A bottom up sensor testbed

Sergio Almendros Díaz

TFG UPF / YEAR 2013

DIRECTOR/S OF THE TFG:
Jaume Barceló
DEPARTMENT:
Departament de Tecnologies de la Informació i les Comunicacions (DTIC)

Universitat Pompeu Fabra Barcelona

Dedicatòria

Acknowledgments

Acknowledgments

Abstract

A bottom up sensor testbed is a sensor platform which collect sensory data. In this thesis we will develop a sensor platform that can be attached to guifi nodes to gather and share sensory data throught the guifi network and opencities. Guifi is an open network built to everyone can join it providing his own connection and opencities is a platform developed in UPF which allows any user to upload and download sensory data.
For the guifi nodes we will use an Arduino YUN (Arduino is an open-source electronics prototyping platform) which will gathered the sensory data and send it to opencities, then an Android application will get and visualize this data.
This solution will show how to create a sensor platform and see the result very quickly which could help to other developers build their own platform to share sensory data.

Resum

Un banc de proves de sensors de baix a dalt és una plataforma de sensors que recull dades de sensors. En aquesta tesi es desenvoluparà una plataforma de sensors que es pot conectar a nodes guifi per recopilar i compartir dades de sensors a través de la xarxa guifi i opencities. Guifi és una xarxa oberta construïda per a tothom pot unir-se a ella proporcionant la seva pròpia connexió i opencities és una plataforma desenvolupada a la UPF, que permet a qualsevol usuari pujar i descarregar dades sensorials.
Per als nodes guifi utilitzarem un Arduino YUN (Arduino és una plataforma de creació de prototips electrònics de codi obert) per reunir les dades de sensors i enviar-les a openities, i a continuació, una aplicació per Android descarregarà i visualitzarà aquestes dades.
Aquesta solució mostrarà com crear una plataforma de sensors i veure el resultat molt ràpid, el que podria ajudar a altres desenvolupadors a construir la seva pròpia plataforma per compartir dades de sensors.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

The development of this project involves two parts, recollect data from sensors with the arduino and send it to opencities, there is when we will make the sensor testbed, and download the data from an Android Application to show it.

A sensor testbed is a small sensor network which has the goal to gather data, and test the technologies used as nodes to see if they are the best options to create a real one.

Bottom-up is, basically, the pattern that we used to build the sensor testbed, where the end users, in this case, guifi.net users, are the ones who have to assemble the sensor nodes and attached them to the guifi nodes to create the sensor network. With the bottom-up model, the data is provide and use by the end users, which prevents big companies or government to hide this information.

This project is an easy way to understand the importance of sensor networks and how they can help us to know, for example, if there is low quality air in our city, and do something about it.

As sensor nodes we will use an Arduino YUN, Arduino is an open-source electronics prototyping platform, that allows the user to obtain analog reads from a sensor very easily and, with a Power over Ethernet module, it can be attached to guifi nodes and send the sensory data to a sensor platform, like opencities.

When the sensory data is stored, we will develop an Android application to visualize this data and make it more accessible to other users not involved with guifi.net.

In the following chapters I will explain the state of sensor networks nowadays 2, which technologies we will use 3, and how the project has been done, as well as all the problems found during the process 4.

The final goal of the project is to build a sensor testbed and there will be the results 5, and, to finish, the conclusions 6 and the future work 7.

Chapter 2

STATE OF THE ART

2.1 Introduction

Sensor networks started as a mecanism of defense develop by the military during the Cold War, with acoustic sensors they try to find Soviet submarines. But, this rearch continue at universities, trying to make this sensors smaller, and with the posibility of real-time data[Chong and Kumar, 2003].

Right now, the sensors are small enough, and the processors with network technology consume low energy, which allows us to deploy a test bed without bothering the people around it.

Smart cities are the next step, a city capable of having real-time information, not only about the environment, it can go from the amount of cars that pass a road, to the amount of rain water in a day. This kind of information help to manage more efficiently the city.

It is important to share this information, in the case that the government build the sensor network, the data should be open to everyone can see it. There are already some sensor networks functioning, some of them are from the government, and, sometimes, there are not that open about their data, but there are also some people who have sensor in his home and share their sensory data with anyone who wants to see it.

2.2 Sensor networks and smart cities

In this section we introduce a few projects of sensor networks that cities deployed:

2.2.1 Amsterdam smart city

Amsterdam have a lot of projects concerning the smart city concept, like the ”Flexible street lighting”, which allows the government to monitor the street and switch off the lights saving energy, or the ”Smart parking” which let drivers to know if there are free spots to park, and, in consecuence, reduce air pollution[city in Amsterdam, 2013].

2.2.2 Santander smart city

Santander has his own sensor network testbed for environmental monitoring, outdoor parking area management, or traffic intensity monitoring[city in Santander, 2013].

2.3 Companies

There are some companies that are in the business of sensor networks, such as ”Schneider Electric”, a multinational company that produces components for energy management, or smartcitizen, a platform that allows a user to have a sensor node and share the data with everyone.

2.4 Opendata services

The sensor networks are useless if we don’t store the data, although we could save it in the device, it would be too expensive to recollect it, so we chose a web opendata service, which is a website that allows the user to upload and download the data with an open API, and normally they have some way to visualize it.

There are some services such as Xively, o sentilo which allows you to install it in your server, and opencities, an opendata service developed in the Pompeu Fabra University.

Chapter 3

TECHNOLOGIES

In this chapter we focused in the techonologies used to develop this project, which is divided in four parts: arduino, sensors, python, Guifi network and opencities, and Android.

3.1 Arduino

The Arduino board that we use is an arduino YUN[1] as we can see in the figure 3.1, which support a Linux distribution based on OpenWRT named Linino, and it has Ethernet and wifi suport, and a micro-SD card slot, those are basically the reason why we decided to use it, as we have to store the recollected data, and send it to opencities.

We also have planned to attached a power over ethernet (PoE) module because the arduino's will be attached to guifi nodes which are also PoE.

3.2 Sensors

A sensor is a device which transform a physical measure to an output signal that can be read by another device, such as an arduino,

In this project we will use five sensors that measured temperature, light, noise, humidity, and gas.

To show how the sensors are connected to the arduino YUN I used the program fritzing[2].

[1] `http://arduino.cc/en/Main/ArduinoBoardYun?from=Main.ArduinoYUN`

[2] `http://fritzing.org/`

Figure 3.1: Arduino YUN.

3.2.1 LM35: Temperature

LM35 is a sensor with an output voltage proportional to the Centigrade temperature, the output pin goes directly to an analog pin in the arduino 3.2 as shown in the figure 3.3.

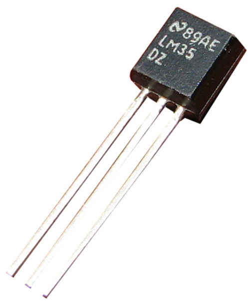

Figure 3.2: LM35 temperature sensor.

3.2.2 Light Dependent Resistor (LDR)

The LDR3.4 is a light sensor that where is connected to an arduino, as shown in the figure 3.5, returns a value between 0 to 1024 depending on the light.

