Kanazawa Institute of Technology
金沢工業大学
JICA Trainee

NINJA ROBOT
FOR FARM LAND MANAGEMENT

Author:
HENRIQUE RENO SAWADA

Supervisor:
PROF. DR. TAKASHI KAWANAMI

October 2016

NINJA ROBOT
FOR FARM LAND MANAGEMENT

Final Report for the conclusion of the JICA Trainee program for Nikkei, presented in October 2016 in Kanazawa Institute of Technology (金沢工業大学).

Author:
Henrique Reno Sawada

Supervisor:
Prof. Dr. Takashi Kawanami

October 2016

Abstract

The main purpose of the project is to use different IoT (Internet of Things) technologies in farm land management, monitoring the conditions like temperature and humidity, being possible to follow and study the weather variations and the consequences into the plantation. Furthermore, it is possible to receive alerts when the conditions changes dangerously for the health of the plants and even to automate some measures to control and maintain the most perfect environment.

The main equipment studied and implemented was Raspberry Pi, Arduino and IMBLE BLE device, being programed in Python and C Language. Some sensors, actuators and cameras was studied during the project, and ThingSpeak web service was used to work with the data acquired.

All the codes and documentation of the project will be available at a GitHub repository [1].

Keywords: IoT (Internet of Things), Raspberry Pi, Arduino, BLE, ThingSpeak

TABLE OF CONTENTS

1 INTRODUCION

The Ninja Robot consists in a ropeway robot that can walk around over a greenhouse monitoring its environment conditions. Moreover, it can receive commands to control the main equipment of the greenhouse, for example activating the sprinkles to decrease the temperature or closing the rooftop when a heavy rain starts to fall. Beyond that, it is possible to analyze the data acquired by the sensors and automate some actions to execute, avoiding the need of the human interaction, using some key element conditions triggers or applying Deep Learning over the data.

In this project it is planned to use many different technologies:

- Arduino board in the sensors end,
- Arduino board in the actuators end,
- Raspberry Pi centralizing the communication,
- Camera attached to Raspberry Pi for monitoring,
- Arduino board controlling the movement of the robot,
- BLE technology in the communication between the equipment.

1.1 Motivation

One of the main components of the economy of Brazil is the agriculture [2]. It is estimated that the sector accounts for about 5.5% of Gross Domestic Product and employ 15.5% of the labor force. In comparison with Japan, it is 1.2% of GDP and employ 3.9% of the labor force [3].

Plantations require a lot of care and can be ruined by sudden disasters, and according to ONU almost 25% of the damaged caused by disaster affects directly the agriculture [4].

However, with automating the monitoring of the conditions of the greenhouse, is possible to stay alert 24 hours, 7 days per week (24/7) and take actions to minimize the loss much faster. Analyzing the data acquired for long periods, is possible to study ways to keep the best conditions for the plantation, improving the quality of the production.

With this project, all the knowledge learned in Japan can be directly applied in Brazil, making a good transfer of technologies between the countries.

1.2 Inspiration

In the early phases of the project, it was project to create a network of sensor distributed around the greenhouse and centralized in the Raspberry Pi. However, there was a huge complexity in the communication system, and there is no flexibility into the monitoring.

Later, was known about another project of a robot that walks in the greenhouse doing measures [5]. Inspired by this idea, the Ninja Robot project was created.

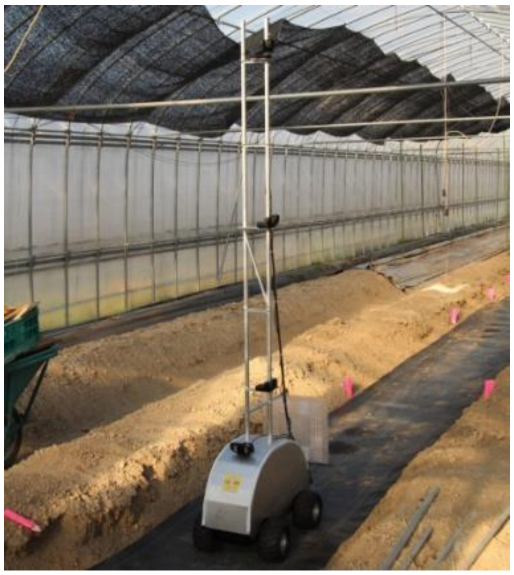


Figure 1 - Robot walking in the greenhouse

The advantages of using a walking robot inside the plantation is that it is not necessary the network of sensors, just one set of sensors in the robot is enough to monitor the entire place. This make the system much cheaper to build and make the maintenance, flexible to make readings anywhere and easily connect more sensors and actuators when necessary.

1.3 Main Goal

The main goal of the project is to develop the as much as possible of the Ninja Robot including all the main features. Due to the lack of enough time in a project with big complexity, some features will be partially implemented as a Concept Proof, and can be improved in future projects.

The list of the key points of the project:

- Communication between Raspberry Pi and Arduino using BLE,
- Arduino controlling sensors and actuators,
- Raspberry Pi receiving the data and providing the information in the internet,
- Raspberry Pi sending commands to control Arduino,
- Raspberry Pi acquiring images from the camera and providing to the end user.

1.4 Documentation

All the code and documents generated in this training will be available at a GitHub repository created for this project [1]. The owner of the repository is a generic created email with the following information:

Email:kit.kawalab@gmail.com
Password: kawalab2016

In this document just some main snippets, sometimes with non-relevant parts cut, will be shown, the majority of the implementation can be accessed in the GitHub repository.

2 ARCHITECTURE

Follow the example of the architecture expected within the project:

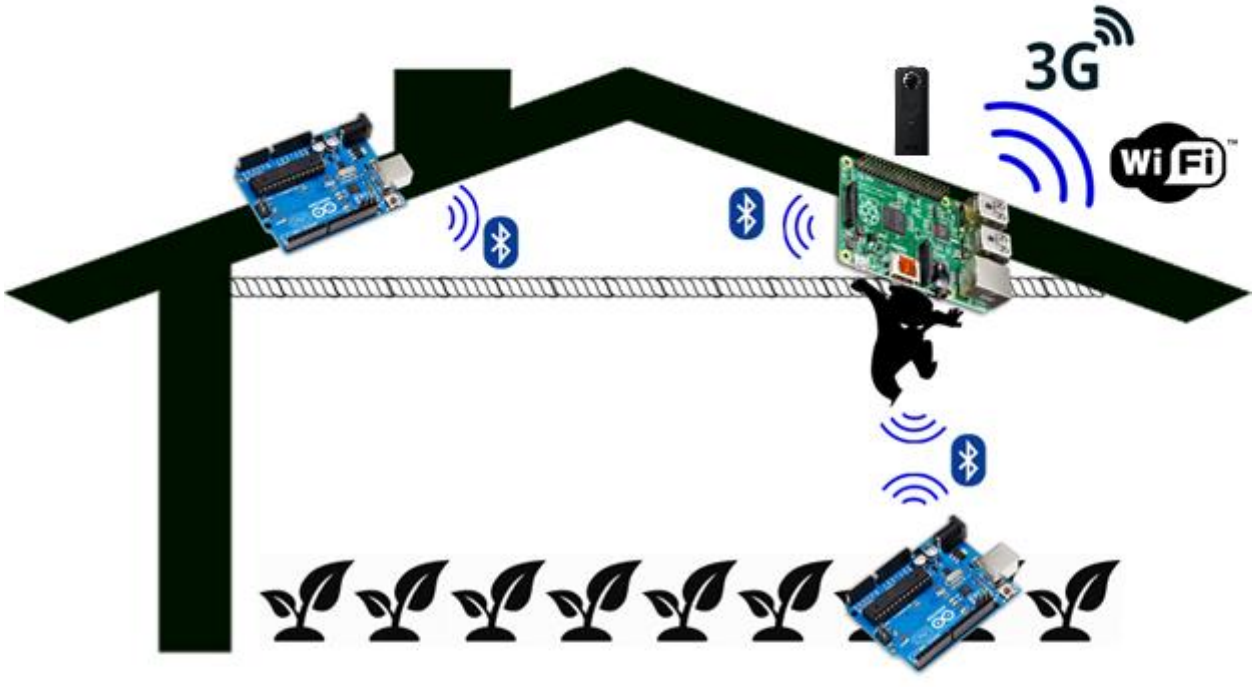


Figure 2 - Example of architecture in Greenhouse

In this architecture we have the Raspberry Pi board moving in the top of the greenhouse like a “ninja”. It has communication with the internet using Wi-Fi or 3G, being able to provide information and be controlled over the web.

The Arduino board is used mainly in the sensor node, that will be hanging from the Ninja Robot, and in the actuators node that can placed in the rooftop of the greenhouse to control the ceiling, or in the main control of the sprinkles for example.

The communication between equipment will be over BLE, that consumes very low power and makes the system have a good energy autonomy.

The camera will be linked to the Raspberry Pi, being positioned everywhere with the robot inside the plantation.

3 COMPONENTS OVERVIEW

Several components were used in this project, requiring a time for study and learn the properly use of those many different equipment. This chapter will briefly describe the main characteristics of hardware and software utilized in the project, and the methodology and references for study.

3.1 Arduino

According to the Arduino website [6] "*Arduino is an open-source prototyping platform based on easy-to-use hardware and software.*". There are several boards models available in the market nowadays, and in this project was used the Arduino UNO and Arduino Pro Mini.

With this powerful microcontroller board is very easy to start using sensor and actuators. The IDE Software is provided by free in the websites [6] [7], however due to legal questions in the trademark and fight between the creators, there are two main websites [8] [9]. The programing is in then Processing [10] language (very similar to C), and supports C and C++.

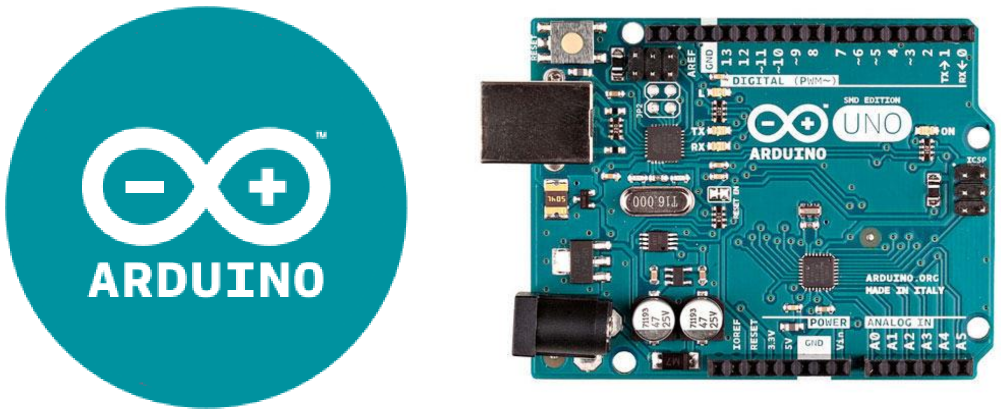


Figure 3 - Arduino Logo and Arduino UNO board

The main resource for studying the functionalities of the board was the book Arduino Cookbook [11], that is very complete and contains many useful examples. Furthermore, in the Arduino IDE is possible to access many examples of the libraries and is very convenient to study and copy the usage of the functions.

3.1.1 Sensors

Several sensors were used during the study phase, for example temperature sensor, light sensor, distance sensor and specially the BME280 [12] sensor, which is the one implemented in the Ninja Robot and can measure the temperature, humidity and atmospheric pressure.

The BME280 uses the I²C (Inter-Integrated Circuit) [13] communication protocol and has low power consumption, which is very important in IoT projects. Can easily communicate with Arduino using the Wire library and, because of the serial communication of the protocol, need just two data connections with the board to receive all measures. In the vendor website is possible to find the sample code and the wire connections [14].

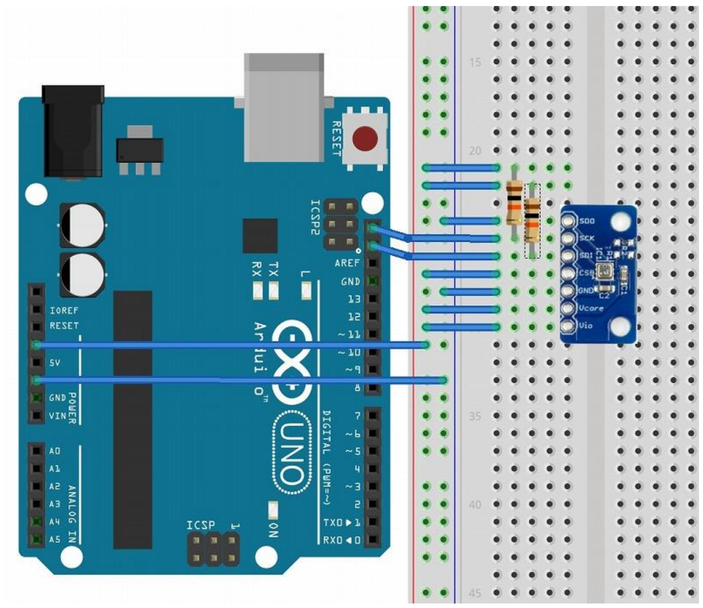


Figure 4 - Wire connection between Arduino and BME280 [14]

3.1.2 Actuators

Arduino can control different kinds of actuators from many different ways, using the digital and analogic ports, sometimes with the use of internal libraries. For example, it can blink a LED just sending an UP signal to the digital port, followed by a DOWN signal, with a time period between then. Is possible to control a servo motor positioning using its internal library, or send musical tones to a sound speaker. With this range of possibilities, the Arduino board will be responsible for controlling the movement of the Ninja Robot.

3.2 Raspberry Pi

In the Raspberry Pi website is defined: "*The Raspberry Pi is a credit card-sized computer that plugs into your TV and a keyboard. It is a capable little computer which can be used in electronics projects, and for many of the things that your desktop PC does, like spreadsheets, word processing, browsing the internet, and playing games. It also plays high-definition video.*" [15].

