phase for the pan controller (being the most complex) and the location of the Aruco in the reference system of the apartment.

5.1.1 Pan controller simulator

The simulator allowed me to study the behavior of the regulator according to the configuration of the camera. For example, it is possible to reverse the directions of the following pan and tilt angles if the mirror and flip modes are on or off.
The figures below represent to the right the camera seen from above with the red cross being the objective to reach and the blue arc being the movement of the camera made. On the left is the error with respect to the setpoint.



(a) (b)

Figure 5.1: Pan controller simulator: Error evolution (a) and evolution of camera movement (b)

In order to better adjust the proportionality coefficient of the implemented controller, I introduced a Gaussian noise in the time lag between two consecutive measurements, which represents the minimum execution time between two HTTP requests from the camera queue. During this time the camera continues its motion, resulting in a minimum angle of about 0.3°. In some cases, a slight overshoot can be obtained, that can be accompanied by oscillations. This can be reduced by choosing a higher Δ_θ (of the order of 0.4°/0.5°) which will represent the uncertainty of the position of the regulator.

5.1.2 Location simulator

In order to locate in the reference frame of the apartment an Aruco detected in the reference frame of the camera, I simulated the calculations to implement the good formulas within the ROS package.
In a first time, it allows to calibrate the initial position of the camera in the apartment. By positioning the Aruco in the center of the apartment's reference frame, measuring its depth knowing the Pan and Tilt angles, the simulator can estimate the initial position P_0 of the camera.
Then in a second time, it allows to check the consistency of the calibration, which provides the location of the Aruco in its reference frame. Thus by recovering, from an image with Aruco detection, the Aruco translation vector V_t as well as the rotation vector V_r, the simulator can locate and return the position and orientation of the Aruco in the reference frame of the apartment.

(a)

```
Input:
     * Translation vector:
[[-1.3031]
 [-0.8679]
 [ 4.2843]]
     * Rotation vector:
[[ 2.10435373]
 [ 1.38719121]
 [-0.57396549]]


Output:
     * Translation vector:
[[3.06067301]
 [0.36520362]
 [1.20915717]]
     * Rotation matrix:
[[ 0.21970145 -0.97388942  0.05718977]
 [ 0.97554763  0.21969012 -0.00656322]
 [-0.00617217  0.05723329  0.99834175]]
```

(b)

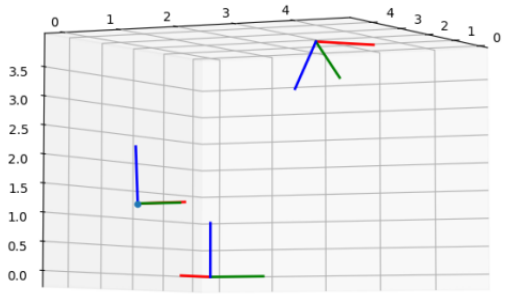


Figure 5.2: Location simulator: Calculated coordinates (a) and the location (b) with the representation of: the camera (up), the Aruco (middle) and the apartment origin (down)

The new translation vector V_t is obtained by reference frame change:

$$V_{t/Apartment} = \underbrace{\underbrace{\begin{pmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\theta=Pan\ angle} . \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{pmatrix}}_{\phi=Tilt\ angle}}_{R_C} .V_{t/Camera} + \underbrace{P_0}_{Initial\ camera\ position}$$

The new matrix rotation $R_{/Apartment}$ is obtained by combining the rotation matrix of the camera R_C, known thanks to its angles Pan and Tilt, and that generated by the Rodrigues' formula (10), $R_{/Camera}$, from the rotation vector V_r provided by the camera.

$$\psi \leftarrow ||V_r||$$
$$V_r \leftarrow \frac{V_r}{\psi}$$
$$R_{/Camera} = cos(\psi).I + (1 - cos(\psi)).V_r.V_r^T + sin(\psi). \begin{pmatrix} 0 & -V_{rz} & V_{ry} \\ V_{rz} & 0 & -V_{rx} \\ -V_{ry} & V_{rx} & 0 \end{pmatrix}$$
$$R_{/Apartment} = R_{/Camera}.R_C$$

5.2 Integration tests

Two main integration tests were carried out. They made it possible to test communicating elements through different links of the IoT chain thus developed. The first is to test the chain between ROS and OpenHAB2 via *iot_bridge*. The second one aims to test the apartment scan algorithm in its entirety until the communication of the location of the Aruco in the apartment.

5.2.1 Wallplug test

The Wallplug test was a test performed in order to check the behavior of OpenHAB2, the OpenHAB MySQL database, the *iot_bridge_upgrade* package and its topics. During the test period, the wallplug is switched `ON` and switched `OFF` two times per day. The command is send to the *iot_command* topics of the *iot_bridge* package from the *wallplug_test* node of the **Wallplug python script**.
At each new command, the database is checked before sending the command and one second later to control the proper operation of the data recording when a change of state occurs.

