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 Nikkeys, 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 RasberryPi, Arduino and IMBLE BLE device, being programed in Python and C Language. Some sensors, actuators and cameras was studied during the project, and ThingSpeak webservice was used to work with the data acquired.

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

[kit.kawalab@gmail.com](mailto:kit.kawalab@gmail.com) – kawalab2016

**Keywords**: IoT (Internet of Things), RaspberryPi, Arduino, BLE, ThingSpeak

**TABLE OF CONTENTS**

[1 INTRODUCION 3](#_Toc464031672)

[1.1 Motivation 3](#_Toc464031673)

[1.2 Inspiration 3](#_Toc464031674)

[1.3 Advantages 3](#_Toc464031675)

[2 ARCHITECTURE 4](#_Toc464031676)

[3 COMPONENTS OVERVIEW 5](#_Toc464031677)

[3.1 Arduino 5](#_Toc464031678)

[3.1.1 Sensors 6](#_Toc464031679)

[3.1.2 Actuators 6](#_Toc464031680)

[3.2 Raspberry Pi 7](#_Toc464031681)

[3.3 PiCamera 7](#_Toc464031682)

[3.4 RICOH Theta S (360° Camera) 8](#_Toc464031683)

[3.5 BLE (Bluetooth Low Energy) 8](#_Toc464031684)

[3.6 ThingSpeak 9](#_Toc464031685)

[4 RESULTS AND DISCUSSION 10](#_Toc464031686)

[4.1 BLE protocol 10](#_Toc464031687)

[4.2 Arduino and BLE 12](#_Toc464031688)

[4.3 Raspberry Pi and BLE 13](#_Toc464031689)

[4.4 ThingSpeak Configuration 15](#_Toc464031690)

[4.5 Raspberry Pi and ThingSpeak 15](#_Toc464031691)

[4.6 ThingSpeak Results and Alerts 15](#_Toc464031692)

[4.7 Raspberry Pi and RICOH Theta S 15](#_Toc464031693)

[5 PROPOSAL FOR FUTURE RESEARCH 16](#_Toc464031694)

[6 CONCLUSION 17](#_Toc464031695)

[7 REFERENCES 18](#_Toc464031696)

# INTRODUCION

## Motivation

## Inspiration

## Advantages

# ARCHITECTURE

Follow the example of the architecture expected with the project:

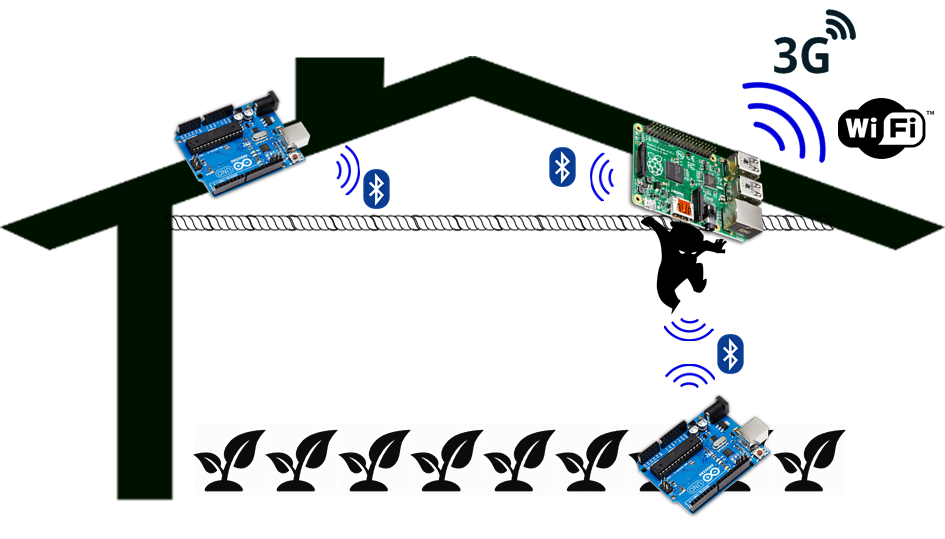


Figure 1 - Example of architecture in Greenhouse

# 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.

## Arduino

According to the Arduino website [2] “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 [2] [3], however due to legal questions in the trademark and fight between the creators, there are two main websites [4] [5]. The programing is in then Processing [6] language (very similar to C), and supports C and C++.



Figure 2 - Arduino Logo and Arduino UNO board

The main resource for studying the functionalities of the board was the book Arduino Cookbook [7], 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.