As a small computer, it has a higher complexity than Arduino and can perform more high level functionalities. Firstly, it needs an Operational System, and the most popular in use is the Raspbian, which is Linux based on Debian and is provided by the manufacturer. Another interesting OS is the Microsoft Windows IoT system [16], however it requires a machine with Windows 10 to develop.

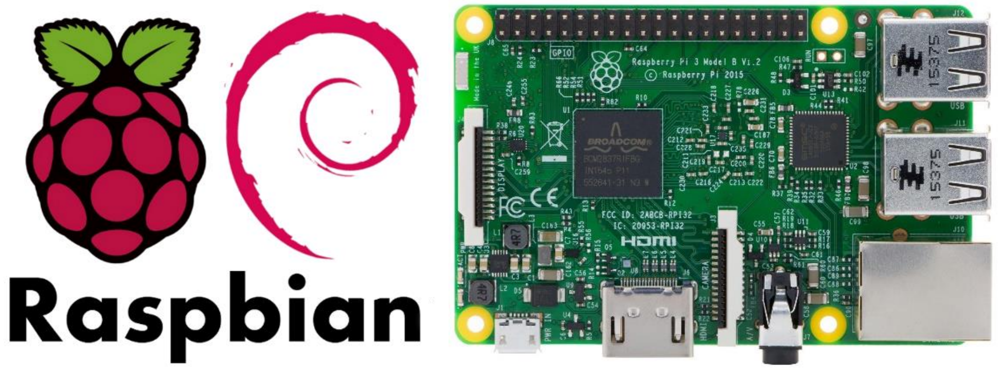


Figure 5 - Raspberry Pi Logo, Raspbian Logo and Raspberry Pi board

The system used in the project was the Raspbian. The main language used to program in the board is Python, due to it easiness, but it is possible to use many other languages and Linux shell scripts. The model of the board used in this project was the Raspberry Pi3 Model B.

During the study the books "Raspberry Pi Cookbook" [17] and "Raspberry Pi Cookbook for Python Programmers" [18] were used and are excellent resources to start using almost all the functionalities of the board.

3.3 PiCamera

The Raspberry Pi has a camera module that connects into a specific socket in the board and can be programmed in Python with its library PiCamera [19]. With the use of OpenCV (Open Source Computer Vision) [20] library is possible to analyze the image acquired by the camera and execute some functions like face recognition or movement track.

3.4 RICOH Theta S (360° Camera)

This camera is described by its vendor as: “*High-spec model that captures all of the surprises and beauty from 360°*” [21]. It can generate 360° images and videos, being an excellent resource for creating a monitoring camera for the environment.

The camera consists in two “fisheye” lens that generates two spherical images. Using the application from the manufacturer, those images are converted into a single 360° environment (equirectangular mode). The camera is compliant with Open Spherical Camera API Version 2.0 [22] from Google, than it is possible to upload videos directly to YouTube [23] and take advantage of its 360 degree control.

The camera has a very helpful developers community [24] [25], where is possible to find sample codes and explanations of how to use the camera.

Figure 6 - RICOH Theta S Camera

3.5 BLE (Bluetooth Low Energy)

Bluetooth Low Energy, or BLE (marketed as Bluetooth Smart) is a wireless personal area network technology based on Bluetooth 4.0, aimed for many applications like beacons, security, home entertainment [26]. The main difference between BLE and Bluetooth is the cost and power consumption.

The reduced size and the low power needed by the device provides a huge energy autonomy, and in the future this autonomy can be 100% using solar cell [27]. As its protocol is merged with Bluetooth 4.0, has a great compatibility with modern computers and smartphones. With all those characteristics it is said that BLE was built for the Internet of Things (IoT) [28].

To communicate with BLE it is used the Generic Attribute Profile (GATT), and is based in a client-server communication. It is possible to read data requesting the handle with the data and to send information writing in the specific handle.

In this project was utilized the IMBLE BLE [29] that was a bit challenging as the datasheet is only in Japanese and is difficult to find information of the usage. To prototype and connect the device with the Arduino, it was necessary an adapter that the manufacturer provides too.

Figure 7 - IMBLE BLE device

The communication with the device is via serial protocol in the server side, i.e. the side using the BLE device to send data information; and in the client side it is limited to 16 bytes transfers through the Bluetooth protocol.

The connection with Arduino is very simple, just being necessary the VCC (3.3V), GND, TX and RX connections. But it is important to set some pins with appropriate values, for example the SLEEP REQUEST and MODE (Data Mode) pins need to stay connected to the ground and is useful to have a button to activate the RESET pin when necessary.

3.6 ThingSpeak

"*ThingSpeak is an Internet of Things (IoT) platform that lets you collect and store sensor data in the cloud and develop IoT applications*." [30]. It is a web service that "*collect, store, analyze, visualize, and act on data from sensors or actuators, such as Arduino®, Raspberry Pi™, BeagleBone Black, and other hardware.*".

Created by MathWorks Company, it is integrated with some MATLAB function that brings the possibility of analyzing the data, make calculations like average or dew point, eliminate data outliers, etc. It is possible to create alerts and visualizations with the processed data, and then take actions manually or automatically when necessary.

The system is based in Channels created by the users that are filled with information provided by IoT devices. Those channels can be Public or Private, and is free to use.

The protocol to send and receive data is HTTP and can be used with XML or JSON requests. To help the implementation there is an official library for Arduino and non-official library for Python implementations [31].

Figure 8 - Logo of ThingSpeak and MathWorks

4 RESULTS AND DISCUSSION

This chapter will approach each part of the implementation with details and explanations, discussing in the end about the results.

4.1 BLE protocol

The BLE device utilized in the project was the IMBLE BLE [29] from Interplan Company. This device uses a Serial Communication with the board that is controlling it, in this case the Arduino board; and the BLE protocol with other devices.

The size of the messages in the BLE protocol are 16 bytes, either for sending or receiving. That way, it was created a message protocol within those 16 bytes to receive all sensors data and send commands to the Arduino board.

Firstly, all the data being send and received can be made only numbers, so we can use the nibbles of the bytes, expanding the amount of information to 32 characters in each message. Then, to avoid synchronization problem between messages, as the communication is asynchrony, all the necessary data will be fitted inside one message. If in future implementations the 32 characters is not enough, will be necessary to implement synchronization messages to acquire all data in multiple calls.

Follow the diagram of the first version of the protocol for received messages:

B1		B2		B3		B4		B5		B6		B7		B8	
N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16
ReadID				Temperature						Pression					

B9		B10		B11		B12		B13		B14		B15		B16	
N17	N18	N19	N20	N21	N22	N23	N24	N25	N26	N27	N28	N29	N30	N31	N32
Humidity						X		Y		Z		Reserved			

Figure 9 - BLE Protocol V1 Receive

The picture above shows that all measures will have 6 digits, the position of the robot will be determined by 2 digits in each axis and each measure acquired will have an ID of 4 digits.

