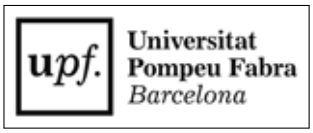
A bottom up sensor testbed

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)



Dedicatòria

Acknowledgments

Acknowledgments

Abstract

Bottom up defines the network design, deployment, and operation initiatives driven by end user needs[1]. Sensor testbed is a deployment of several sensors over a defined location.

In this project we will develop a bottom up sensor testbed, which means a deployment of a sensor testbed driven by end user needs. The sensor nodes will consist of an Arduino[2] YUN (Arduino is an open-source electronics prototyping platform) and 5 environmental sensors attach to it.

The arduino YUN is connected to Guifi[3], an open network built to everyone can join it providing his own connection. The Guifi nodes are Power Over Ethernet (PoE), and the Arduino YUN can be connected to it with a PoE module, but there is no module yet.

The arduino send the sensory data to opencities[4], a platform to browse, visualize, and download open data from different participants.

Finally, an Android aplication shows the data from opencities.

At the end of this thesis we made a real testbed with 3 nodes, it shows that the data is send correctly and without any interruption.

Resum

Cuando el Abstract esté perfecto, lo traduciré al catalan.

[1] http://bubforeurope.net
[2] http://arduino.cc/
[3] http://guifi.net/
[4] http://opendata.nets.upf.edu/web/index.php/en/

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

The development of this project involves two parts, recollect sensory data and show it.

A sensor testbed is a sensor network that has the goal to gather sensory data, and test the technologies used as nodes to see if they are the best option.

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 their guifi nodes to create the sensor network. With the bottom-up model, the data is provide and use by the end users. 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, which 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 (Chapter 2), which technologies we will use (Chapter 3), and how the project has been done (Chapter 4).

Finally there will be the results (Chapter 5) of the testbed, and conclusions

(Chapter 6) and future work (Chapter 7).

Chapter 2

STATE OF THE ART

2.1 Introduction

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

Right now, sensors are small enough, and processors with network technology have low energy consumption, which allows us to deploy a test bed without people notice 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 could 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 could 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 sensors nodes at home and share the information with everyone.

2.2 Sensor networks and smart cities

In this section we introduce a few projects of sensor networks 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 to save 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. That is why we have chosen Opencities[1], a platform to browse, visualize, and download open data from different participants. We will only use this platform to store and download the data in the Android App.

There are some similar services such as Xively, or sentilo, but opencities is develop in the Pompeu Fabra University, what means that if I have any problem, a solution will be found more quickly.

[1] `http://opendata.nets.upf.edu/web/index.php/en/`

Chapter 3

TECHNOLOGIES

In this chapter we focused in the techonologies used to develop this project, which are 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, it supports a Linux distribution based on OpenWRT named Linino, 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 through the Internet.

We have also planned to attached a power over ethernet (PoE) module to the arduino to make more clean the deployment.

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.

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

Figure 3.1: Arduino YUN.

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

3.2.1 LM35: Temperature

LM35 [Figure 3.2] is a sensor to mesure temperature, in the figure 3.3 we can see the way to connect it to the arduino.

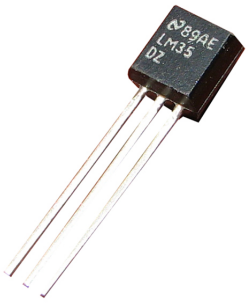


Figure 3.2: LM35 temperature sensor.

[2] http://fritzing.org/

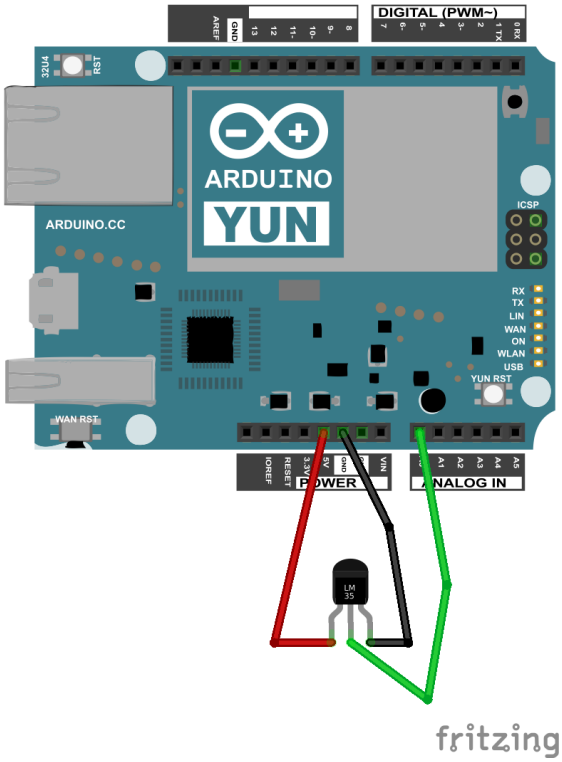


Figure 3.3: Temperature Sensor Breadboard.

3.2.2 Light Dependent Resistor (LDR)

LDR [Figure 3.4] is a light sensor, in the figure 3.5 we can see the way to connect it to the arduino.