The command line is tested as follow:

- An user wants to activate an device through ROS
- The *iot_command* topic catch the order
- The order is send to OpenHAB through the *iot_bridge*
- OpenHAB changes the state of the `Item`

- The MySQL persistence saves the new value into the OpenHAB database

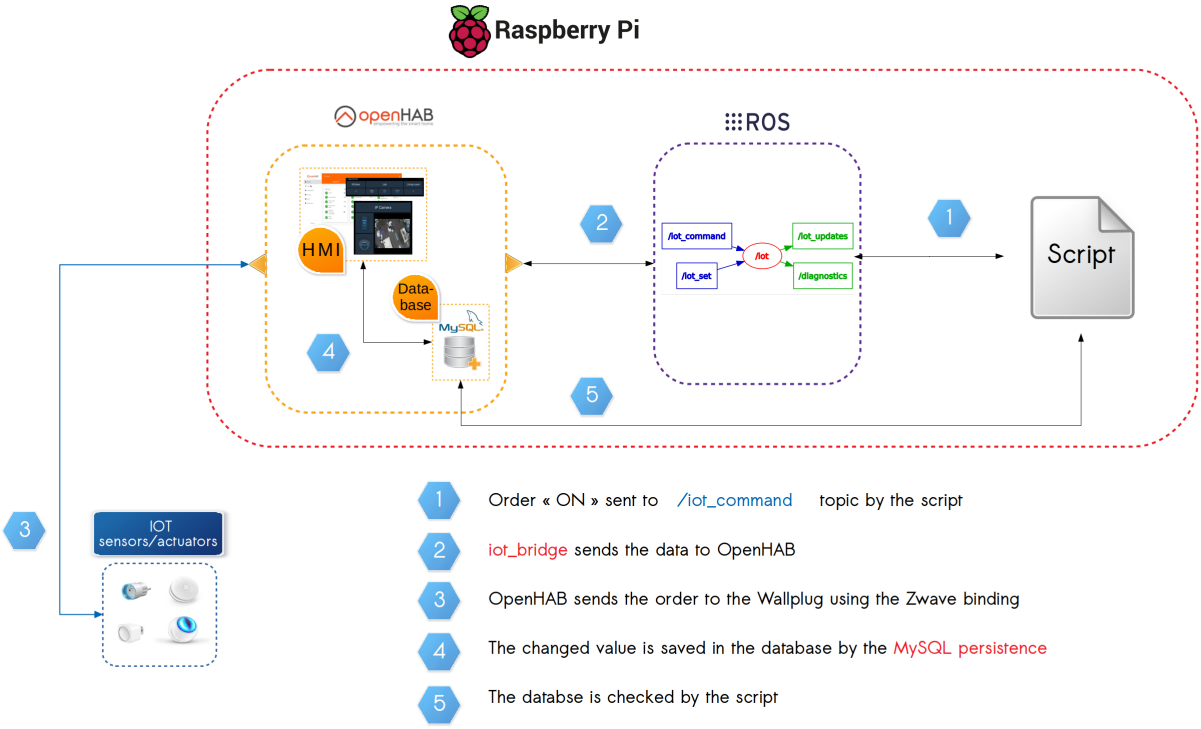


Figure 5.3: Wallplug test process

You can check the results of the test in the `report.log` file in annex which was generated by the **wallplug_test** script during the test. Notice that on the Raspberry (and so in the database) the time is UTC (so there are two hours difference between the time in the log info and the time in the database info). As hoped, the command line works well and fast, there is less than one second between the publication of the command to the *iot_command* topic and the data saving into the database.

5.2.2 Scan routine

It is a matter of testing the entire scan algorithm of the camera. For this, the package is launched on the Rapsberry Pi and must provide to a computer - connected on the same local network - the Aruco position in the apartment.

The command line tested is:

- Export the ROS Master to a network computer
- Stimulation of the *ip_camera_order* node to start the scan
- Running the scan algorithm internally
- Report the current state of the camera via the *diagnostics* node
- Sending results via *ip_camera_scan_result* broadcast node of the camera and Aruco TF
- Visualization of the results via RVIZ