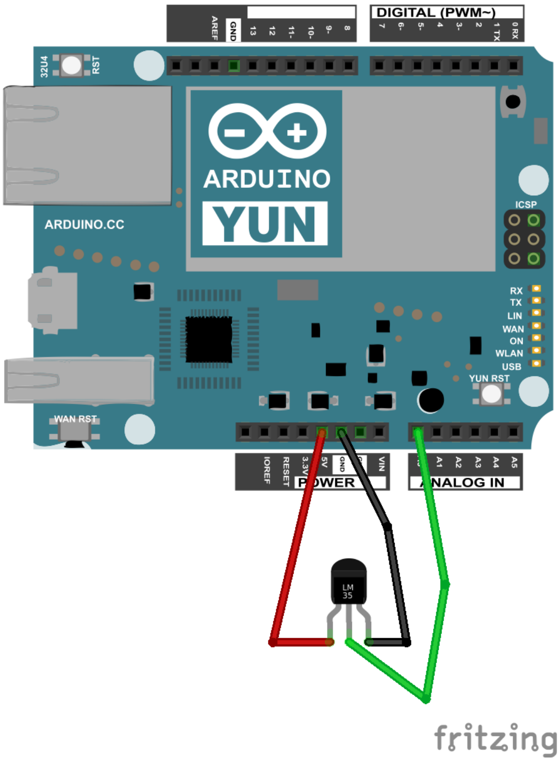

Figure 3.3: Temperature Sensor Breadboard.

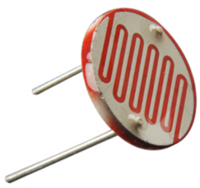

Figure 3.4: photoresistor or light-dependent resistor.

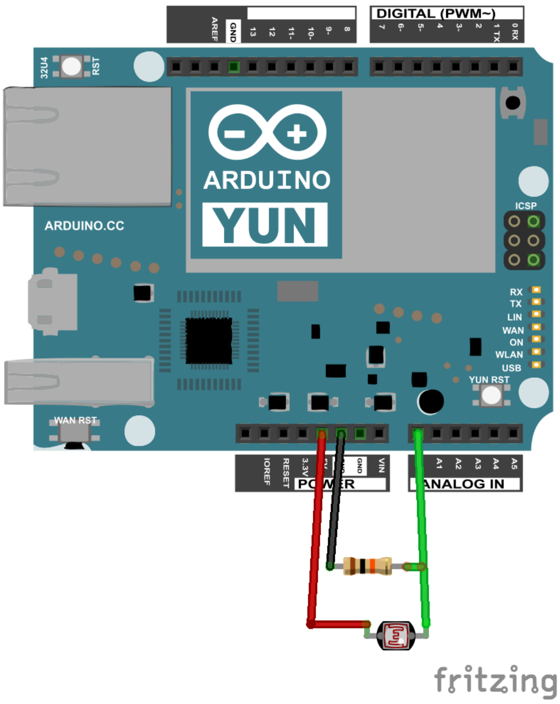

Figure 3.5: Light Sensor Breadboard.

3.2.3 Emartee Mini Sound Sensor and Analog Sound Sensor Board Microphone MIC Controller: Noise

This two sensors are used to measured noise levels 3.6 and 3.7, which is similar to the LM35 in the connexion, the output pin goes directly to an analgo pin as shown in the figure 3.8, but I did not find a figure for this sensor, so I put a microphone that has the same pins.

Figure 3.6: Mini Sound Sensor.

3.2.4 Aosong DHT22 and DHT11: Humidity

DHT223.9 is a humidity and temperature sensor, although we will only use the humidity measure. The output is digital, and to read it, we use an external library[3]. The arduino and the humidity sensor will be connected as shown in the figure 3.10.

At the time I started the testbed, there was only 1 DHT22 sensor, so I had to use two DHT11 sensor, which change the breadboard a little bit as shown in the figure **??**.

3.2.5 MQ135: Gas sensor

This sensor is gas sensor, and we will use it to measured air quality. In this webpage[4] explain how to connect the sensor to the arduino, but the one that I bought

[3]`https://github.com/adafruit/DHT-sensor-library`
[4]`http://arduino-info.wikispaces.com/Air-Gas-Sensors`

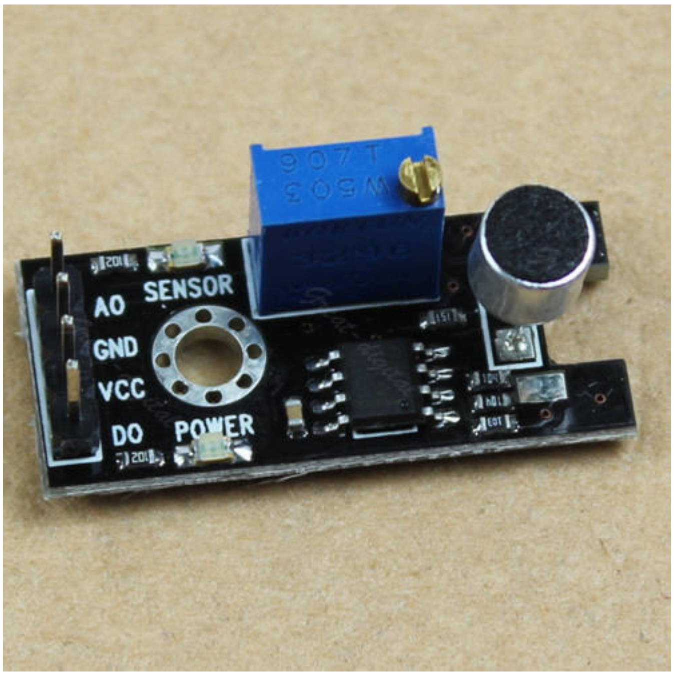

Figure 3.7: Analog noise sensor.

has three pins joined in one, which make it easier 3.12, but this has no figure in fritzing, so I used a gas sensor that has the same output, and the resutl is in the figure 3.13.

The previous sensors are more similar, but this one can be used to measure the environment by the citizens and, it can be used to control the gas in a factory for example.

3.2.6 BreadBoard with all the sensors

This 3.14 is the final prototype of the arduino YUN and all the sensors connected to it.

3.3 Python

Because of the low memory for the arduino sketches, we have to use a python script to communicate with opencities. The version of python in the arduino is the 4.2.5.

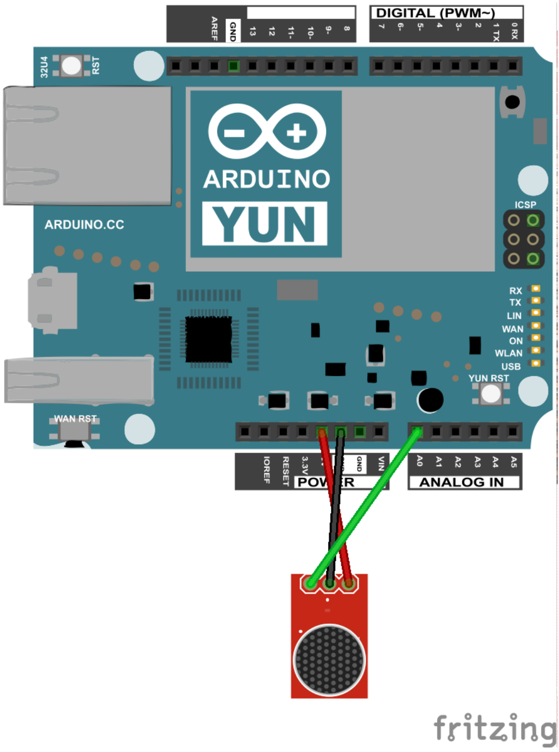

Figure 3.8: Noise Sensor Breadboard.