Supposing that all numbers of measures are received with two decimal places, we can check the example:

B1		B2		B3		B4		B5		B6		B7		B8	
N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16
0	0	1	1	0	0	2	5	4	6	1	0	0	5	9	9
ReadID				Temperature						Pression					

B9		B10		B11		B12		B13		B14		B15		B16	
N17	N18	N19	N20	N21	N22	N23	N24	N25	N26	N27	N28	N29	N30	N31	N32
0	0	5	6	1	1	2	2	5	5	9	9				
Humidity						X		Y		Z		Reserved			

Figure 10 - Example BLE Protocol V1

In this message it was received the reading with ID 0011 and the values: Temperature of 25.46°C, Atmospheric Pressure of 1005.99 hPa, 56.11% of Humidity and Position (22,55,99).

Next, the protocol for send messages:

B1		B2		B3		B4		B5		B6		B7		B8	
N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16
SendID				CommandID				Arguments							

B9		B10		B11		B12		B13		B14		B15		B16	
N17	N18	N19	N20	N21	N22	N23	N24	N25	N26	N27	N28	N29	N30	N31	N32
Arguments															

Figure 11 - BLE Protocol V1 Send

In this protocol, the SendID is to avoid duplicate commands to being executed for any communication problem, so the board will ignore two commands consecutives with the same ID. The CommandID consists in a list of commands supported by the Arduino board implementation, followed by the arguments necessary for the command. Example:

B1		B2		B3		B4		B5		B6		B7		B8	
N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	N14	N15	N16
0	0	2	2	0	0	0	1	0	0	1	0	0	0		
SendID				CommandID				Arguments							

B9		B10		B11		B12		B13		B14		B15		B16	
N17	N18	N19	N20	N21	N22	N23	N24	N25	N26	N27	N28	N29	N30	N31	N32
Arguments															

Figure 12 - Example BLE Protocol V1 Send

In this example the command with ID 0001 is send to with the arguments 001000. If we define that this command is to Arduino call a function to blink a LED and the argument is the time interval, the LED will start blinking every second after receiving this command.

4.2 Arduino and BLE

The Arduino integration with the IMBLE BLE is via Serial Port. In the datasheet [29] is possible to verify the possible commands accepted by the BLE device. To be possible to keep debugging the results of Arduino, which by default is made via Serial Port Communication with the PC, it was used “SoftwareSerial” library to create a serial communication in port 10 and 11 with the BLE device.

```
// using port 10/11 to receive from BLE, this way can use normal serial port for debug
// SoftwareSerial portOne(10, 11);

void setup() {
  Serial.begin(9600);          // initialize serial communication at 9600 bits per second
  portOne.begin(19200);     // initialize serial with BLE at 19200 bauds
  (…)
}
```

As the protocol defined before expects a fixed size data, a function to do padding with the numbers was necessary, including leading zeros in the beginning of the string. If the number is too big to fit the 6 characters, it will not fill with any number to indicate the error.

```
// function to conv the number in fixes size string
String convInt2StringFixedSize (long int num, int fsize)
{
  String result = String(num);
  // while string smaller than size, fill with zeros
  while (result.length() < fsize)
    result = "0" + result;
  // if string is too big, error: become zero
  if (result.length() > fsize)
    result = convInt2StringFixedSize (0, fsize);
  return result;
}
```

To send the message is just necessary to append the command “TXDA ” and the mounted string.

In the main loop first is verified if there is data available to be read, and all data received is analyzed and treated line by line. If the line contains the “CONECT” string, the global Boolean indicating that the connection is on is updated and then can start sending the data from the sensors. If and bad Acknowledge “NG” or a Disconnection string “DISCON”, the indicator of connection is disabled.

Case the string received starts with the data header “00,0000,00”, the program needs to parse the fields of the command and execute the determined function.

```
// verify if there is data to receive
  while (portOne.available() > 0) {
    String recv = portOne.readStringUntil('\n');
    // update connection status
    if (recv.startsWith(Conect)) conn = true;
    if (recv.startsWith(Discon)) conn = false;
    if (recv.startsWith(Ack_NG)) conn = false;

    // treatment of commands
    if (recv.startsWith(Comand))
    {
      // parse and execute the command
      (…)
    }
  }
```

4.3 Raspberry Pi and BLE

To use the BLE functionality from the Raspberry Pi, it is necessary to use the Generic Attributes (GATT) protocol.

When using the command line, we can use the gatttool, passing the MAC address of the BLE device and the –I argument for interactive commands (the BLE address can be acquired sending the command "RDAD" through a serial port to the device):

```
$ gatttool -b 80:7B:85:A0:00:EA -I
```

In the interactive mode is possible to connect, get the list of handles available from the device, read and send messages:

```
[80:7B:85:A0:00:EA][LE]> connect
Attempting to connect to 80:7B:85:A0:00:EA
Connection successful
[80:7B:85:A0:00:EA][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
(…)
handle: 0x001a, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001b, uuid: ada99a7f-888b-4e9f-8081-07ddc240f3ce
handle: 0x001c, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x001d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001e, uuid: ada99a7f-888b-4e9f-8082-07ddc240f3ce
[80:7B:85:A0:00:EA][LE]> char-read-hnd 0x001b
Characteristic value/descriptor: 11 00 00 00 04 17 00 33 74 10 16 87 03 98 09 99 55 33 00 00
[80:7B:85:A0:00:EA][LE]> char-write-req 0x001e 12
Characteristic value was written successfully
```

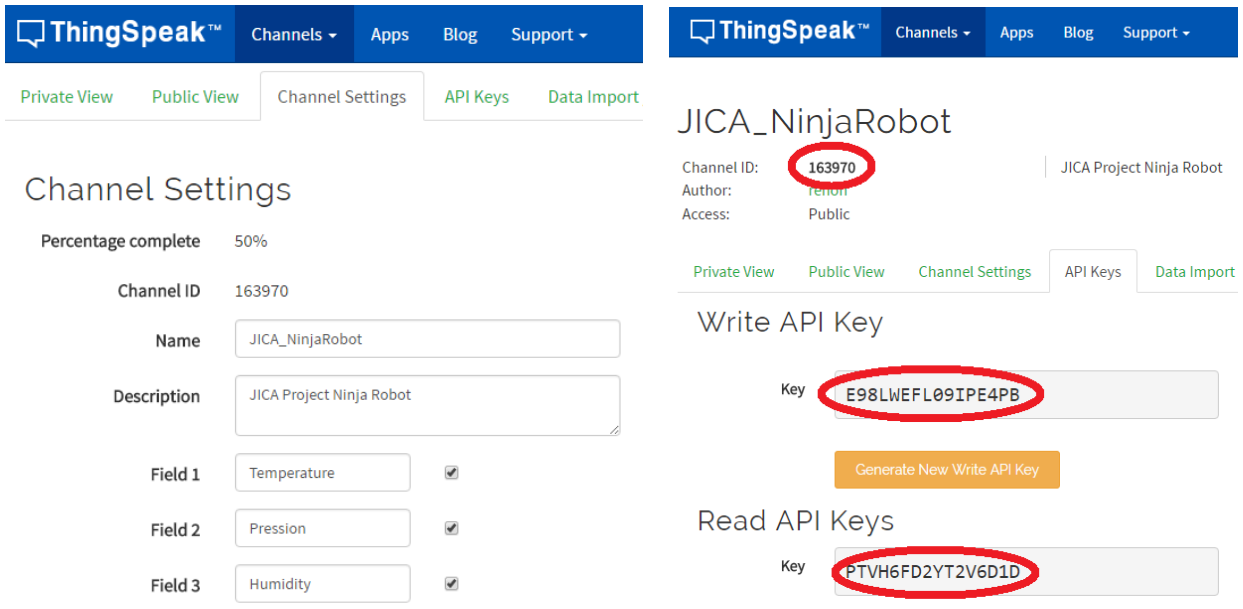
The reading and write handles were acquired reading the code of Android apk from the manufacturer [29].