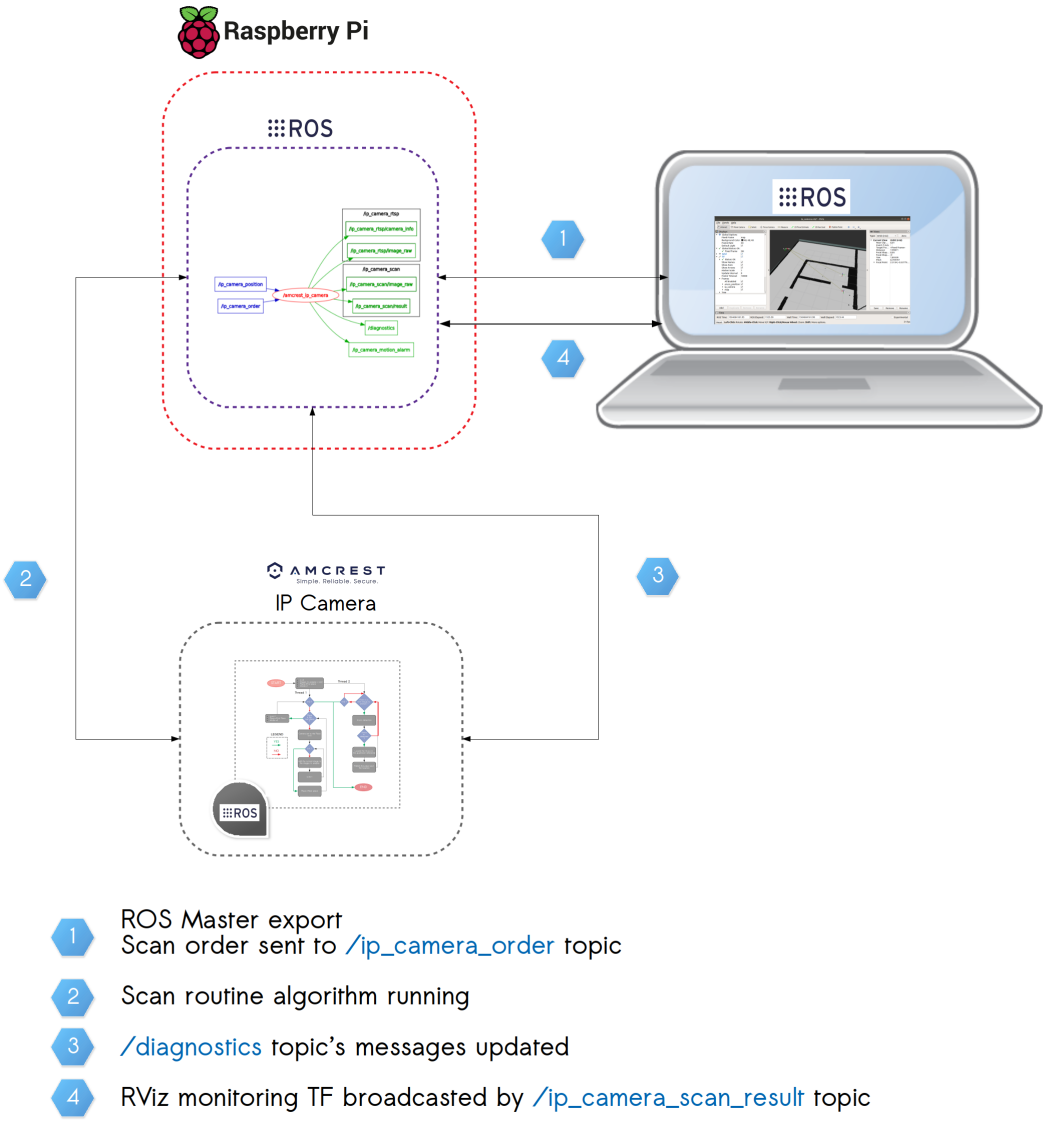


Figure 5.4: Scan routine test process

You can thus compare the result of RVIZ with the actual location of the Aruco in the apartment.

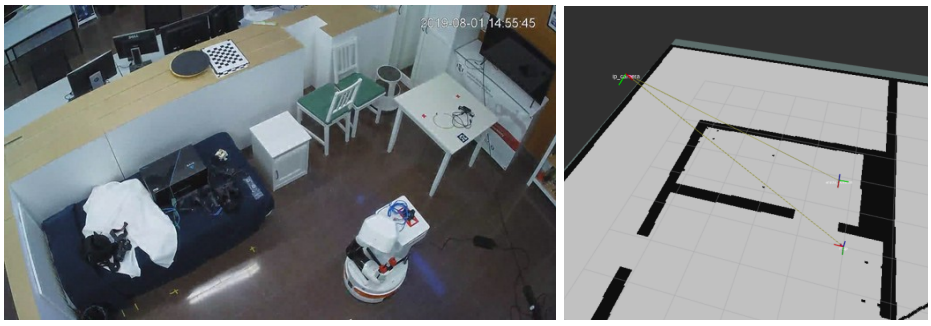


Figure 5.5: Result of the Scan routine test

As expected the detection of Aruco works well and its location in the apartment is good with an error of the decimeter. The error on the position is not a real problem because the purpose of this algorithm is to provide an indication to the robot so that it can get a visual itself on the Aruco. Thus the geographical indication - Kitchen, TV, Linving room, etc. - is sufficient for this use.

5.3 Mission of Aruco detection

After having implemented the scan algorithm in the ROS package, we are now able to use it with TIAGo in a mission of Aruco detection. Here is, under a flowchart form, an algorithm tested to success this kind of mission.

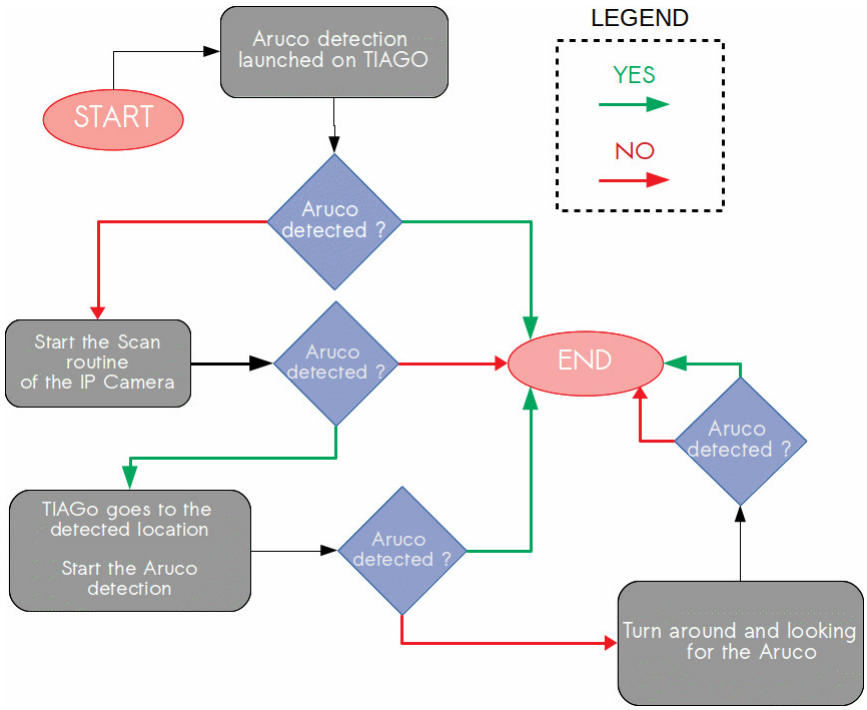


Figure 5.6: Flowchart of the algorithm used in the mission of Aruco detection

To make it possible, I used the Lab's API on a computer making the bridge between the Raspberry Pi and TIAGo and providing a way to monitor the mission.