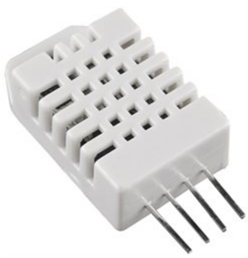

Figure 3.9: DHT22 humidity and temperature sensor.

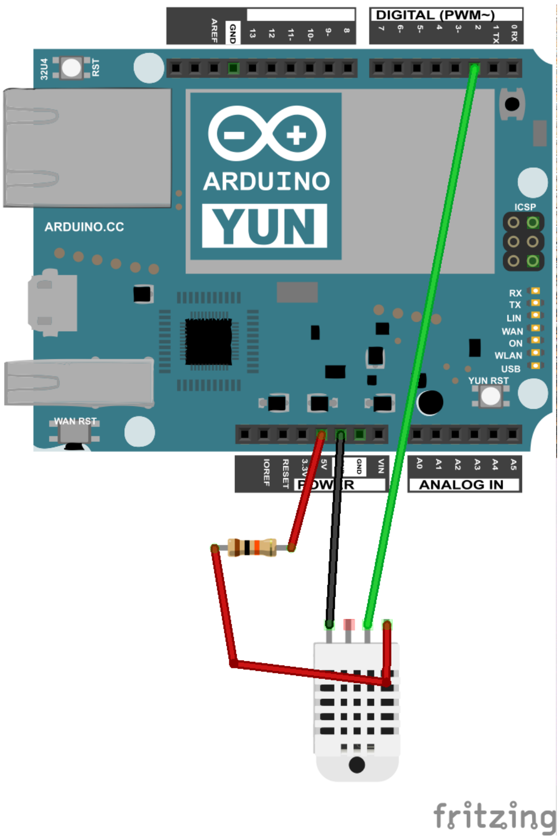

Figure 3.10: Humidity Sensor DHT22 Breadboard.

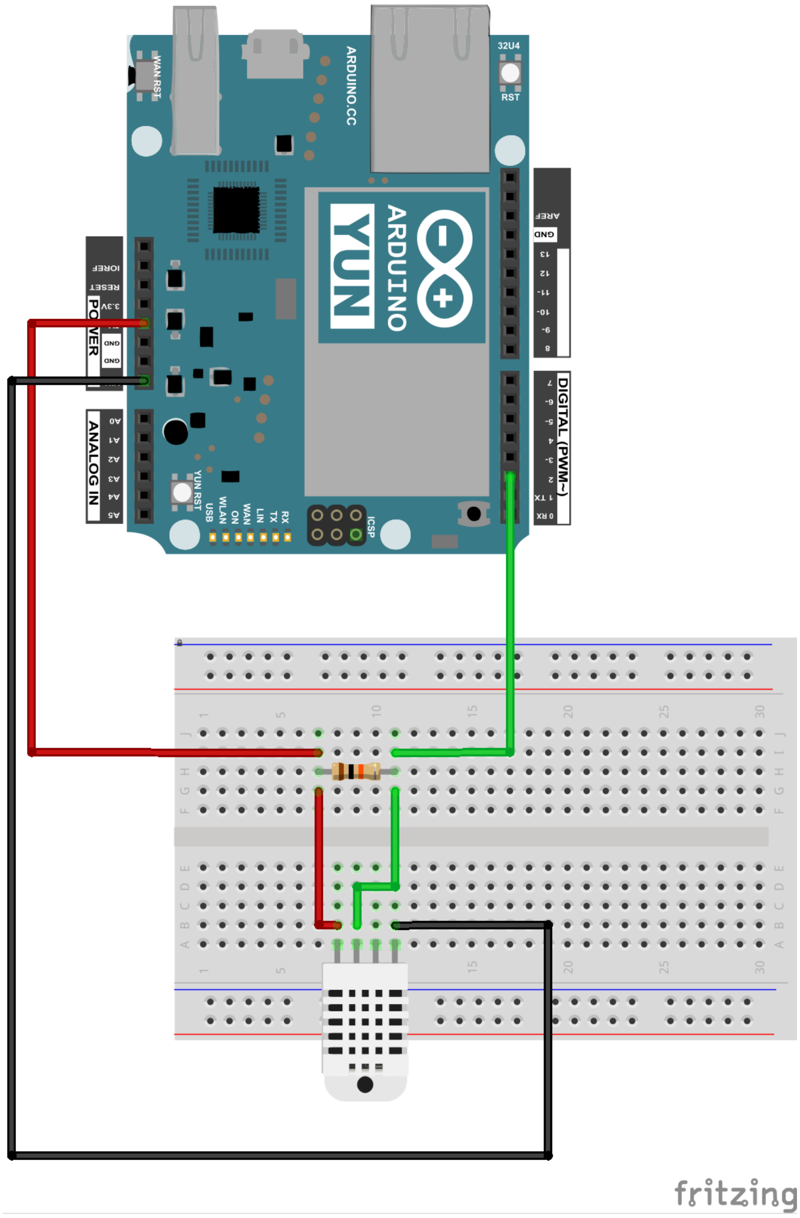

Figure 3.11: Humidity Sensor DHT11 Breadboard.

Figure 3.12: MQ135 Air Quality sensor.

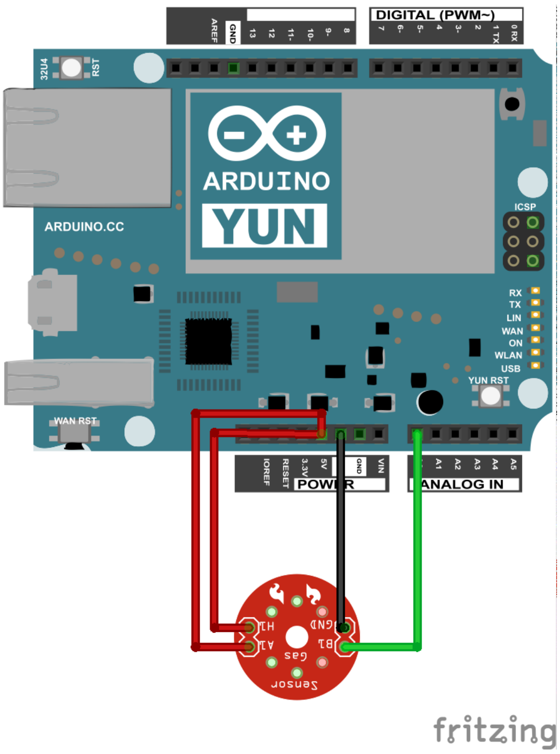

Figure 3.13: Gas Sensor Breadboard.