To make a Python program that can use the BLE resources, it was necessary to install the "pygattlib" library [32]. In the website there is the step by step how to install and some example codes.

When sending and receiving data to IMBLE BLE, is necessary to make a treatment of the deprecated header of their protocol, composed by three zero bytes after the size of the message. Attention: the size of the message will be in hexadecimal format, it is filled by the device and does not make part of the protocol determined by this project.

4.4 ThingSpeak Configuration

The initial configuration of ThingSpeak is very easy and trivial. First, create the channel [33] with the description and the fields that will be received, and take note of the important information as Channel ID, Write/Read API Key.



Channel Settings

Percentage complete	50%
Channel ID	163970
Name	JICA_NinjaRobot
Description	JICA Project Ninja Robot
Field 1	Temperature ✓
Field 2	Pression ✓
Field 3	Humidity ✓

JICA_NinjaRobot

Channel ID:	163970
Author:	fenon
Access:	Public

JICA Project Ninja Robot

Write API Key

Key E98LWEFL09IPE4PB

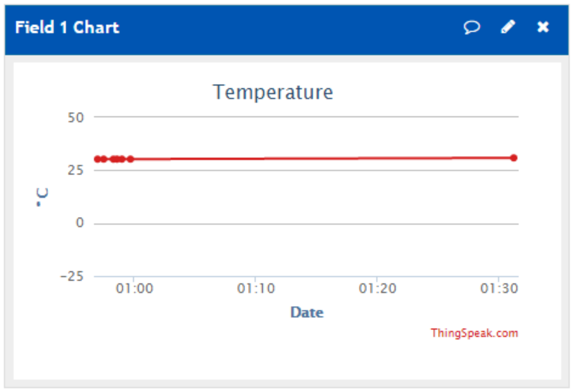
Read API Keys

Key PTVH6FD2YT2V6D1D

Figure 13 - ThingSpeak Initial Setup

Then is possible to create visualization to monitor the data received. The views can be public or private. The user can create and customize their own view and plugging, or can use pre-made apps like the Google Gauge for example.

To create alerts and actions is necessary using the Apps tab. There you can take simple decisions based on the values from the measures, and do some actions like tweeting or making http request to some other application that can make alerts or do some automation accordingly with the alert received.



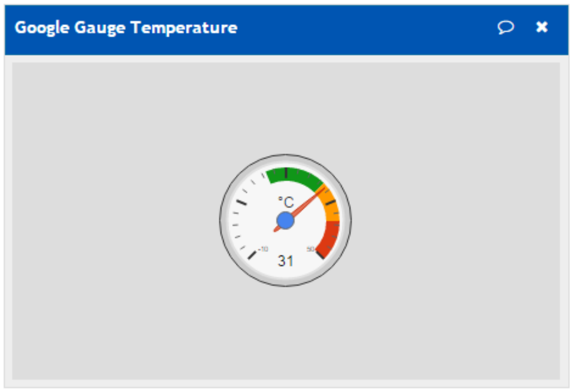


Figure 14 - ThingSpeak example of visualizations

4.5 Raspberry Pi and ThingSpeak

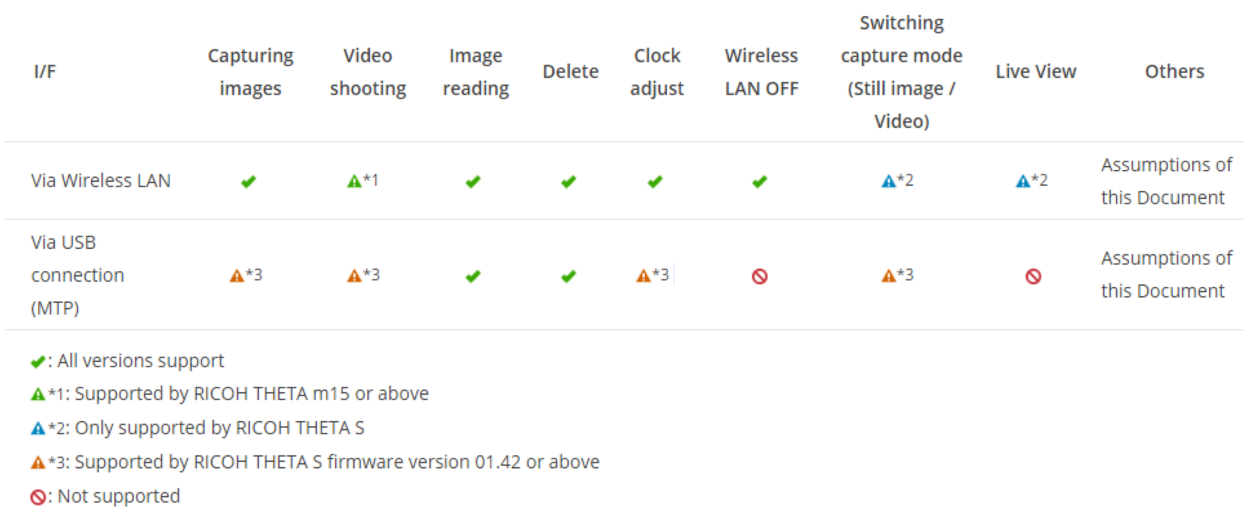
The interface with ThingSpeak is via http POST calls, and to use in python there is a library "thingspeak" [31] that can be used. The API claims that it can only receive one call each 15 seconds, however in tests we could send data in shorter intervals.

```
# send data to ThingSpeak
    channel = thingspeak.Channel(id=channel_id,write_key=write_key)
    try:
        response =
channel.update({1:protocolV1['Temperature'],2:protocolV1['Pression'],3:protocolV1['Humidity']})
    except:
        print "ThingSpeak Error - connection failed"
    print response
```

4.6 Raspberry Pi and RICOH Theta S

The RICOH Theta S camera can be accessed by three different ways: Wi-Fi connection, USB connection or USB Live Streaming Connection.

The diagram below show the differences between Wi-Fi mode and USB mode (not included Live Streaming Mode in comparison)



I/F	Capturing images	Video shooting	Image reading	Delete	Clock adjust	Wireless LAN OFF	Switching capture mode (Still image / Video)	Live View	Others
Via Wireless LAN	✓	▲*1	✓	✓	✓	✓	▲*2	▲*2	Assumptions of this Document
Via USB connection (MTP)	▲*3	▲*3	✓	✓	▲*3	⊘	▲*3	⊘	Assumptions of this Document

✓: All versions support
▲*1: Supported by RICOH THETA m15 or above
▲*2: Only supported by RICOH THETA S
▲*3: Supported by RICOH THETA S firmware version 01.42 or above
⊘: Not supported

Figure 15 - Supported features in RICOH Theta API

4.6.1 Wi-Fi Connection

The Wi-Fi protocol is based on HTTP requests with the body in JSON format. There is no official library developed yet, and in the community there are some different implementations in Python, but none is completed in the moment. That way, mixing those examples in the community a library was partially implemented to communicate with the camera.