Here is the process used in the performed test:

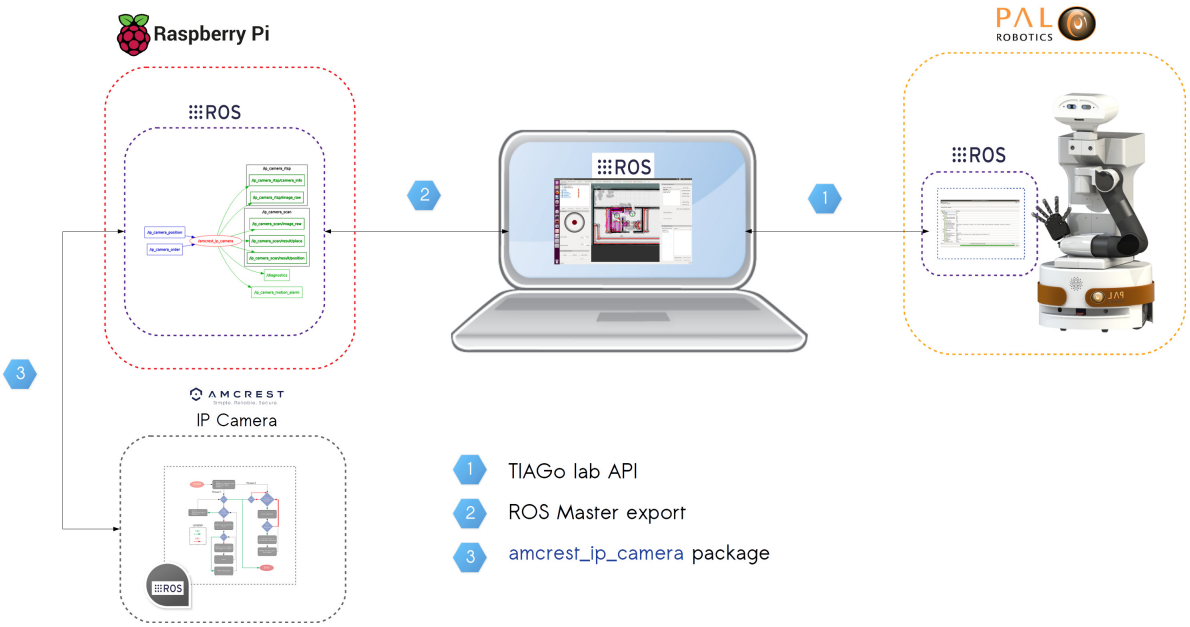


Figure 5.7: Architecture used in the mission of Aruco detection

1. An Aruco is placed on the table in the living room near to the sofa. The TIAGo is the bedroom.

2. TIAGo does not find the Aruco using its own camera. The scan routine from the **amcrest_ip_camera** package on the Raspberry Pi is started with a ROS export on the monitoring computer. When the scan routine finds an Aruco, it publishes its location on the *ip_camera_scan/result/place* topic.
3. TIAGo is ordered to go to the corresponding place.
4. TIAGo finds the Aruco by itself using its own camera.

Here are the Aruco detection from the IP Camera and that by TIAGo at the end of the process.

a. b.

Figure 5.8: Aruco detection by the camera (a.) and by TIAGo (b.)

This example of implementation shows that it is useful to use IoT devices in a mission for assistive automation. We can now imagine more complex algorithms and missions where IoT devices are part of the robot's environment.

Chapter 6

Conclusion

6.1 About the project

All my work has shown that it was currently possible to integrate the many devices of the world of IoT within a service robot using the ROS middleware. Such an implementation not only enriches the robot's sensors but also increases its range of actions. The use of IoT in robotics thus makes it possible to increase the speed with which a robot can take information by providing it these new tools. As we do it, the robot - according to the mission which has been given to it - can use or not this information. I was able to highlight this phenomenon through the mission of detection and location of an Aruco in the apartment. However, I have found that the fragility of this union relies on - like most of our current technologies - the capacity and the robustness of the network in which it evolves. Indeed, the use of these many IoT protocols, the IP camera in real time, as well as the ROS middleware in multimaster and / or export consumes a lot of bandwidth of the local network. Furthermore, using several IoT devices such IP cameras can consume more resources than a Raspberry Pi 3B+ is able to provide. It would therefore be interesting to study solutions to ensure that the exchange of information between these various elements is at their maximum capacity and is neither curbed nor slowed down by the network nor by the hardware in which they operate.
More globally, this project adds a brick to the features developed by the Perception and Manipulation laboratory. This is how the laboratory works, each thesis subject, each project, each research allows, thanks to the modularity of the ROS architecture implemented in the Tiago robots, to build a much larger and more robust system, project after project. My project has therefore formed the basis for IoT within this laboratory and should allow its integration into other projects.
To finish all of my work is also available in the form of a technical report (1) which resulted in a publication under the reference IRI-TR-19-03, published by the CSIC-UPC on the site of the institute.