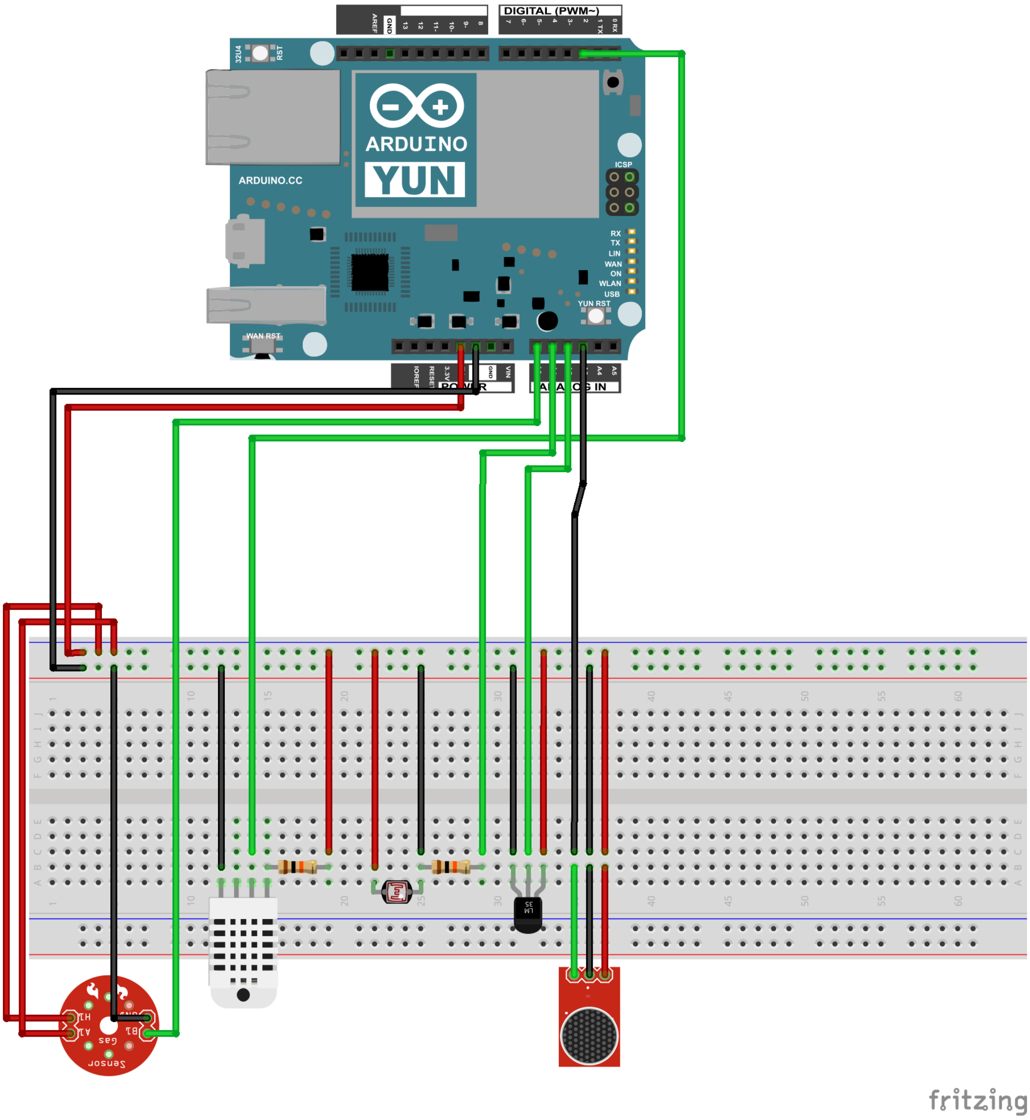

Figure 3.14: Breadboard of all the sensors connected to the Arduino YUN.

3.4 Guifi network and opencities

Guifi network[5] is the network where the arduino's will be installed, and the one which will provide the access opencities through the Internet.

Opencities[6] is the opendata services that we have chosen, the strengths of opencities are that give us free storage, a great and easy API to upload and download the data, and, also, the developers are in the UPF and problems can be solved more easily.

3.5 Android

Android is an open source mobile operating system from Google, it runs on smartphones, and we will use this OS to develop an application to see the sensory data stored in opencities, and show it to the user in a way that anybody can understand the values.

This application it will be tested on a Sony Xperia Z1, with an Android 4.3.

[5] `https://www.guifi.net/`

[6] `http://opencities.upf.edu/web/index.php/en/`

Chapter 4

BOTTOM UP SENSOR TESTBED

This chapter focused in the process that it is been followed to complet the project, which has two main parts, the software to recollect and send the sensory data and the Android application to show it.

4.1 Arduino Code

The arduino YUN will run only one file, CollectAndSendData.ino, but the arduino sketches are executed with a very low memory, so we need to use a python script which will run in the linino.

To get started with how the arduino YUN visit this website[1].

The arduino sketch is responsible for collecting the data, write it down in the logData file, and call the python script with the collected values and an unique ID, and the python script create the GeoJSON and send it to opencities.

4.1.1 Collect sensory data

To collect almost all the data the skecth does not need to include any libray because it is read by the analog read, except for the humidity sensor (DHT22) which need an external library.

To read and write into the logData file the FileIO library[2] is needed, and to call the python script we need the Process library[3].

The figure 4.1 explain how the arduino sketch works, but because is a process a little bit long, a more specific explanation will be in the appendix:

[1]`http://arduino.cc/en/Guide/ArduinoYun`
[2]`http://arduino.cc/en/Reference/YunFileIOConstructor`
[3]`http://arduino.cc/en/Reference/YunProcessConstructor`

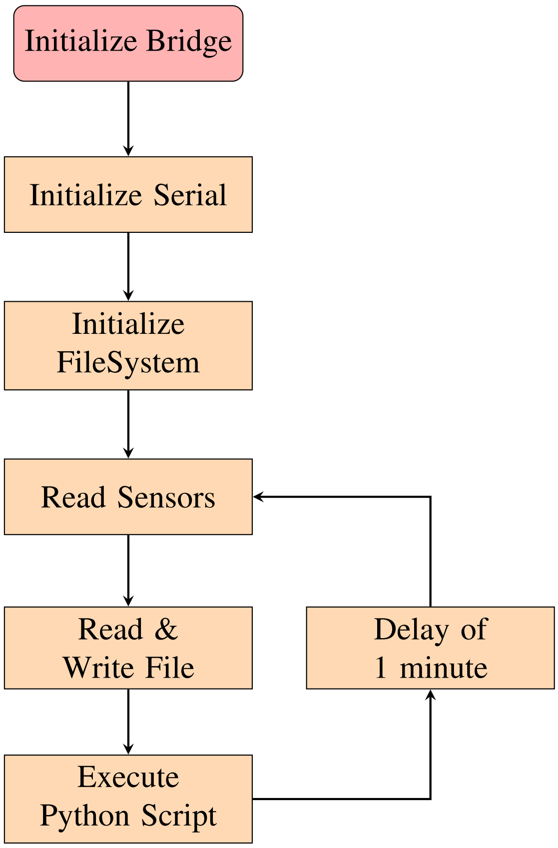

Figure 4.1: Arduino sketch Flow Chart.