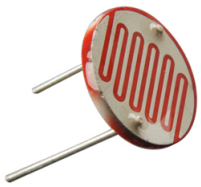


Figure 3.4: photoresistor or light-dependent resistor.

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 [Figure 3.6] and [Figure 3.7]. The code to read the noise values is the same for both. In the figure 3.8 we can see the way to connect them to the arduino.

3.2.4 Aosong DHT22 and DHT11: Humidity

DHT22 [Figure 3.9] and DHT11 [Figure 3.10] are humidity and temperature sensors, 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 3.11.

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 3.12.

[3] https://github.com/adafruit/DHT-sensor-library

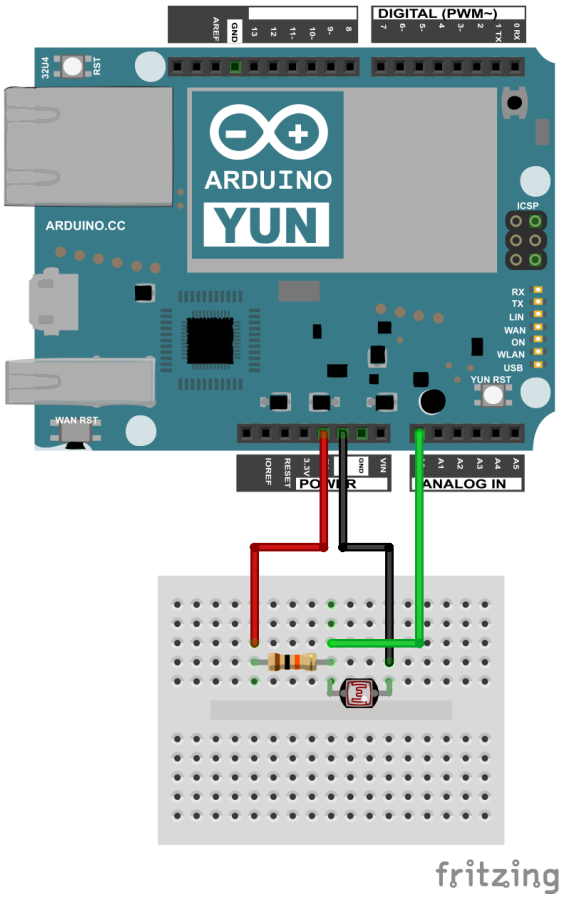


Figure 3.5: Light Sensor Breadboard.

Figure 3.6: Mini Sound Sensor.

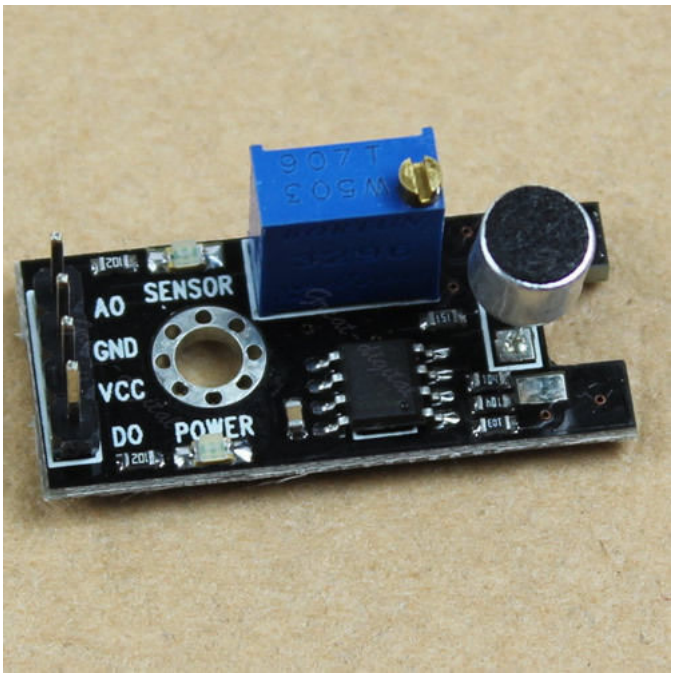


Figure 3.7: Analog noise sensor.

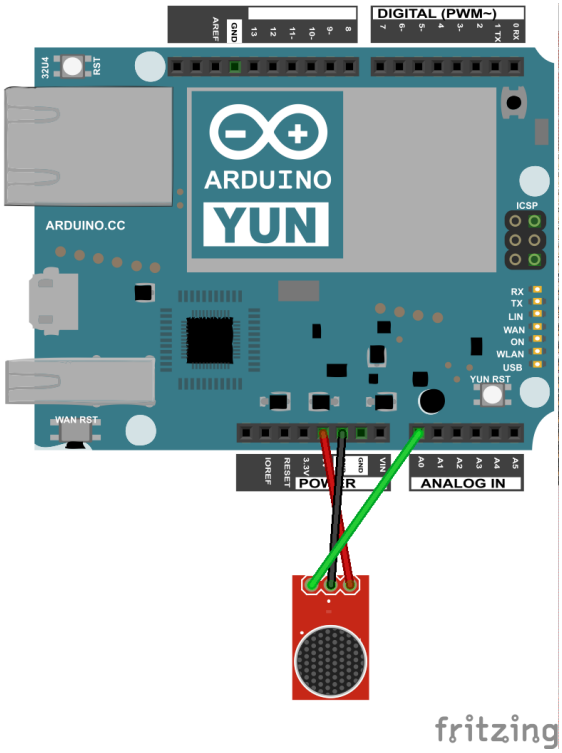


Figure 3.8: Noise Sensor Breadboard.