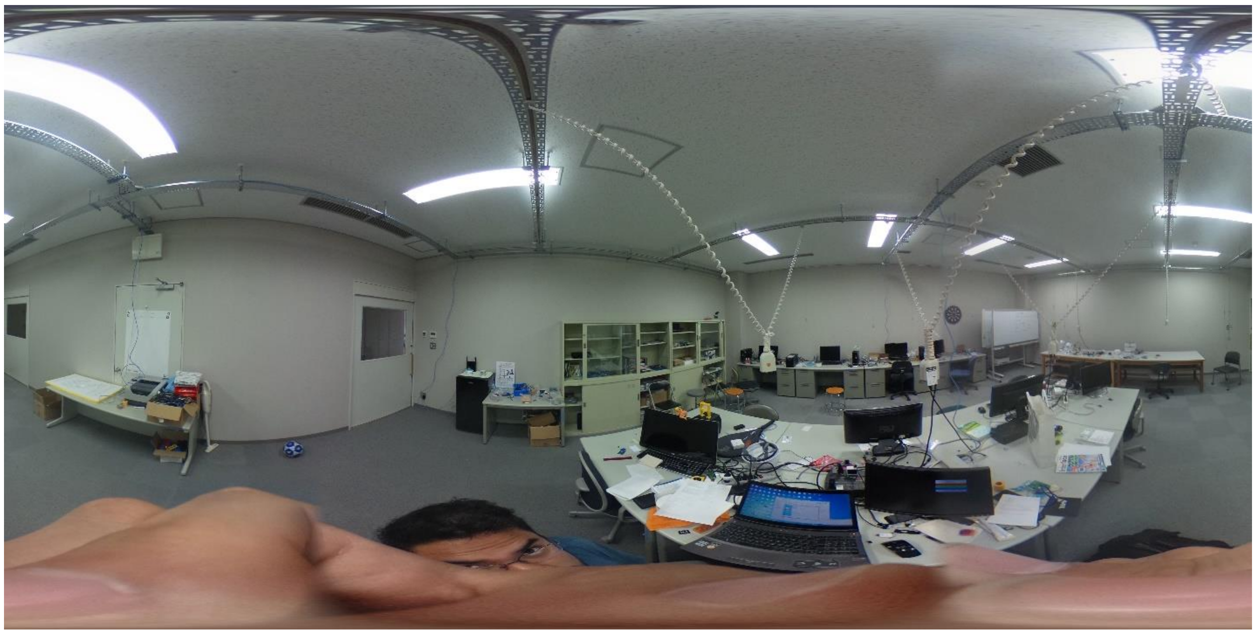


Figure 16 - Example of image from RICOH Theta S Camera

The good points of this mode is the mobility as the camera is wireless connected to the device and has the Live Preview mode, which preview the image before taking pictures, very useful for monitoring applications that does not need to record all the time. In the other hand, the disadvantage of this connection is that it needs another Wi-Fi adapter if the board needs the Wi-Fi connection to communicate with the internet.

4.6.2 USB Connection

The USB connection uses the MTP protocol. The advantages of this method is because file transfer is much faster, the camera can stay powered during all the time and the Wi-Fi connection of the board stays vacated. However, this mode lacks the Live View mode, and there are few implementations of MTP protocol in Python.

4.6.3 USB Live Streaming Connection

To turn on the camera in the USB Live Streaming mode it is necessary to press the down button while turning on the camera, then it will turn on the Live icon.

In this mode, the camera can be accessed like a normal USB webcam. This is very useful for surveillance systems as the one we intend to apply in this project.

However, the image generated is not equirectangular, being obtained two spherical images. This way it is necessary some software implementation to process and transform those images into a more plane image possible to navigate.

In Raspberry Pi, the camera is recognized as an UVC Camera. The tools “fswebcam”, “motion” and “luvcview” are useful to control and view the camera in the command line.

```
$ sudo apt-get install fswebcam, motion, luvcview
$ # take image
$ sudo fswebcam image.jpg
$# initiate camera and take pictures when some motion is detected
$ motion
$ # view the camera image
$luvcview
```

Follow an example of the image obtained:

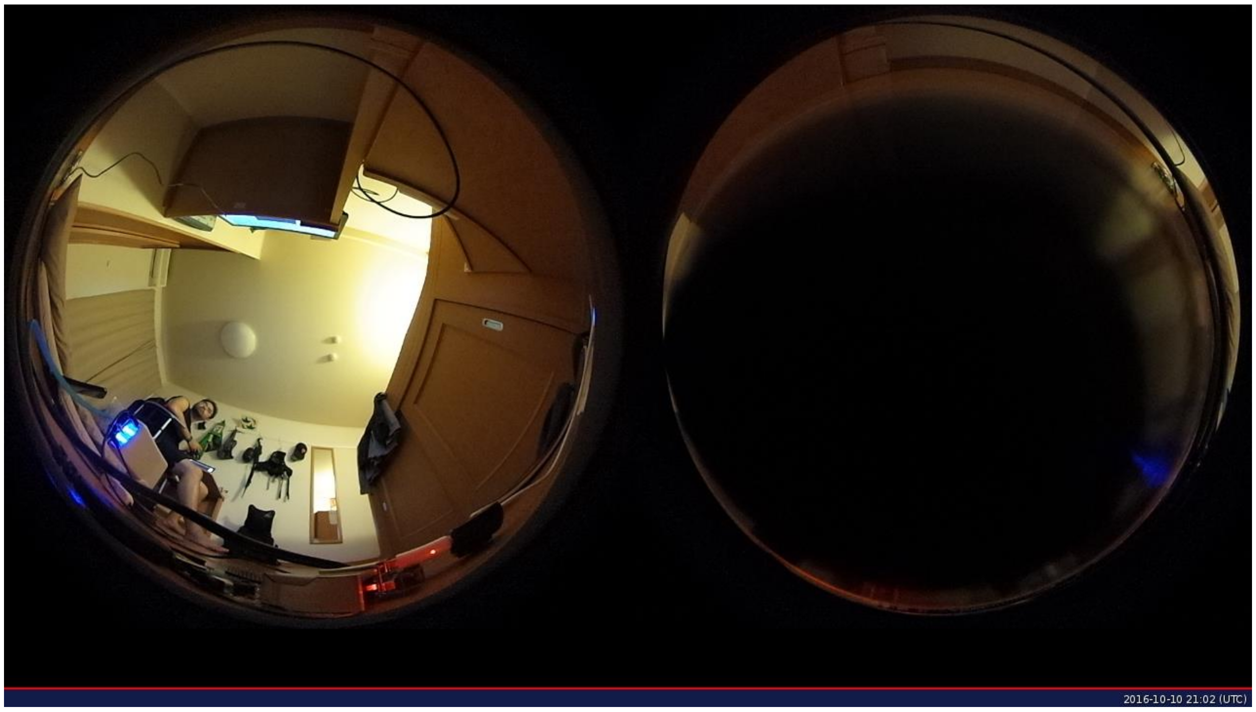
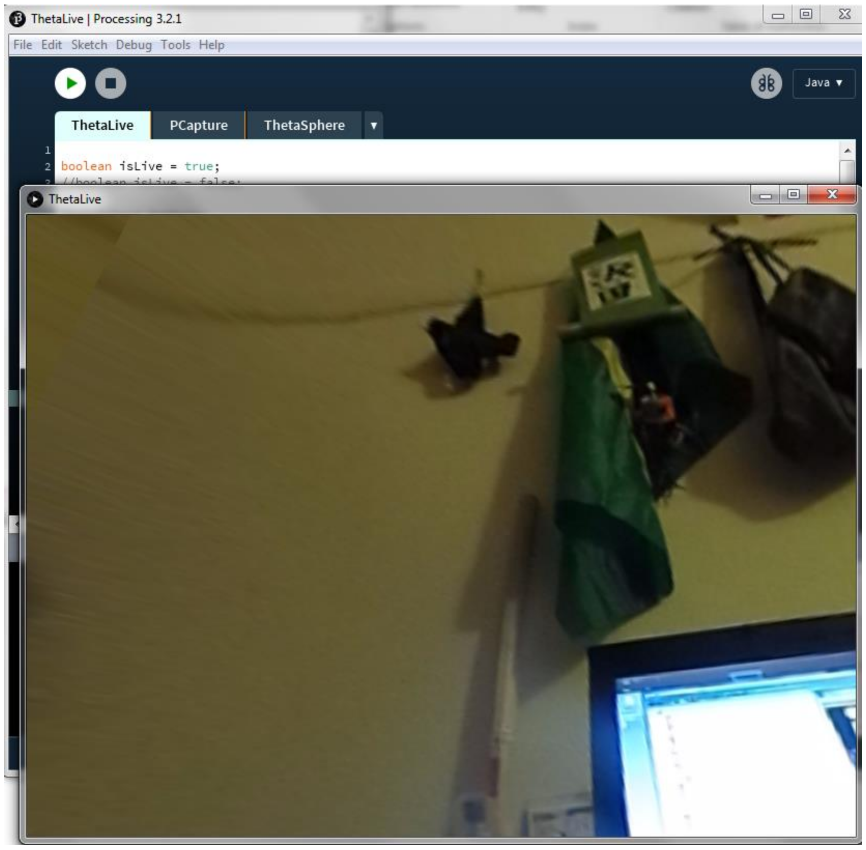