### Sensors

Several sensors were used during the study phase, for example temperature sensor, light sensor, distance sensor and specially the BME280 [8] 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) [9] 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 [10].

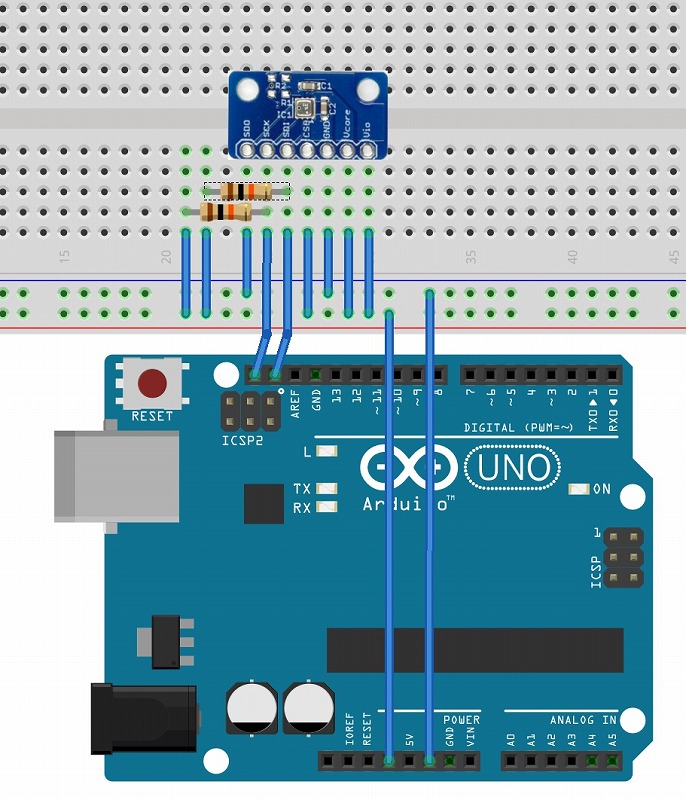


Figure 3 - Wire connection between Arduino and BME280 [10]

### 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.

## 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.” [11].

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 [12], however it requires a machine with Windows 10 to develop.

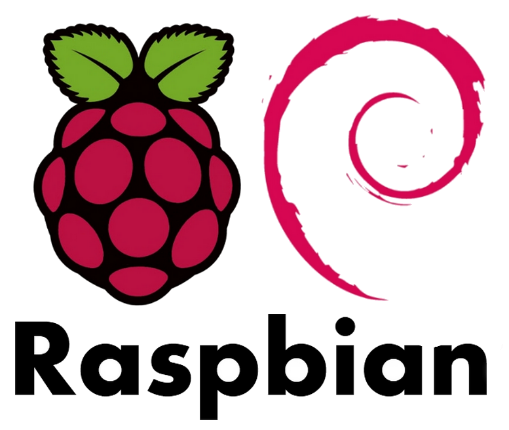
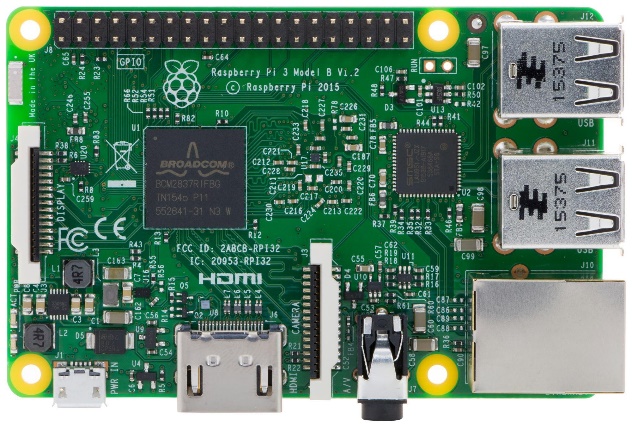
 

Figure 4 - Raspberry Pi Logo, Raspbian Logo and Raspberry Pi board

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” [13] and “Raspberry Pi Cookbook for Python Programmers” [14] were used and are excellent resources to start using almost all the functionalities of the board.

## 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 [13]. With the use of OpenCV (Open Source Computer Vision) [14] library is possible to analyze the image acquired by the camera and execute some functions like face recognition or movement track.

## 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°” [15]. 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 [18] from Google, than it is possible to upload videos directly to YouTube [19] and take advantage of its 360 degree control.