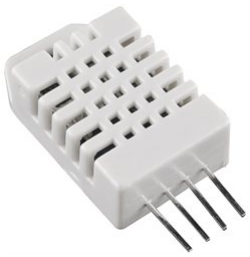


Figure 3.9: DHT22 humidity and temperature sensor.

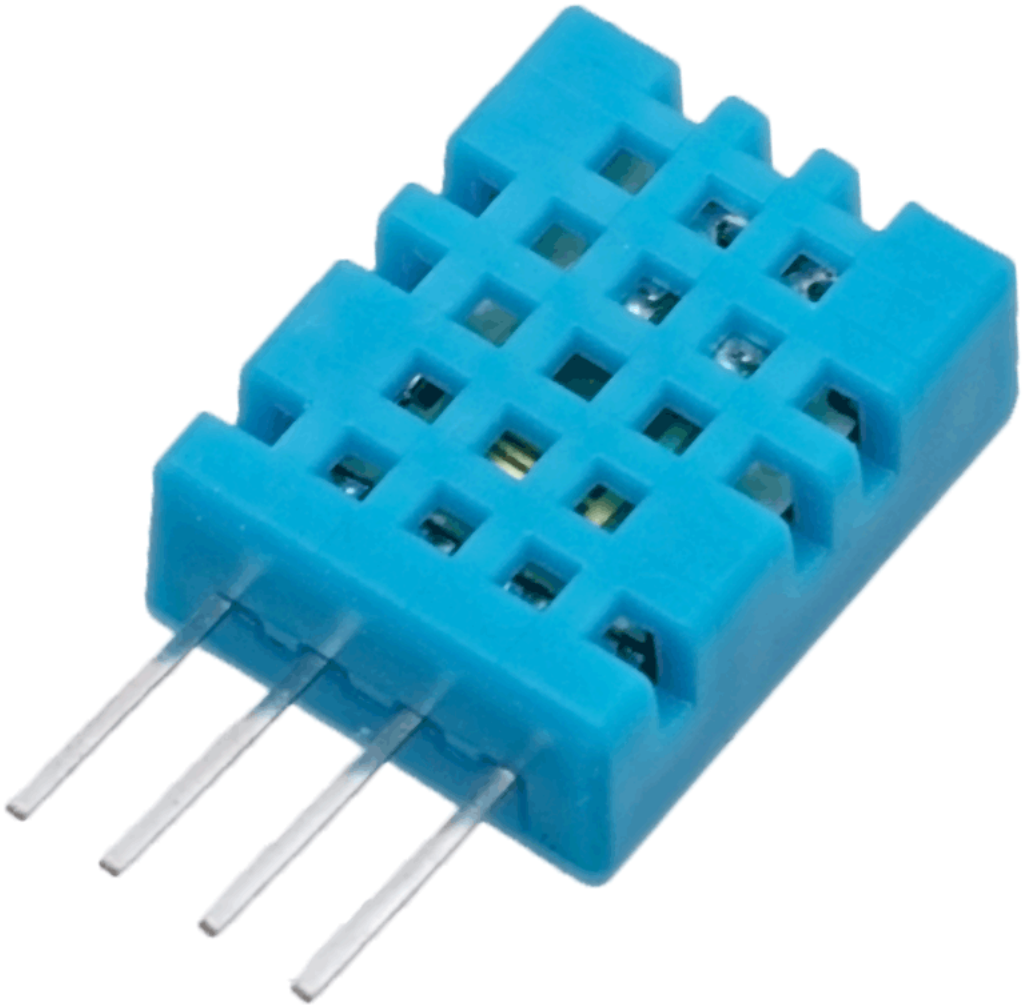


Figure 3.10: DHT11 humidity and temperature sensor.

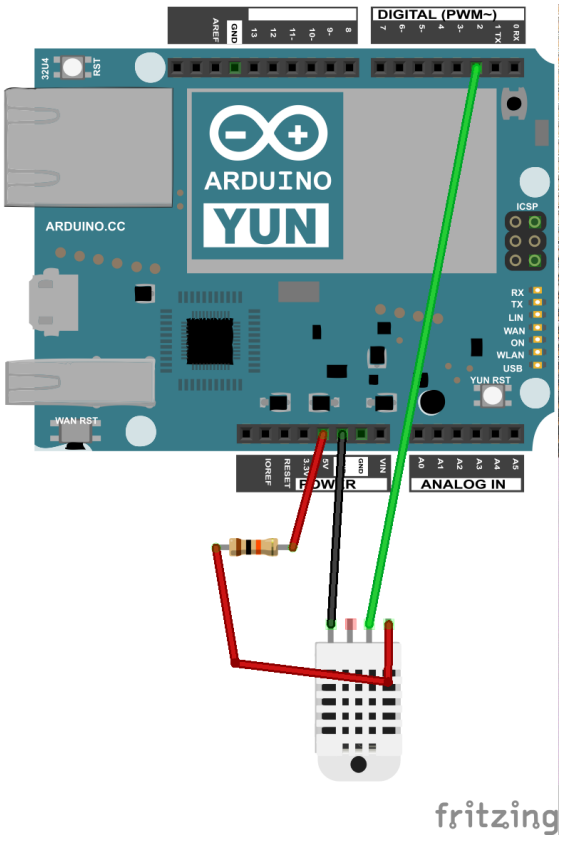


Figure 3.11: Humidity Sensor DHT22 Breadboard.