6.2 Personal

More personally this project allowed me to discover the functioning of a laboratory animated by thesis and end of studies project. Indeed I was able to take part in various experiments conducted by doctoral students. I have also been able to improve my english by writing daily and communicating within the laboratory as well as my Spanish using it in everyday life.
From a technical point of view, I learned a lot about IoT and ROS. In addition, the daily use of Git has brought me a lot in terms of the structure and operation of a collaborative program. It is the same regarding the C, C++ and Python programming.
I have also been able to use and enrich knowledge acquired in the classroom such as controller, reference frame changes, parsing techniques and image processing.
Thus this internship was very enriching as well on the technical, theoretical as human level.

Bibliography

[1] K. Bedin, S. Foix, and G. Alenya, "Robots and IoT devices for assistive automation IRI Technical Report," p. 40.

[2] L. Columbus, "10 charts about the iot's growth," June 2018. [Online]. Available: https://www.forbes.com/sites/louiscolumbus/2018/06/06/10-charts-that-will-challenge-your-perspective-of-iots-growth/#7baeeca33ecc

[3] I. F. of Robotics, "Why service robots are booming worldwide IFR forecasts sales up 12%," Oct. 2017. [Online]. Available: https://ifr.org/downloads/press/2017-10-11_PR_IFR_World_Robotics_Report_2017_Service_Robots_ENG_FINAL_1.pdf

[4] "IRI - Robots and IoT devices for assistive automation." [Online]. Available: https://www.iri.upc.edu/pfc/show/179

[5] G. A. A. de Oliveira, R. W. de Bettio, and A. P. Freire, "Accessibility of the smart home for users with visual disabilities: an evaluation of open source mobile applications for home automation." ACM, Apr. 2016, p. 29.

[6] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3. Kobe, Japan, 2009, p. 5.

[7] "Camera Calibration and 3d Reconstruction OpenCV 2.4.13.7 documentation." [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

[8] Z. Zhang, "A Flexible New Technique for Camera Calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, Dec. 2000.

[9] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *cvpr*, vol. 97. Citeseer, 1997, p. 1106.

[10] J. S. Dai, "EulerRodrigues formula variations, quaternion conjugation and intrinsic connections," *Mechanism and Machine Theory*, vol. 92, pp. 144–152, Oct. 2015.