Figure 17 - Example of image from RICOH Theta S Camera via Live Streaming

In the developers community [25] there is an example using processing to get the video and transform into a plane image that is navigable. To it works with Raspberry Pi it is necessary to update the java library used to get the camera [33] and to increase the GPU amount of memory of the Raspberry Pi in “sudo raspi-config”. However, the processing of the image is very slow and is not very useful, but it proves that is possible to treat the image received directly from the Live Stream mode of the camera.


ThetaLive | Processing 3.2.1
File Edit Sketch Debug Tools Help
Java
ThetaLive
PCapture
ThetaSphere
ThetaLive

```
boolean isLive = true;
//boolean isLive = false;
```

Figure 18 - Processing Example

To work with the images programmatically, the best approach should use OpenCV [20] library for Python. However, during the preliminary tests the image showed different colors and positioning when showed in the UI. With more time of project would be possible to fix the image and process it to become plane, creating an

4.7 Practical Tests and Results

To simplify the moving system it was used プラレール (Plarail) just to prove the concept. Ideally the system would be mounted over the laboratory structure, but as the system is still using protoboards, Arduino UNO (in future should be migrated to Arduino Pro Mini) and big USB battery power source, the was not safe to build the whole system and make it moving.

In this case, two different tests were made:

4.7.1 Moving train with Raspberry Pi, controlling the stop train actuator via BLE

Ideally, the system should detect it stopped analyzing images from the camera, and then send a signal to the actuator to release the bridge again so it can start moving. However, there was no time for concluding the image detection, so it was used the luminous sensor signal send via BLE to the Raspberry, and then it decides the moment to send the signal to keep moving.

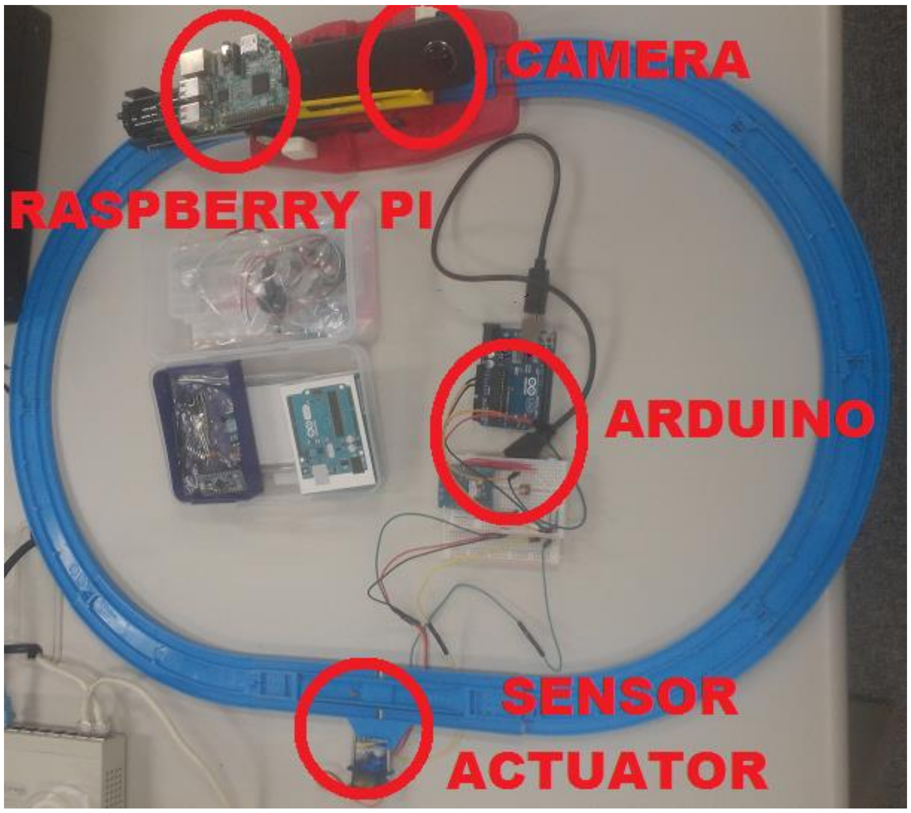


Figure 19 - Experimental part 1

4.7.2 Raspberry Pi receiving data from the BME280 sensor through Arduino and BLE

The BME280 sensors connected to Arduino and sending measures to Raspberry Pi via BLE connection is the most close to the sensor node we expected in the Ninja Robot. With all the received data being send to ThingSpeak, it is possible to make analysis and create alerts over the measures obtained.

5 PROPOSAL FOR FUTURE RESEARCH

During the implementation of the project, some other concerns appeared that could improve the project in future researches, and some points showed to be bigger than imagined and can become future projects.