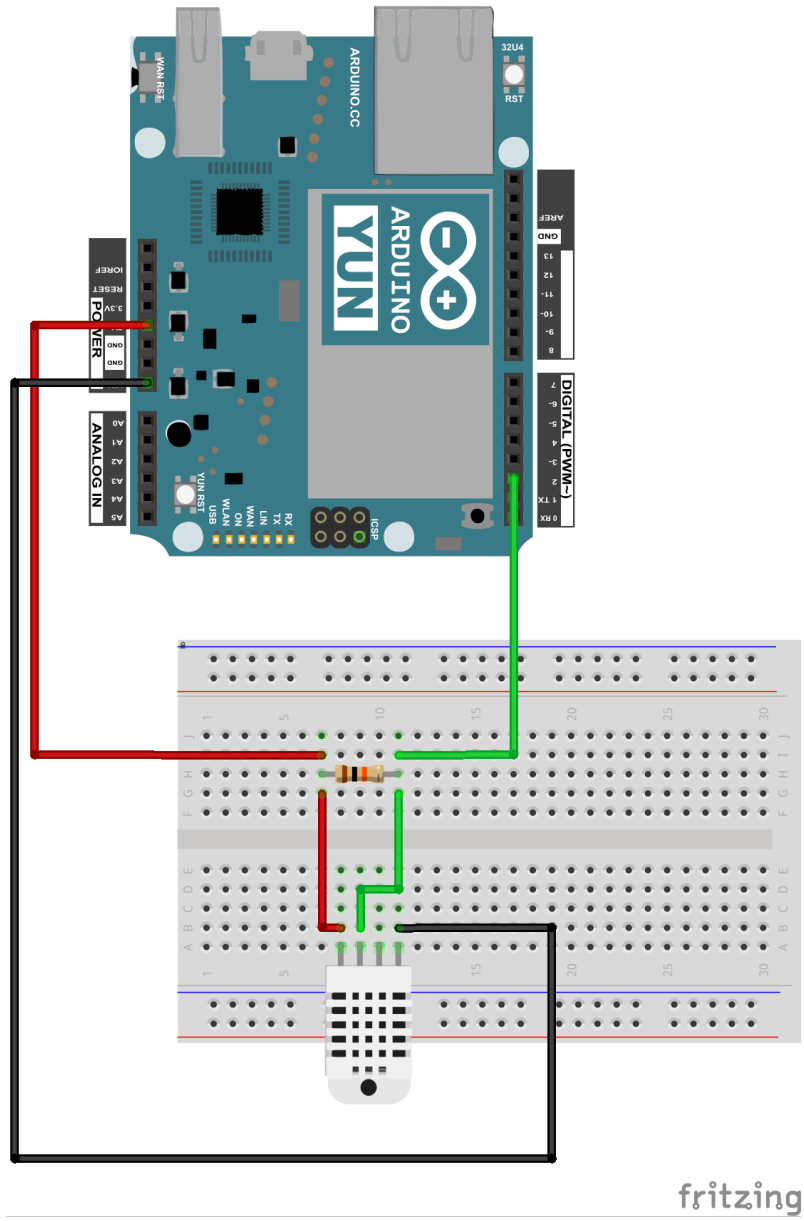


Figure 3.12: Humidity Sensor DHT11 Breadboard.

3.2.5 MQ135: Gas sensor

This is a gas sensor [Figure 3.13], and we will use it to measure air quality. This sensor does not have a figure in fritzing, so I used a gas sensor that has the same output, and int the figure 3.14 we can see how to connect it to the arduino.

Figure 3.13: MQ135 Air Quality sensor.

3.2.6 BreadBoard with all the sensors

In the figure 3.15 we can see the final prototype with the DHT22 sensor, and in the figure 3.16 with the DHT11 sensor.

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.

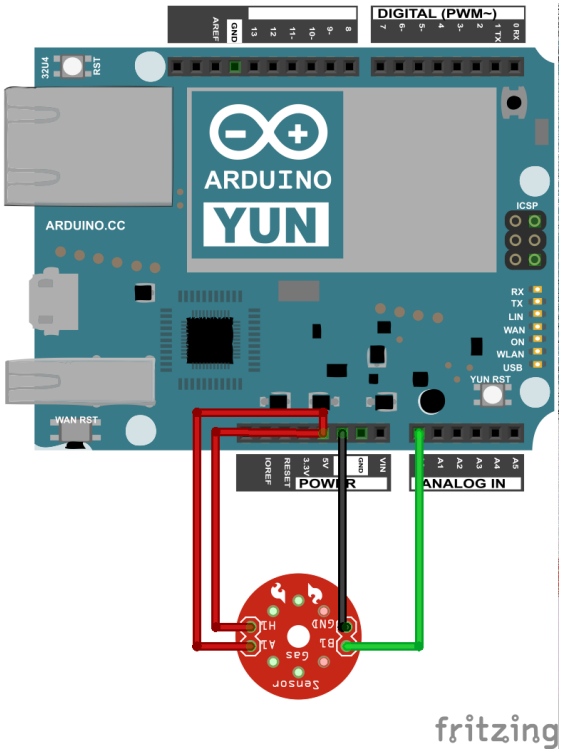


Figure 3.14: Gas Sensor Breadboard.