4.1.2 Communication with opencities

The communication with opencities is done by a python script, to do this a set of libraries are needed and there are some which have to be installed in the linino.

We need the sys and datetime library that are already installed, but we need the geopy library to get the latitud and longitude that will be include in the geojson, to do that we need the geojson library. To post the sensory data into opencities we use the httplib2.

Now all the steps to install the libraries mentioned above are explained:

1. First we need to configure the onboard wifi, in this website it is explained how[4]

2. When the YUN have an IP, now we can get into the linino by Secure Shell:
   ssh root@X.Y.Z.W

3. Now that we are in the linino, we begin to install the necesary packets:
   opkg update
   opkg install distribute
   opkg install python-openssl
   easy_install pip
   pip install geojson
   pip install geopy
   pip install httplib2

With all this libraries we can communicate with opencities and store the sensory data recollected by the arduino. The 4.2 figure explain how the arduino sketch works, but because is a process a little bit long, a more specific explanation will be in the appendix:

4.2 Android app

4.2.1 Summary

To make easy to see the results of the testbed, I created an android application for the purpose of visualize them. The application shows the data of the sensors in two ways, with markers which will show the actual value of the temperature or noise in that point, and, also, with heatmap points, the larger the value of the temperature or noise, the more intense the red will be. This can be seen in figure 4.3.

[4] http://arduino.cc/en/Guide/ArduinoYun

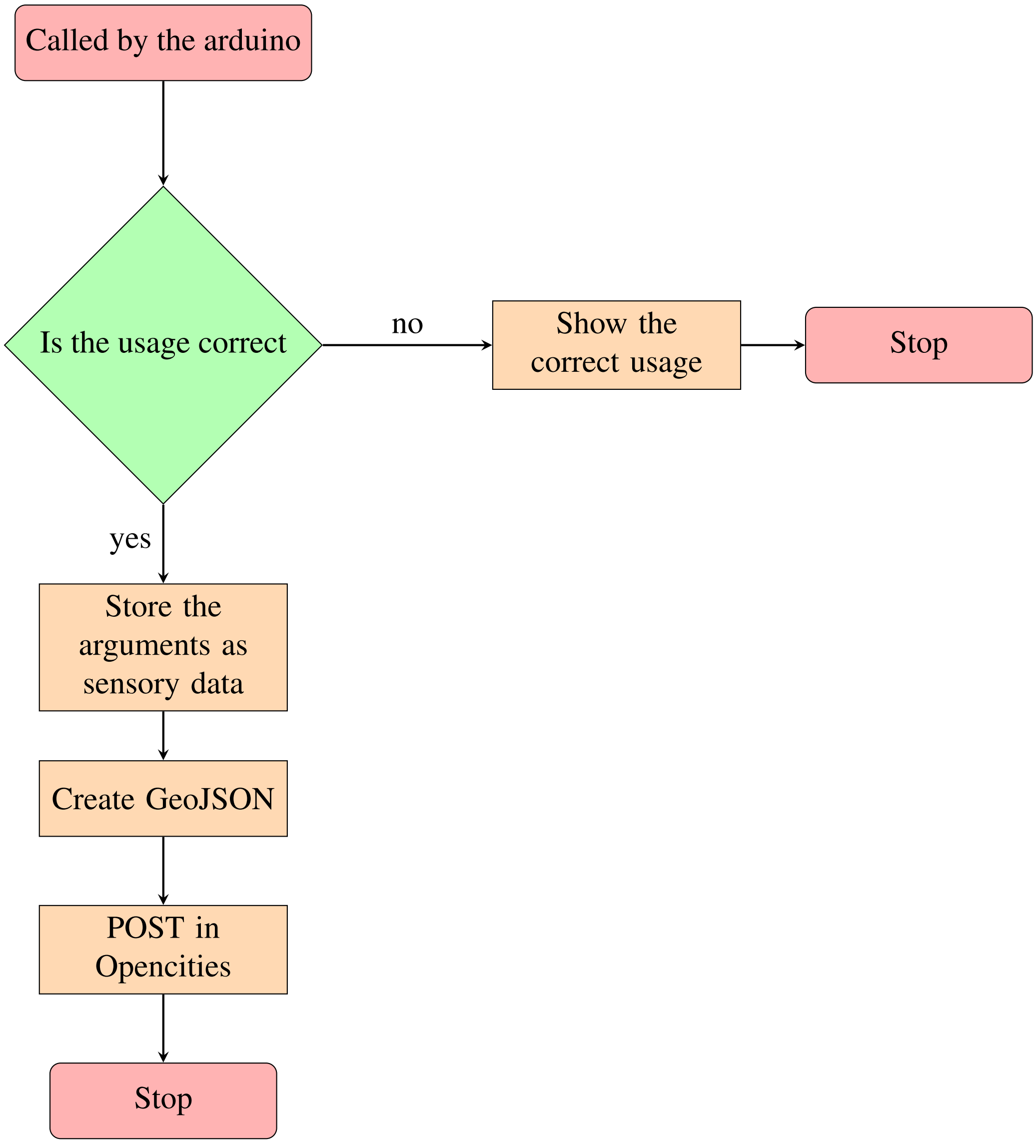

Figure 4.2: Python Script Flow Chart.

Figure 4.3: App Screenshot.

4.2.2 Interface

The application interface is a unique map view, where the user can zoom to a limit, go to their location, and, use the top buttons. From left to right, the first button is the Marker button, the user decides whether the markers are displayed or not, and the next buttons refer to the type of data sensor which is shown as markers and as heatmap points (Temperature, Humidity, Noise, Light, and Air Quality). If the Marker button is checked, the user can click on the marker in the map and it will show the value of the temperature, humidity,... and the unit. In the figure 4.4 we can see the app with the Marker button checked, and in the figure 4.5 without.

4.2.3 Code

First of all, to create this application I have used the Google Maps Android API v2[5] for the map view, and the Google Maps Android API utility library[6] for the heatmaps.

This application has the next classes, in the figure 4.6 and fig:ClassDiagram2 are the class diagram:

- MainActivity: Is the controller of the whole application.
- GPSTracker: Is a class to get the current location of the user.

[5] `https://developers.google.com/maps/documentation/android/`
[6] `http://googlemaps.github.io/android-maps-utils/`

Figure 4.4: App Screenshot.

Figure 4.5: App Screenshot.

- Feature, Geometry, and Properties: This are the classes where it will stored the data from the parsed JSON from opendata.

- DataBase: This is a singleton class where all the variables are stored, because is more easy to access from different clases.