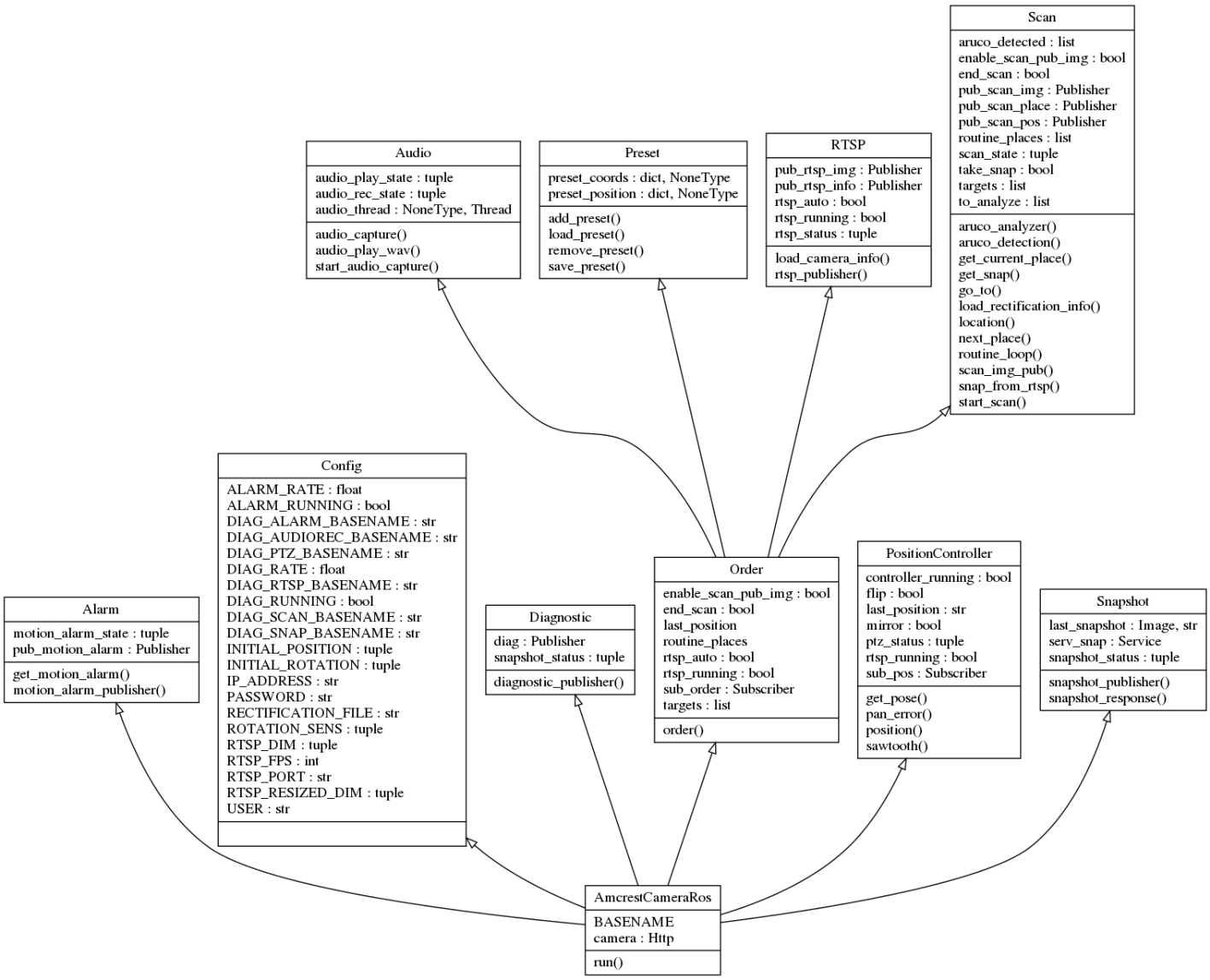
List of Figures

4.1 HAB Panel screenshot . . . 6
4.2 Database from MySQL Persistence . . . 7
4.3 Overview diagram of the **iot_bridge_upgrade** package . . . 8
4.4 Graph node of the **amcrest_ip_camera** package . . . 9
4.5 *rqt_robot_monitor* from **amcrest_ip_camera** package . . . 10
4.6 Illustration of the *Pan_error* function . . . 11
4.7 Illustration of the pinhole camera model (7) . . . 12
4.8 Calibration process . . . 13
4.9 Illustration of the correction: during (a) and after (b) the calibration . . . 13
4.10 Aruco detection directly using OpenCV library . . . 14
4.11 Aruco detection using *aruco_detect* ROS package . . . 14
4.12 Result of an Aruco detection . . . 14
4.13 Flowchart of the simplified scan algorithm . . . 15
4.14 The script generator process (a) and its result in HABPanel (b) . . . 16
4.15 Global IOT architecture developed . . . 17

5.1 Pan controller simulator: Error evolution (a) and evolution of camera movement (b) . . . 18
5.2 Location simulator: Calculated coordinates (a) and the location (b) with the representation of: the camera (up), the Aruco (middle) and the apartment origin (down) . . . 19
5.3 Wallplug test process . . . 20
5.4 Scan routine test process . . . 21
5.5 Result of the Scan routine test . . . 21
5.6 Flowchart of the algorithm used in the mission of Aruco detection . . . 22
5.7 Architecture used in the mission of Aruco detection . . . 22
5.8 Aruco detection by the camera (a.) and by TIAGo (b.) . . . 23