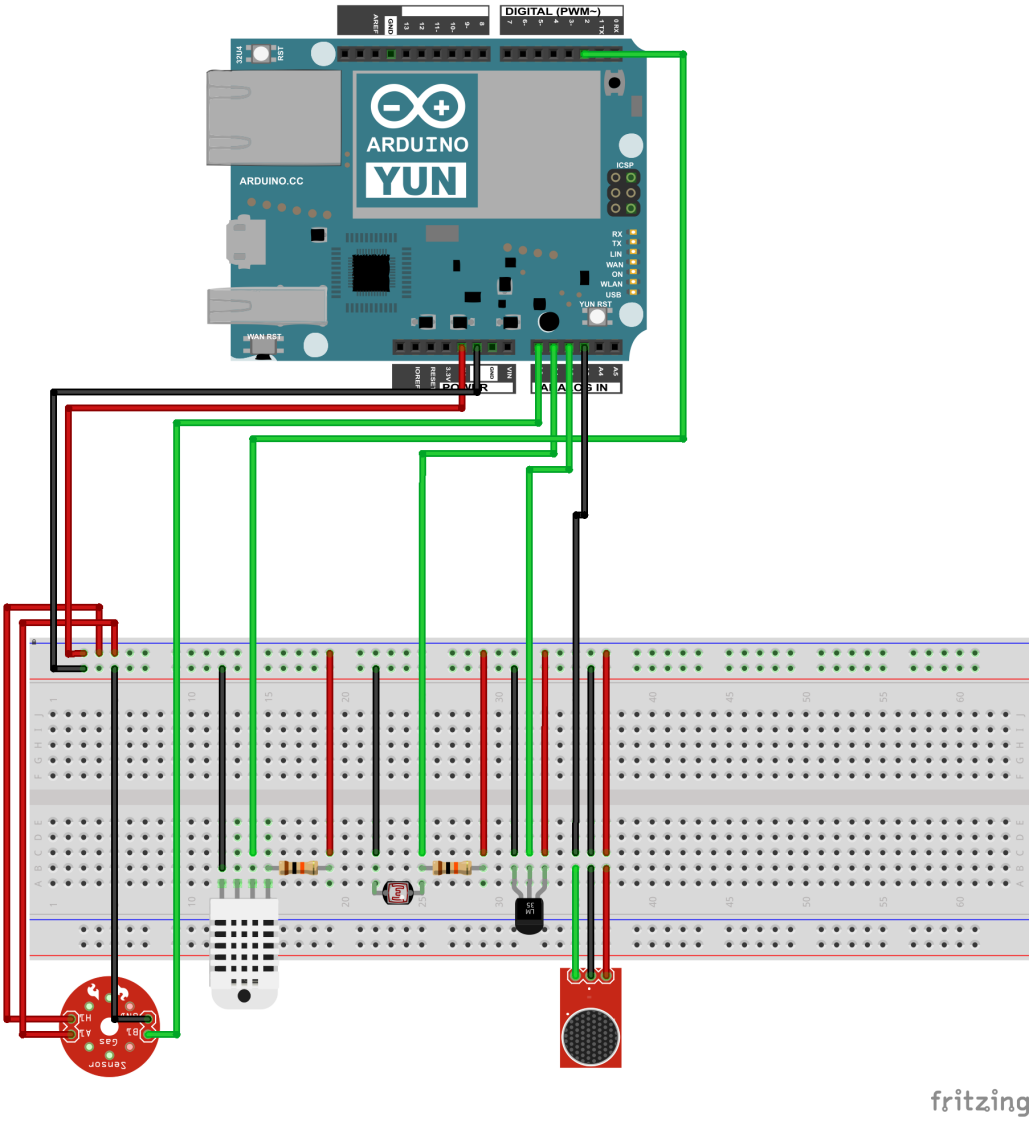


Figure 3.15: Sensor node Prototype with DHT22.