The explanation of the code will be in the following flow chart 4.8:

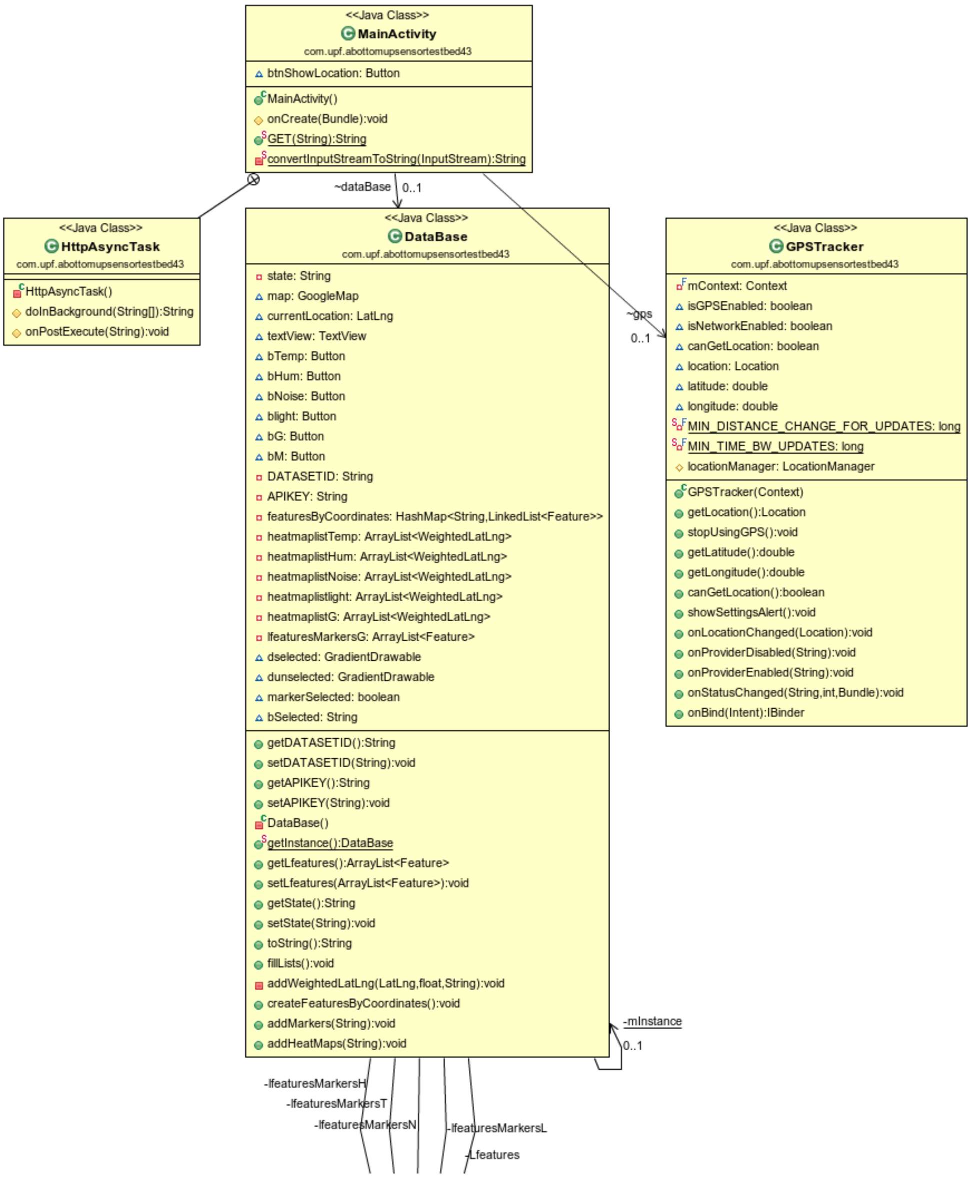

Figure 4.6: Class Diagram of the Android App part 1.

Figure 4.7: Class Diagram of the Android App part 2.

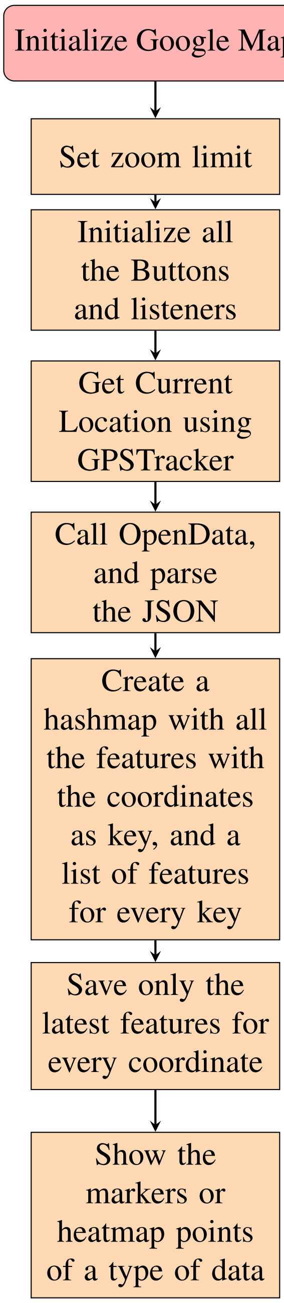

Figure 4.8: Android App Flow Chart.

Chapter 5

TESTBED RESULTS

In this chapter I will explained the procedure I followed to make this testbed, and an explanation of the results.

In the figure 5.1 there is a photograph of the prototype I will use in this testbed. It is composed of an Arduino YUN, a microSD card, a breadboard, and all the sensor connected (temperature, humidity, noise, light, and gas).

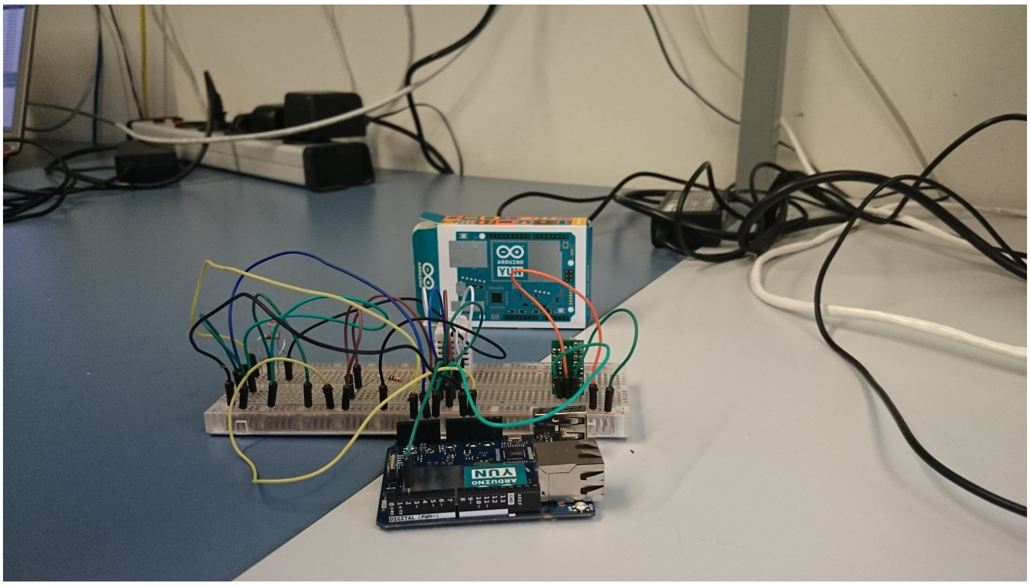

Figure 5.1: TestBed Prototype.