The camera has a very helpful developers community [20] [21], where is possible to find sample codes and explanations of how to use the camera.

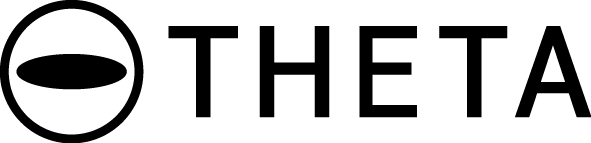


Figure 5 - RICOH Theta S Camera

## BLE (Bluetooth Low Energy)

BLE is a variation of the Bluetooth, but that consumes much less energy.

## ThingSpeak

“ThingSpeak is an Internet of Things (IoT) platform that lets you collect and store sensor data in the cloud and develop IoT applications.” [22]. 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.”.

# RESULTS AND DISCUSSION

## BLE protocol

The BLE device utilized in the project was the IMBLE BLE [23] 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:



Figure 6 - 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:



Figure 7 - 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:



Figure 8 - 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:



Figure 9 - 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.

## Arduino and BLE

The Arduino integration with the IMBLE BLE is via Serial Port. In the datasheet [23] 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

(…)

}

}

## Raspberry Pi and BLE

To use the BLE functionality from the Raspberry Pi, it is necessary to use the 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:

The reading and write handles were acquired reading the code of Android apk from the manufacturer [23].

[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: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb

handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb

handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb

handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb

handle: 0x0007, uuid: 00002a02-0000-1000-8000-00805f9b34fb

handle: 0x0008, uuid: 00002803-0000-1000-8000-00805f9b34fb

handle: 0x0009, uuid: 00002a04-0000-1000-8000-00805f9b34fb

handle: 0x000c, uuid: 00002800-0000-1000-8000-00805f9b34fb

handle: 0x000d, uuid: 00002803-0000-1000-8000-00805f9b34fb

handle: 0x000e, uuid: 00002a05-0000-1000-8000-00805f9b34fb

handle: 0x000f, uuid: 00002902-0000-1000-8000-00805f9b34fb

handle: 0x0010, uuid: 00002800-0000-1000-8000-00805f9b34fb

handle: 0x0011, uuid: 00002803-0000-1000-8000-00805f9b34fb

handle: 0x0012, uuid: 00002a29-0000-1000-8000-00805f9b34fb

handle: 0x0013, uuid: 00002803-0000-1000-8000-00805f9b34fb

handle: 0x0014, uuid: 00002a24-0000-1000-8000-00805f9b34fb

handle: 0x0015, uuid: 00002803-0000-1000-8000-00805f9b34fb

handle: 0x0016, uuid: 00002a28-0000-1000-8000-00805f9b34fb

handle: 0x0017, uuid: 00002803-0000-1000-8000-00805f9b34fb

handle: 0x0018, uuid: 00002a50-0000-1000-8000-00805f9b34fb

handle: 0x0019, uuid: 00002800-0000-1000-8000-00805f9b34fb

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

To make a Python program that can use the BLE resources, it was necessary to install the “pygattlib” library [24]. 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.

## ThingSpeak Configuration

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

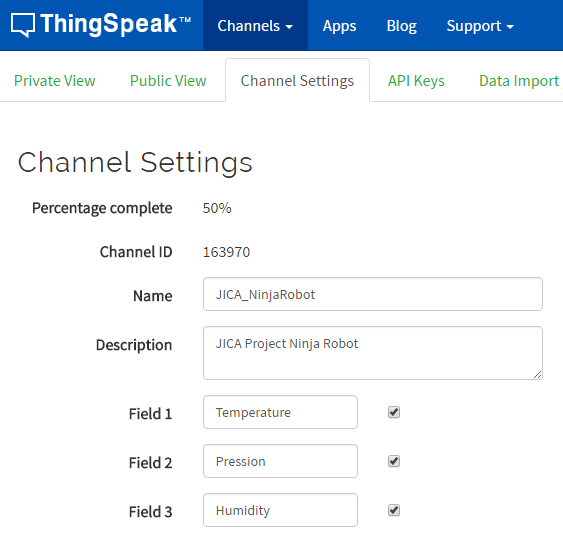
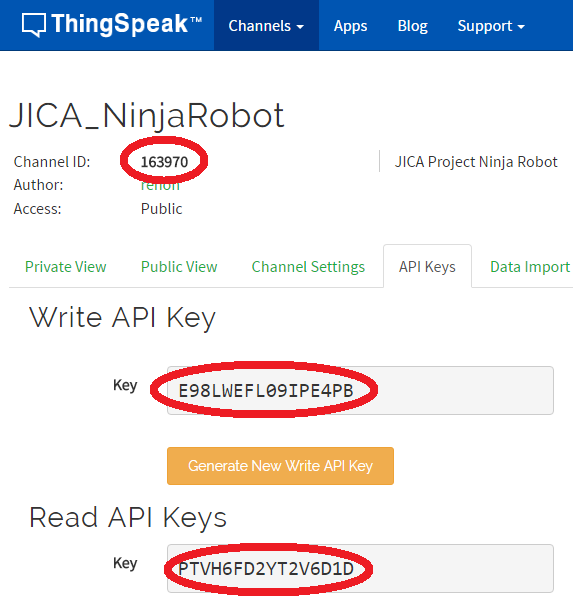
 