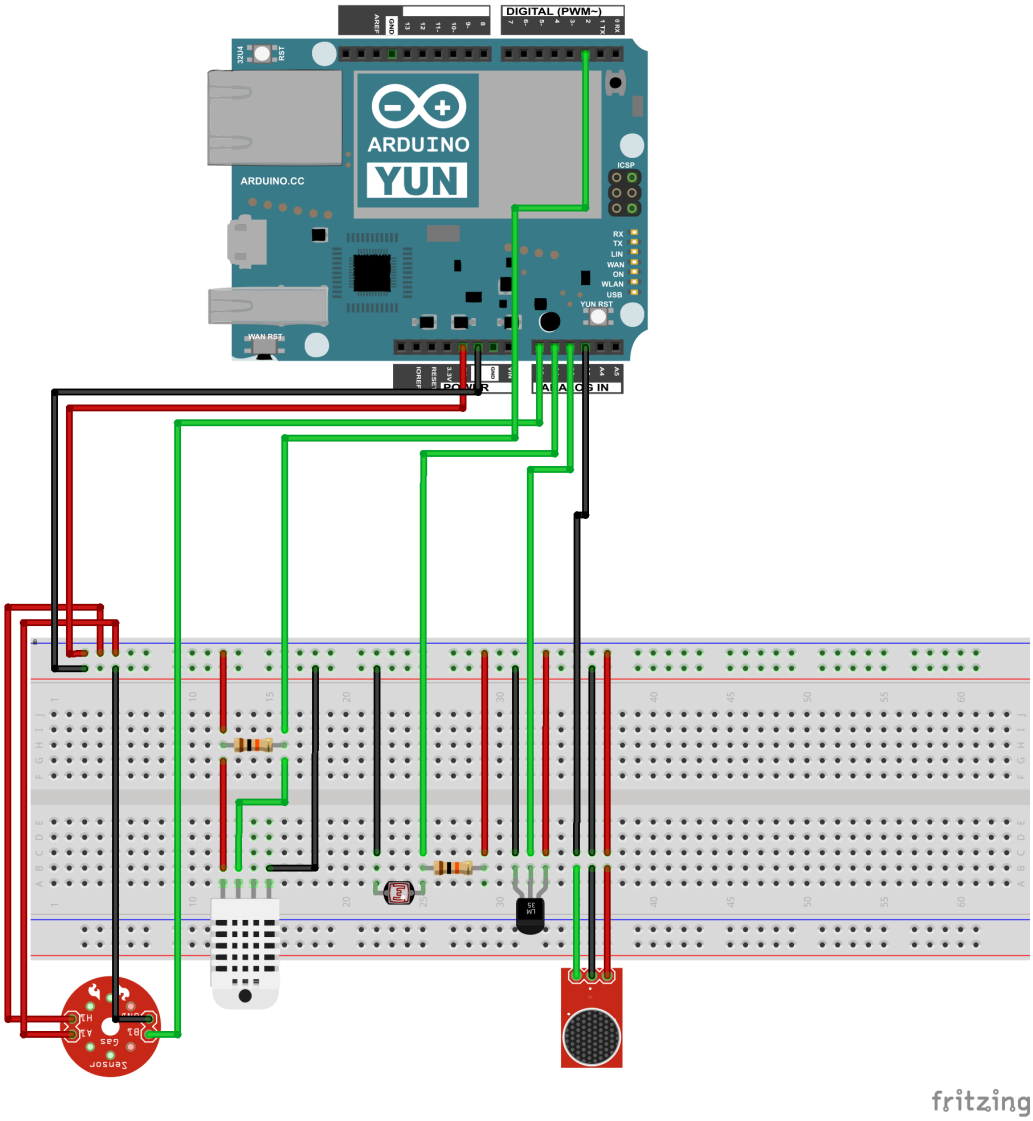


Figure 3.16: Sensor node Prototype with DHT11.

3.4 Guifi network and opencities

Guifi [4] is the network where the arduino's will be deployed, and the one providing the access to Opencities through the Internet.

Opencities[5] is the opendata services that we have chosen, the strengths of opencities are:

- They give us free storage.
- Easy API to upload and download the data.
- 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 it 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 it.

This application it will be tested on a Sony Xperia Z1, with an Android 4.4.2.

[4] https://www.guifi.net/

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

Chapter 4

BOTTOM UP SENSOR TESTBED

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

4.1 Collect and Send

We will need to scripts, one to collect the data, and other to send it. This is because the memory to run an arduino sketch is very low, and the creation of the message to opencities is too big. That is why we use a python script called by the arduino sketch.

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 a logData file, and call the python script with the collected values and an unique ID, and the python script have to create a 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 sensors (DHT22 and DHT11) which need an external library.

To read and write into the logData file we need the FileIO library[2], and to call

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

[2] `http://arduino.cc/en/Reference/YunFileIOConstructor`

the python script we need the Process library[3].

The figure 4.1 explain how the arduino sketch works.

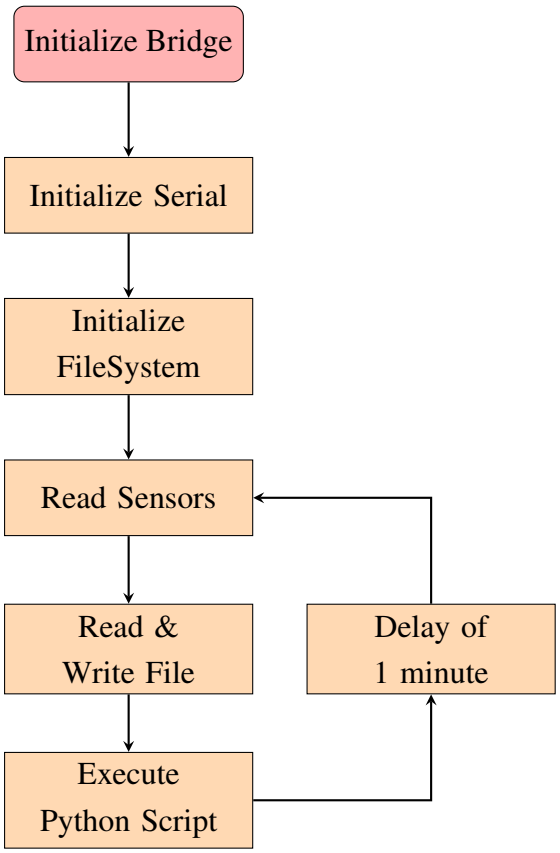


Figure 4.1: Arduino sketch Flow Chart.