5.1 Sensor node

Now, I will show the process to configure the node, part of this process is taken of the website of arduino[1]:

5.1.1 Connection to the Internet

First of all we have to provide of Internet connection to the arduino.

5.1.1.1 Through Ethernet

This is the fastest way to provide of Internet connection, the arduino will behave the same way as a computer, automatically will have an IP address.

5.1.1.2 Through WiFi

This is a slowest way. The process is the following:

1. First we power the arduino YUN.
2. The arduino will create his own WiFi network (ArduinoYun-XXXXXXXXXXXX), and with a computer we connect to it.
3. When we are connected to the YUN network, we go to a web browser and go to `http://arduino.local` or `192.168.240.1`. We have to put the password, which is "arduino". As we can see in the figure 5.2.
4. The next page will be an information page, click on the configure button 5.3.
5. We will give a unique name to the Yun, and the network we want to connect 5.4.
6. We press the configure & restart button.
7. Finally we have to connect to same network as the Yun is connected.

[1] `http://arduino.cc/en/Guide/ArduinoYun`

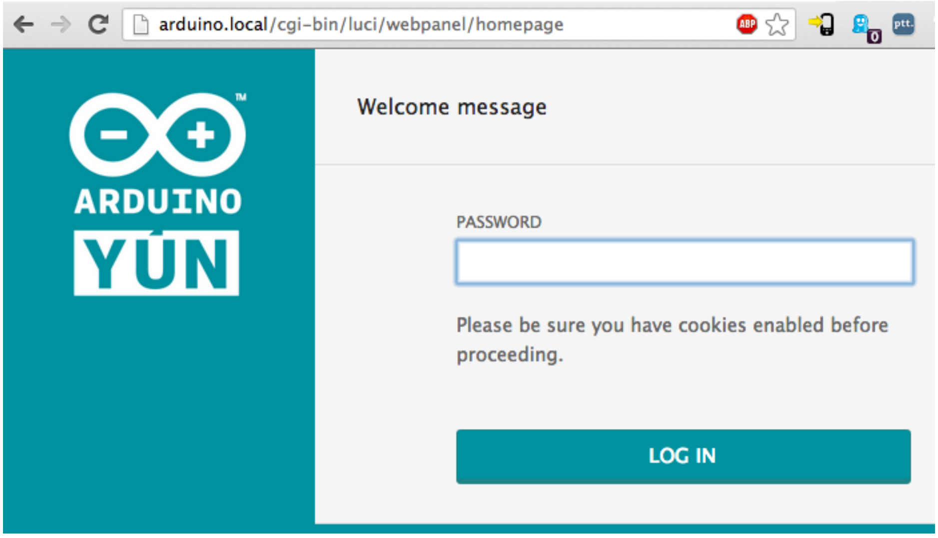

Figure 5.2: Yun web Password.

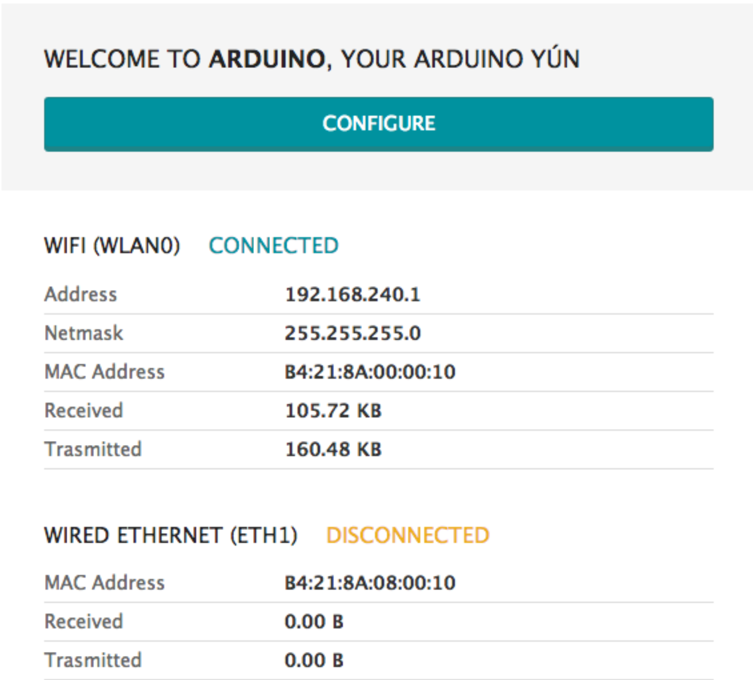

Figure 5.3: Yun web Diagnostic.

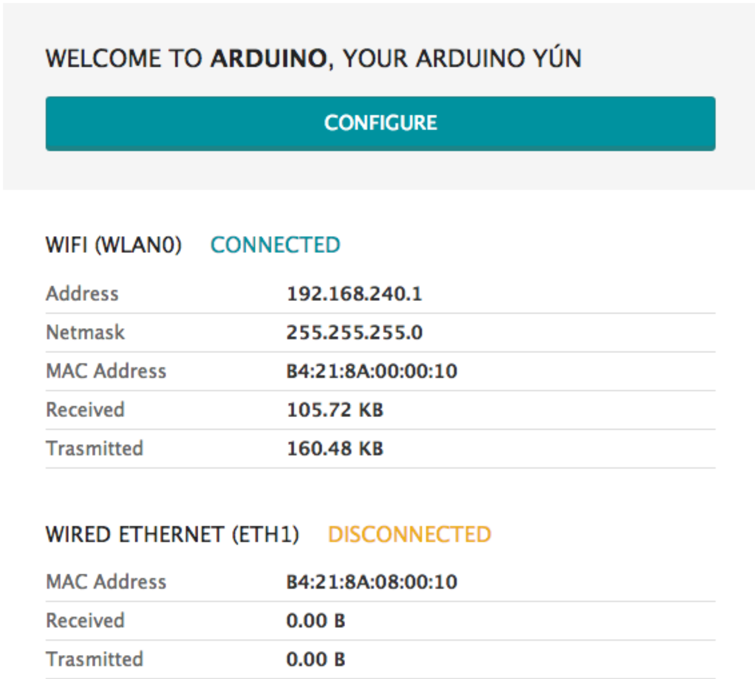

Figure 5.4: Yun web Configuration.

5.1.2 Install necessary packets

The arduino will run an arduino script and a python script, for the first one we do not have to install anything, but for python we will have to install some packets.

To install any packet we first have to connect to the arduino (The password will be the one we entered when we configured the onboard WiFi or ”arduino”):

ssh root@X.Y.Z.W

Now we can install all the necessary packets:

opkg update
opkg install distribute
opkg install python-openssl
easy_install pip
pip install geojson
pip install geopy
pip install httplib2

5.1.3 Copy the scripts

First of all we have to create the some directories, so once we make the ssh command, go to ”/mnt/sda1”, make the following comands:

mkdir arduino
cd arduino
mkdir www

We have to copy the python script ”main.py” into the SD-Card. We have two ways to do this, put the SDCard into a computer an save the files in there, or we can copy the files into the arduino through the network with the following comand:

scp main.py root@192.168.2.149:/mnt/sda1/arduino/www/main.py

5.1.4 Attach the sensors

Now that we have the python step done, we have to attach the sensors to the arduino Yun, to do that look at figure 3.14.