Annex

Class diagram of the amcrest_ip_camera ROS package



Wallplug test : *report.log* file

```
[21_06_2019-16H07m22s] 'Start test'
[21_06_2019-16H07m22s] '* ON order date: ['21_06_2019-16H08m10s',
    '21_06_2019-22H14m12s', '22_06_2019-10H00m34s', '22_06_2019-18H06m10s',
    '23_06_2019-14H00m55s', '23_06_2019-21H44m10s', '24_06_2019-03H00m10s',
    '24_06_2019-15H22m10s']'
[21_06_2019-16H07m22s] '* OFF order date: ['21_06_2019-16H11m25s',
    '21_06_2019-22H15m10s', '22_06_2019-10H05m08s', '22_06_2019-18H07m10s',
    '23_06_2019-14H01m03s', '23_06_2019-21H46m10s', '24_06_2019-05H00m10s',
    '24_06_2019-15H34m10s']'
[21_06_2019-16H08m10s] '  It's time to send <ON>'
[21_06_2019-16H08m10s] '      [MySQL] Previous value - 2019-06-21 14:03:25 OFF'
[21_06_2019-16H08m10s] '      [ROS] Command <ON> sent to <HTC02_Switch>'
[21_06_2019-16H08m11s] '      [MySQL] New value - 2019-06-21 14:08:10 ON'
[21_06_2019-16H11m25s] '  It's time to send <OFF>'
[21_06_2019-16H11m25s] '      [MySQL] Previous value - 2019-06-21 14:10:00 ON'
[21_06_2019-16H11m25s] '      [ROS] Command <OFF> sent to <HTC02_Switch>'
[21_06_2019-16H11m26s] '      [MySQL] New value - 2019-06-21 14:11:25 OFF'
[21_06_2019-22H14m12s] '  It's time to send <ON>'
[21_06_2019-22H14m12s] '      [MySQL] Previous value - 2019-06-21 20:00:00 OFF'
[21_06_2019-22H14m12s] '      [ROS] Command <ON> sent to <HTC02_Switch>'
[21_06_2019-22H14m13s] '      [MySQL] New value - 2019-06-21 20:14:12 ON'
[21_06_2019-22H15m10s] '  It's time to send <OFF>'
[21_06_2019-22H15m10s] '      [MySQL] Previous value - 2019-06-21 20:14:12 ON'
[21_06_2019-22H15m11s] '      [ROS] Command <OFF> sent to <HTC02_Switch>'
[21_06_2019-22H15m12s] '      [MySQL] New value - 2019-06-21 20:15:11 OFF'
[22_06_2019-10H00m34s] '  It's time to send <ON>'
[22_06_2019-10H00m34s] '      [MySQL] Previous value - 2019-06-22 08:00:00 OFF'
[22_06_2019-10H00m34s] '      [ROS] Command <ON> sent to <HTC02_Switch>'
[22_06_2019-10H00m35s] '      [MySQL] New value - 2019-06-22 08:00:34 ON'
[22_06_2019-10H05m08s] '  It's time to send <OFF>'
[22_06_2019-10H05m08s] '      [MySQL] Previous value - 2019-06-22 08:00:34 ON'
[22_06_2019-10H05m08s] '      [ROS] Command <OFF> sent to <HTC02_Switch>'
[22_06_2019-10H05m09s] '      [MySQL] New value - 2019-06-22 08:05:08 OFF'
[22_06_2019-18H06m10s] '  It's time to send <ON>'
[22_06_2019-18H06m10s] '      [MySQL] Previous value - 2019-06-22 16:00:00 OFF'
[22_06_2019-18H06m10s] '      [ROS] Command <ON> sent to <HTC02_Switch>'
[22_06_2019-18H06m11s] '      [MySQL] New value - 2019-06-22 16:06:10 ON'
[22_06_2019-18H07m10s] '  It's time to send <OFF>'
[22_06_2019-18H07m10s] '      [MySQL] Previous value - 2019-06-22 16:06:10 ON'
[22_06_2019-18H07m10s] '      [ROS] Command <OFF> sent to <HTC02_Switch>'
[22_06_2019-18H07m11s] '      [MySQL] New value - 2019-06-22 16:07:10 OFF'
[23_06_2019-14H00m55s] '  It's time to send <ON>'
[23_06_2019-14H00m55s] '      [MySQL] Previous value - 2019-06-23 12:00:00 OFF'
[23_06_2019-14H00m55s] '      [ROS] Command <ON> sent to <HTC02_Switch>'
[23_06_2019-14H00m56s] '      [MySQL] New value - 2019-06-23 12:00:55 ON'
[23_06_2019-14H01m03s] '  It's time to send <OFF>'
[23_06_2019-14H01m03s] '      [MySQL] Previous value - 2019-06-23 12:00:55 ON'
[23_06_2019-14H01m03s] '      [ROS] Command <OFF> sent to <HTC02_Switch>'
[23_06_2019-14H01m04s] '      [MySQL] New value - 2019-06-23 12:01:03 OFF'
[23_06_2019-21H44m10s] '  It's time to send <ON>'
[23_06_2019-21H44m10s] '      [MySQL] Previous value - 2019-06-23 19:30:00 OFF'
[23_06_2019-21H44m10s] '      [ROS] Command <ON> sent to <HTC02_Switch>'
[23_06_2019-21H44m11s] '      [MySQL] New value - 2019-06-23 19:44:10 ON'
[23_06_2019-21H46m10s] '  It's time to send <OFF>'
[23_06_2019-21H46m11s] '      [MySQL] Previous value - 2019-06-23 19:44:10 ON'
[23_06_2019-21H46m11s] '      [ROS] Command <OFF> sent to <HTC02_Switch>'
[23_06_2019-21H46m12s] '      [MySQL] New value - 2019-06-23 19:46:11 OFF'
[24_06_2019-03H00m10s] '  It's time to send <ON>'
[24_06_2019-03H00m10s] '      [MySQL] Previous value - 2019-06-24 01:00:00 OFF'
[24_06_2019-03H00m10s] '      [ROS] Command <ON> sent to <HTC02_Switch>'
[24_06_2019-03H00m11s] '      [MySQL] New value - 2019-06-24 01:00:10 ON'
[24_06_2019-05H00m10s] '  It's time to send <OFF>'
```

```
[24_06_2019-05H00m10s] '      [MySQL] Previous value - 2019-06-24 03:00:00 ON'
[24_06_2019-05H00m10s] '      [ROS] Command <OFF> sent to <HTC02_Switch>'
[24_06_2019-05H00m11s] '      [MySQL] New value - 2019-06-24 03:00:10 OFF'
[24_06_2019-15H22m10s] '  It's time to send <ON>'
[24_06_2019-15H22m10s] '      [MySQL] Previous value - 2019-06-24 13:00:00 OFF'
[24_06_2019-15H22m10s] '      [ROS] Command <ON> sent to <HTC02_Switch>'
[24_06_2019-15H22m11s] '      [MySQL] New value - 2019-06-24 13:22:10 ON'
[24_06_2019-15H34m10s] '  It's time to send <OFF>'
[24_06_2019-15H34m10s] '      [MySQL] Previous value - 2019-06-24 13:30:00 ON'
[24_06_2019-15H34m10s] '      [ROS] Command <OFF> sent to <HTC02_Switch>'
[24_06_2019-15H34m11s] '      [MySQL] New value - 2019-06-24 13:34:10 OFF'
[25_06_2019-08H41m06s] 'End test'
```

AMCREST

IP3M-941

Amcrest 3MP Dual Band Wi-Fi PTZ IP Camera

Technical Specifications

Model		**IP3M-941B, IP3M-941W**
Camera		
Image Sensor		1/3" Megapixel progressive scan CMOS
Effective Pixels		3M (2304x1296)
Scanning System		Progressive
Electronic Shutter Speed		Auto/Manual 1/3 (4) ~1/100000
Min. Illumination		0. 1Lux / F2.2 (Color), 0Lux / F2.2 (IR on)
S/N Ratio		More than 56dB
Video Output		N/A
Camera Features		
Max. IR LEDs Length		10m (32.8ft)
Day/Night		Auto (ICR) / Color / B&W
Backlight Compensation		BLC / HLC / WDR
White Balance		Auto / Manual
Gain Control		Auto / Manual
Digital Zoom		16x
Noise Reduction		3D
Privacy Masking		Up to 4 areas
Lens		
Focal Length		4mm
Max Aperture		F2.2
Focus Control		Manual
Angle of View		90°
Lens Type		Fixed lens
Video		
Compression		H.264H / H.264B / H.264 / MJPEG
Resolution		3M(2304x1296)/1080P(1920×1080)/720P(1280×720)/VGA(640x480)
Frame Rate	Main Stream	3M(2304x1296) (1 ~ 20fps), 1080P/720p (1 ~ 30fps)
	Sub Stream	VGA (1 ~ 30fps)