5.1 Positioning Mapping

It is necessary to track the position of the robot over the greenhouse to acquire data with precision and to navigate through key point. Maybe using RFID tags in key points, the robot could detect easily when it get in those places and could freely navigate through the entire greenhouse.

5.2 User Interface – Web or Local

An interface was started in the final phase of the project, however there was no time to make something very useful. Instead of local interface, a web user interface would help to control the equipment over the internet.

5.3 Treatment of the images from the 360° Canera

The images from the 360° camera follow the Open Spherical Camera API from Google, making possible to create user interfaces to interact and navigate through the environment. Analyzing the images can be crucial to detect climate changes and even plagues in the plantation.

5.4 Creating a case for the system to work in field tests

Creating a case for the system that works in agriculture is very hard, it needs to be water proof, resistant to many temperatures and to solar rays. Beyond that, it needs to still make accurate measures of the environment, so the case can not interfere in the data acquiring.

6 CONCLUSION

The main goal of the project was learning new technologies and making good use of then, bringing a big amount of knowledge back to my country. Undoubtedly this goal was achieved as there was a big phase of study of many different kinds of equipment.

Despite some functionalities were not implemented due to the lack of time and the size of the project, all the main features were almost fully programed or at least proved the possibility. Unfortunately, some very interesting items could not be fully implemented and are commented in the previous chapter, and it was not possible to make real field tests.

The communication with the Bluetooth Low Energy was succeeded and was one of the key points of the project, making possible to acquire the measures data and to send commands to actuators, becoming the basis of the whole project. It will save time for future researches using the same components as they can be reused and adapted.

The images were obtained from the camera, but it still needs more study to work with then, for navigating through the 360° images and making analysis over it. It can be an entire future project, since it is a huge area to develop.

It was very profitable to study and learn so many different components and very challenging to make use of some newest technologies that are not very mature yet, for example the BLE and the 360° Camera.

7 REFERENCES

[1] GitHub, "JICA Ninja Robot Repository," 2016. [Online]. Available: https://github.com/KIT-Kawalab/JICA_2016. [Accessed 2016].

[2] Wikipedia, "Economy of Brazil," [Online]. Available: https://en.wikipedia.org/wiki/Economy_of_Brazil. [Accessed 2016].

[3] Wikipedia, "Economy of Japan," [Online]. Available: https://en.wikipedia.org/wiki/Economy_of_Japan. [Accessed 2016].

[4] ONU BR, "Agricultura é o setor mais afetado por desastres, diz relatório da FAO," 18 March 2015. [Online]. Available: https://nacoesunidas.org/agricultura-e-o-setor-mais-afetado-por-desastres-diz-relatorio-fao/. [Accessed 2016].

[5] NikkeiBP, "農業の未来変える切り札ロボット、頭脳は 5000 円の「ラズパイ」だった," 3 June 2016. [Online]. Available: http://www.nikkeibp.co.jp/atcl/tk/DTrans/ecs/021500015/. [Accessed 2016].

[6] Arduino S.R.L., "Arduino," [Online]. Available: http://www.arduino.org/. [Accessed 2016].

[7] Arduino LLC, "Arduino," [Online]. Available: https://www.arduino.cc/. [Accessed 2016].

[8] H. Barragán, "The Untold History of Arduino," [Online]. Available: http://arduinohistory.github.io/. [Accessed 2016].

[9] E. Williams, "ARDUINO VS. ARDUINO," 25 February 2015. [Online]. Available: http://hackaday.com/2015/02/25/arduino-v-arduino/. [Accessed 2016].

[10] Processing, "Processing," [Online]. Available: https://processing.org/. [Accessed 2016].

[11] M. Margolis, in *Arduino Cookbook, 2nd Edition*, O'Reilly Media, Inc., 2011.

[12] Switch Science, "BME280," [Online]. Available: https://www.switch-science.com/catalog/2236/. [Accessed 2016].

[13] Wikipedia, "I²C," [Online]. Available: https://en.wikipedia.org/wiki/I%C2%B2C. [Accessed 2016].

[14] Switch Science, "BME280 Wiki," [Online]. Available: http://trac.switch-science.com/wiki/BME280. [Accessed 2016].

[15] Raspberry Pi Foundation, " Raspberry Pi," [Online]. Available: https://www.raspberrypi.org. [Accessed 2016].

[16] Microsoft, "Windows IoT," [Online]. Available: https://developer.microsoft.com/en-us/windows/iot. [Accessed 2016].

[17] S. Monk, in *Raspberry Cookbook*, O'Relly Media, Inc., 2014.

[18] T. Cox, in *Raspberry Pi Cookbook for Python Programmers*, Birmingham, Packt Publishing Ltd..

[19] Raspberry Pi Foundation, "PiCamera," [Online]. Available: https://www.raspberrypi.org/learning/getting-started-with-picamera/. [Accessed 2016].

[20] itseez, "OpenCV," [Online]. Available: http://opencv.org/. [Accessed 2016].

[21] Ricoh Company, Ltd., "Theta360," [Online]. Available: https://theta360.com/en/about/theta/s.html. [Accessed 2016].

[22] Google, "Open Spherical Camera API Version 2.0," [Online]. Available: https://developers.google.com/streetview/open-spherical-camera/. [Accessed 2016].

[23] Google, "YouTube," [Online]. Available: https://www.youtube.com/. [Accessed 2016].

[24] J. a. O. C. Casman, "THETA Unofficial Guide," [Online]. Available: http://theta360.guide/.

[25] GitHub, "THETA 360 Developer Community," [Online]. Available: https://github.com/theta360developers. [Accessed 2016].

[26] Wikipedia, "Bluetooth low energy," [Online]. Available: https://en.wikipedia.org/wiki/Bluetooth_low_energy. [Accessed 2016].

[27] Fujitsu, "Fujitsu Develops Industry's First Flexible IoT-Supporting Beacon That Needs No Battery Replacement," 25 March 2015. [Online]. Available: http://www.fujitsu.com/global/about/resources/news/press-releases/2015/0325-02.html. [Accessed 2016].

[28] Bluetooth, "Low Energy," [Online]. Available: https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics/low-energy. [Accessed 2016].

[29] interplan Co.,Ltd., "IMBLE BLE Device," [Online]. Available: http://www.interplan.co.jp/solution/wireless/imble.php. [Accessed 2016].

[30] The MathWorks, Inc., "ThingSpeak," [Online]. Available: https://www.mathworks.com/help/thingspeak/. [Accessed 2016].

[31] M. Chwalisz, "thingspeak python," [Online]. Available: https://pypi.python.org/pypi/thingspeak/. [Accessed 2016].

[32] O. Acena, "pygattlib," [Online]. Available: https://bitbucket.org/OscarAcena/pygattlib. [Accessed 2016].

[33] ThingSpeak, "JICA_NinjaRobot Channel," [Online]. Available: https://thingspeak.com/channels/163970. [Accessed 2016].

[34] Maven, "Sarxos Drivers," [Online]. Available: http://search.maven.org/#search%7Cga%7C1%7Cg%3A%22com.github.sarxos%22. [Accessed 2016].