4.1.2 Communication with opencities

The communication with opencities is done by a python script, to do this we need a set of libraries installed in the arduino.

We need the sys and datetime library that are already installed, but the geopy, geojson, and httplib2 libraries has to be installed.

This are the steps I followed to install the libraries:

[3] `http://arduino.cc/en/Reference/YunProcessConstructor`

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 python script works.

4.2 Android app

4.2.1 Summary

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

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

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

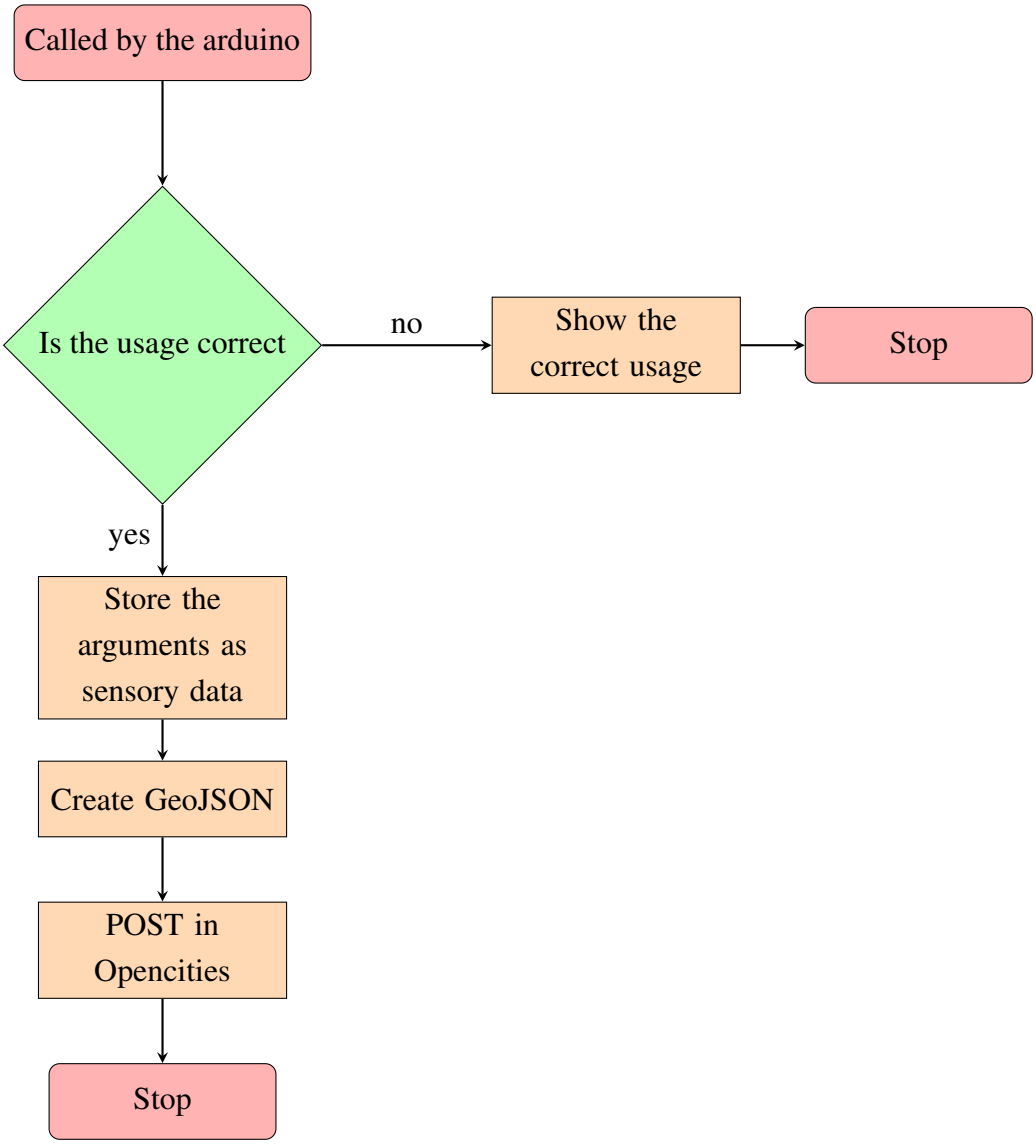


Figure 4.2: Python Script Flow Chart.

Figure 4.3: App Screenshot.

the Marker button, the user decides whether the markers are displayed or not, and the next buttons refer to the type of sensor data to show as markers and/or 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:

- 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.