Figure 10 - ThingSpeak Initial Setup

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.

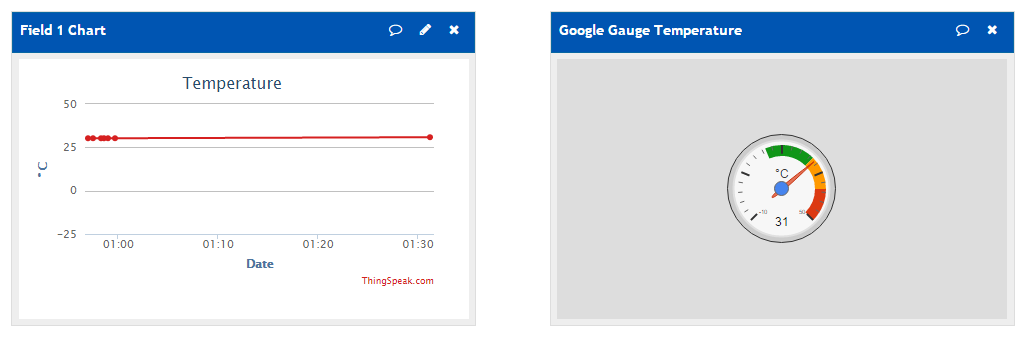


Figure 11 - ThingSpeak example of visualisations

## Raspberry Pi and ThingSpeak

## Raspberry Pi and RICOH Theta S

# PROPOSAL FOR FUTURE RESEARCH

# CONCLUSION

# REFERENCES

|  |  |
| --- | --- |
| [1] | GitHub, "JICA Ninja Robot Repository," 2016. [Online]. Available: https://github.com/KIT-Kawalab/JICA\_2016. [Accessed 2016]. |
| [2] | Arduino S.R.L., "Arduino," [Online]. Available: http://www.arduino.org/. [Accessed 2016]. |
| [3] | Arduino LLC, "Arduino," [Online]. Available: https://www.arduino.cc/. [Accessed 2016]. |
| [4] | H. Barragán, "The Untold History of Arduino," [Online]. Available: http://arduinohistory.github.io/. [Accessed 2016]. |
| [5] | E. Williams, "ARDUINO VS. ARDUINO," 25 February 2015. [Online]. Available: http://hackaday.com/2015/02/25/arduino-v-arduino/. [Accessed 2016]. |
| [6] | Processing, "Processing," [Online]. Available: https://processing.org/. [Accessed 2016]. |
| [7] | M. Margolis, in *Arduino Cookbook, 2nd Edition*, O’Reilly Media, Inc., 2011. |
| [8] | Switch Science, "BME280," [Online]. Available: https://www.switch-science.com/catalog/2236/. [Accessed 2016]. |
| [9] | Wikipedia, "I²C," [Online]. Available: https://en.wikipedia.org/wiki/I%C2%B2C. [Accessed 2016]. |
| [10] | Switch Science, "BME280 Wiki," [Online]. Available: http://trac.switch-science.com/wiki/BME280. [Accessed 2016]. |
| [11] | Raspberry Pi Foundation, " Raspberry Pi," [Online]. Available: https://www.raspberrypi.org. [Accessed 2016]. |
| [12] | Microsoft, "Windows IoT," [Online]. Available: https://developer.microsoft.com/en-us/windows/iot. [Accessed 2016]. |
| [13] | Raspberry Pi Foundation, "PiCamera," [Online]. Available: https://www.raspberrypi.org/learning/getting-started-with-picamera/. [Accessed 2016]. |