5.1.5 Arduino Code

To upload an arduino sketch to the yun we have to use the following IDE: arduino 1.5.5. There are two ways to upload an sketch, through a USB cable connected to the arduino, or through the Internet, if we are in the same network as the arduino, it will appear in the IDE.

5.2 Actual Testbed

We have three arduino, so we have to mount the sensor node and put it in three different places in free space but being sure it will not get wet.

When we have decided when we are going to put them, we have to introduce manually the location and the unique ID into the python script. We can do that by entering into the arduino by secure shell as we did earlier, go to the folder where the ”main.py” is, and modified the following line by using ”nano”:

self.address = ’Sagrada Familia, Carrer de Mallorca, Barcelona’

After a day of collecting data, we will get the arduino’s back, take the three logData files, and show the data into some graphics.

The figures **??** show the testbed I did.

Chapter 6

CONCLUSIONS

Chapter 7

FUTURE WORK

Chapter 8

APPENDIXES

8.1 Pilot Charter

Fellow: Sergio Almendros Diaz
Mentor:
Advisor: Jaume Barcelo

8.1.1 Pilot purpose or justification

The purpose of this pilot is to build a sensor platform that can be attached to guifi nodes to gather and share sensory data.

8.1.2 Measurable pilot objectives and related success criteria

- Gather data about temperature, humidity, light, and noise.
- Share the data as open data.
- Deploy at least two nodes and gather data for at least two weeks.

8.1.3 High-level requirements

- Outdoor enclosure.
- Use open hardware and open software to the possible extent.
- Use standardized interfaces to integrate with other projects.

8.1.4 High-level pilot description

The goal is to use an arduino platform to create a bottom-up broadband wireless sensor networks. As guifi.net has already over 20,000 nodes, the idea is to co-locate the sensory platforms together with the guifi.net nodes and use the guifi.net network to transmit the data. This data should be gathered and shared. Ideally, the pilot should include a presentation interface for the users to visualize the data.

8.1.5 High-level risks

A possible risk is that the prototypes are not rugged enough for outdoor environments. It is also a risk that the prototype is not stable and needs to be reset very often.

8.1.6 Summary milestone schedule

- From 20/09/2013 to 23/09/2013
  - Establish the general idea of the TFG and specifics goals.
- From 23/09/2013 to 11/10/2013
  - Specify the tasks to do and make a planning.
- From 11/10/2013 to 30/10/2013
  - Connect first sensors to the Arduino.
- From 31/10/2013 to 10/01/2014
  - Connect to guifi network and upload data to an open data platform.
- From 10/01/2014 to 01/06/2014
  - Integration of sensors and communication aspects.
  - Install prototypes.
  - Data sharing and visualization.
  - Data analysis and evaluation of the testbed.
- From 02/06/2014 to 30/06/20014
  - Preparation of the final memory.
- From 01/07/2014 to the date of the presentation
  - Make the presentation.

8.1.7 Summary budget

The cost of this pilot will be approximately 4000 €. This quantity is for the scholarship to the student that will develop this pilot, budget for attending a conference or visiting collaborators, and the purchase of the necessary hardware.

8.2 Planning Report

The following sections explain the tasks that I will do in the course of this project.

8.2.1 Familiarization with the Arduino Yun

In this project I will be working with an arduino Yun, but I never worked before with any type of arduino, so the first task is to start coding different kind of programs. Then I will have to learn how to interact with the linux in the arduino Yun.

8.2.2 Preliminary testbed

I want to do an easy example to how to connect an arduino with a server running in my computer, what I want to do is establish a bridge between an arduino program and the linux within the arduino to be able to communicate with a server in my laptop, and send a string with the value returned by a sensor. This is a reduce problem of the real ”bottom-up sensor testbed” because, at the end, in every arduino will be a program that will have to send a message to a server with the data of the sensors attached to it.

8.2.3 Collect Data from sensors

First I will connect a temperature sensor to the arduino YUN, then, I will develop a program to collect the information from it, and send it to a server. When the temperature sensor works, I will do the same process with a humidity, light, and noise sensor.

8.2.4 Install Sentilo

Sentilo (www.sentilo.io) is an open source sensor and actuator platform that I will install in my laptop to act as the server between the sensor network and the interface for the users to visualize the data.

8.2.5 Communication with Sentilo

I will adapt the messages that the arduino send to fit with the Sentilo.

8.2.6 Real deployment

At this moment, the part of the arduino and the server will be done, so I will test the server installing the arduino in real nodes of the guifi network, for example, the node in the Universitat Pompeu Fabra, and any other node that allow me to install it. The arduino will have a temperature, humidity, light, and noise sensor.

8.2.7 Interface

I want to do an interface for any user to understand the meaning of the temperature, humidity, light, and noise values. This interface will be develop for an android mobile application.

8.2.8 Sentilo module

I will contribute to Sentilo and other sensor data brokerage platforms accommodating the sensor testbed deployed in the previous tasks.

8.2.9 Final report

This task have to be done in parallel with all the other ones, and its purpose is document all the work that I will do.

8.2.10 Gantt chart

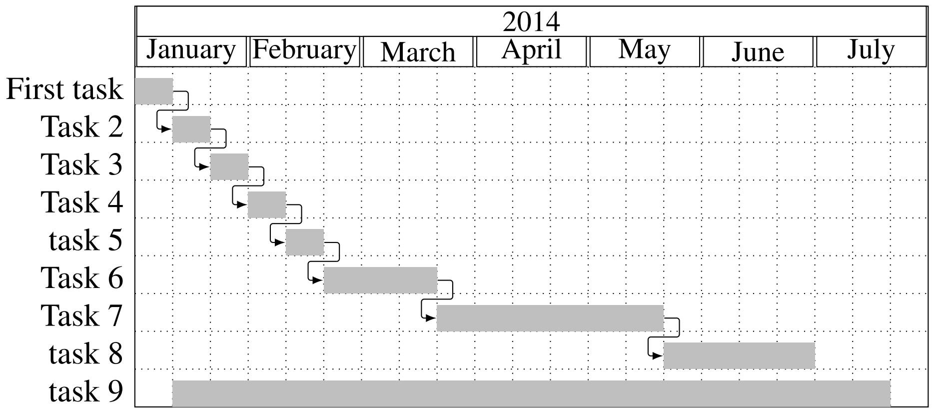

Figure 8.1: Gantt Chart

Bibliography

[Chong and Kumar, 2003] Chong, C.-Y. and Kumar, S. P. (2003). Sensor networks: evolution, opportunities, and challenges. *Proceedings of the IEEE*, 91(8):1247–1256.

[city in Amsterdam, 2013] city in Amsterdam, S. (2013). Smart city in amsterdam. [Online; accessed 07/02/14].

[city in Santander, 2013] city in Santander, S. (2013). Smart city in santander. [Online; accessed 10/02/14].