Bit Rate	H.264H: 12K ~ 10240Kbps
Audio	
Compression	G.711MU, G711A, AAC
Interface	1 in/ 1 out
Network	
Ethernet	RJ-45 (10/100Base-T)
Wi-Fi	2.4GHz (802.11b/g/n) / 5GHz (802.11ac/a/n)
Protocol	IPv4/IPv6, HTTP, HTTPS, TCP/IP, UDP, UPnP, ICMP, IGMP, RTSP, RTP
Interoperability	ONVIF, PSIA, CG
Streaming Method	Unicast / Multicast
Edge Storage	NAS, FTP, Local PC, MicroSD Card (128GB)
Management Software	Amcrest Surveillance Pro (Windows/MAC) Amcrest View Pro app for IOS and Android AmcrestCloud.com Video Storage Subscription Service (Chrome, Edge, Firefox, Safari) Blue Iris Professional (Third Party) Web Browser (Pale Moon, Sea Monkey, Firefox 49.0, Internet Explorer, Chrome with Amcrest Extension, Safari)
Auxiliary Interface	
Memory Slot	Micro SD
Alarm	Alarm in/ Alarm out
General	
Power Supply	DC5V, 2.0A
Power Consumption	<7.5W
Working Environment	-10°C~+45°C, Less than 95%RH

ENSTA Bretagne

RAPPORT D'EVALUATION
ASSESSMENT REPORT

Merci de retourner ce rapport par courrier ou par voie électronique en fin du stage à :
At the end of the internship, please return this report via mail or email to:

ENSTA Bretagne – Bureau des stages - 2 rue François Verny - 29806 BREST cedex 9 – FRANCE
☎ 00.33 (0) 2.98.34.87.70 / stages@ensta-bretagne.fr

I - ORGANISME / *HOST ORGANISATION*

NOM / *Name* INSTITUT DE ROBÒTICA I INFORMÀTICA INDUSTRIAL

Adresse / *Address* C/ LLORENS I ARTIGAS 4-6
08028 BARCELONA

Tél / *Phone (including country and area code)* +34 93 401 57 51

Nom du superviseur / *Name of internship supervisor*
GUILLEM ALENYÀ

Fonction / *Function* RESEARCHER

Adresse e-mail / *E-mail address* galenya@iri.upc.edu

Nom du stagiaire accueilli / *Name of intern* BEDIN Kévin

II - EVALUATION / *ASSESSMENT*

Veuillez attribuer une note, en encerclant la lettre appropriée, pour chacune des caractéristiques suivantes. Cette note devra se situer entre **A (très bien)** et **F (très faible)**
*Please attribute a mark from **A (excellent)** to **F (very weak).***

MISSION / *TASK*

- La mission de départ a-t-elle été remplie ? (A) B C D E F
  Was the initial contract carried out to your satisfaction?

- Manquait-il au stagiaire des connaissances ? ☐ oui/*yes* ☒ non/*no*
  Was the intern lacking skills?

  Si oui, lesquelles ? / *If so, which skills?* _____

ESPRIT D'EQUIPE / *TEAM SPIRIT*

- Le stagiaire s'est-il bien intégré dans l'organisme d'accueil (disponible, sérieux, s'est adapté au travail en groupe) / *Did the intern easily integrate the host organisation? (flexible, conscientious, adapted to team work)*
  (A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

COMPORTEMENT AU TRAVAIL / *BEHAVIOUR TOWARDS WORK*

Le comportement du stagiaire était-il conforme à vos attentes (Ponctuel, ordonné, respectueux, soucieux de participer et d'acquérir de nouvelles connaissances) ?
Did the intern live up to expectations? (Punctual, methodical, responsive to management instructions, attentive to quality, concerned with acquiring new skills)?

(A) B C D E F

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____
We are very happy with his behavior and attitude.

INITIATIVE – AUTONOMIE / *INITIATIVE – AUTONOMY*

Le stagiaire s'est –il rapidement adapté à de nouvelles situations ? (A) B C D E F
(Proposition de solutions aux problèmes rencontrés, autonomie dans le travail, etc.)

Did the intern adapt well to new situations? (A) B C D E F
(eg. suggested solutions to problems encountered, demonstrated autonomy in his/her job, etc.)

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____
He was active in proposing solutions and evaluation methods

CULTUREL – COMMUNICATION / *CULTURAL – COMMUNICATION*

Le stagiaire était-il ouvert, d'une manière générale, à la communication ? (A) B C D E F
Was the intern open to listening and expressing himself /herself?

Souhaitez-vous nous faire part d'observations ou suggestions ? / *If you wish to comment or make a suggestion, please do so here* _____

OPINION GLOBALE / *OVERALL ASSESSMENT*

❖ La valeur technique du stagiaire était : (A) B C D E F
Please evaluate the technical skills of the intern:

III - PARTENARIAT FUTUR / *FUTURE PARTNERSHIP*

❖ Etes-vous prêt à accueillir un autre stagiaire l'an prochain ?

Would you be willing to host another intern next year? ☒ oui/*yes* ☐ non/*no*

Fait à Barcelona, le 30/08/2019
In _____, on _____

Signature Entreprise _____ Signature stagiaire
Company stamp _____ *Intern's signature*

INSTITUT DE ROBÒTICA I INFORMÀTICA INDUSTRIAL
C/ Llorens i Artigas, 4 - 6
08028 Barcelona Tel: 93.401.57.51
Mail: admin-iri@iri.upc.edu
CSIC

Merci pour votre coopération
We thank you very much for your cooperation

Version du 05/04/2019