In the figure 4.6 and fig:ClassDiagram2 are the class diagram:
The explanation of the code is in the following flow chart 4.8:

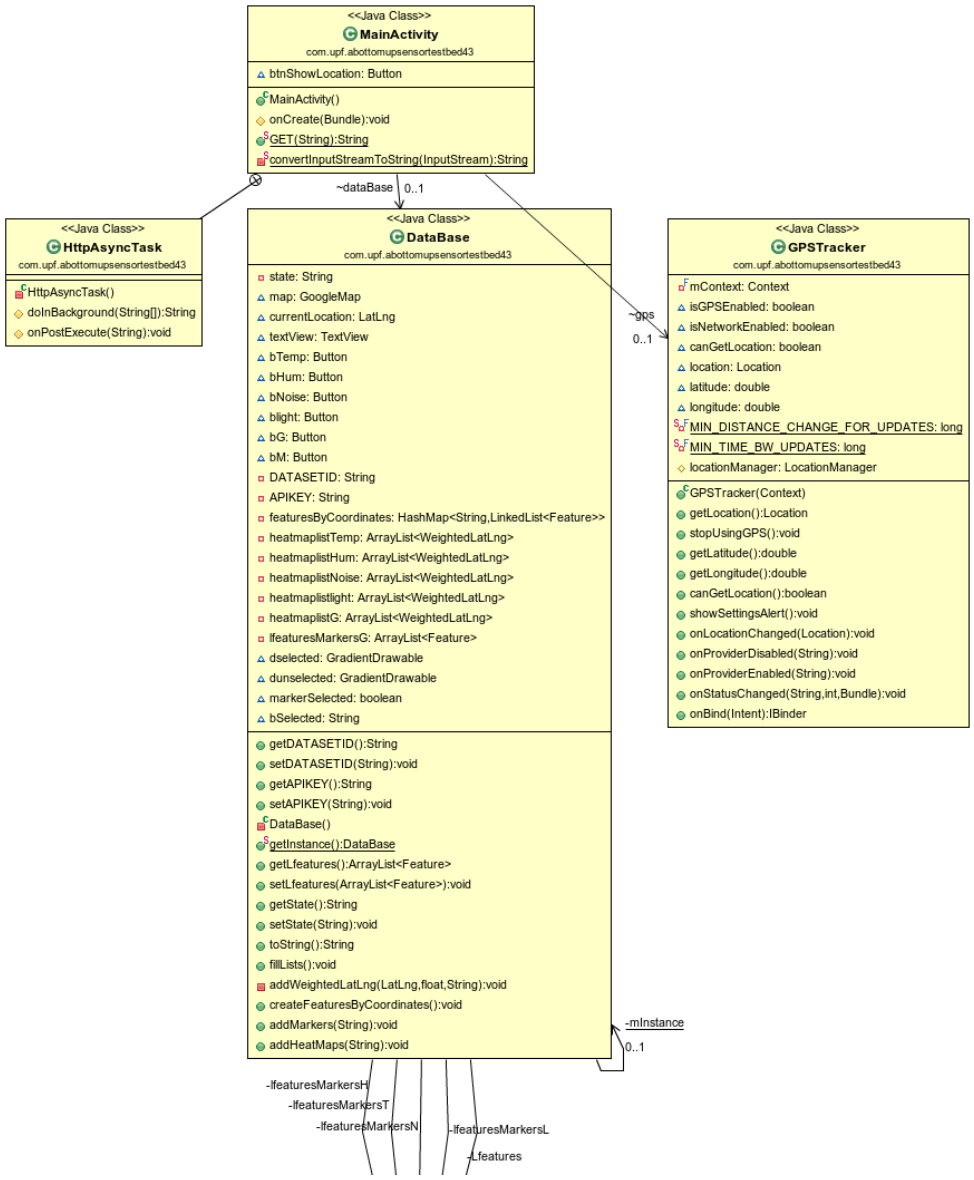


Figure 4.6: Class Diagram of the Android App part 1.

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

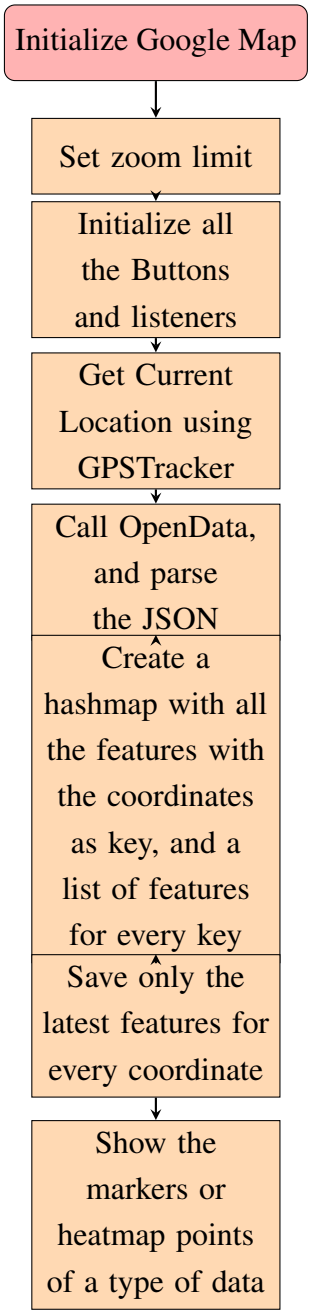


Figure 4.8: Android App Flow Chart.

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).

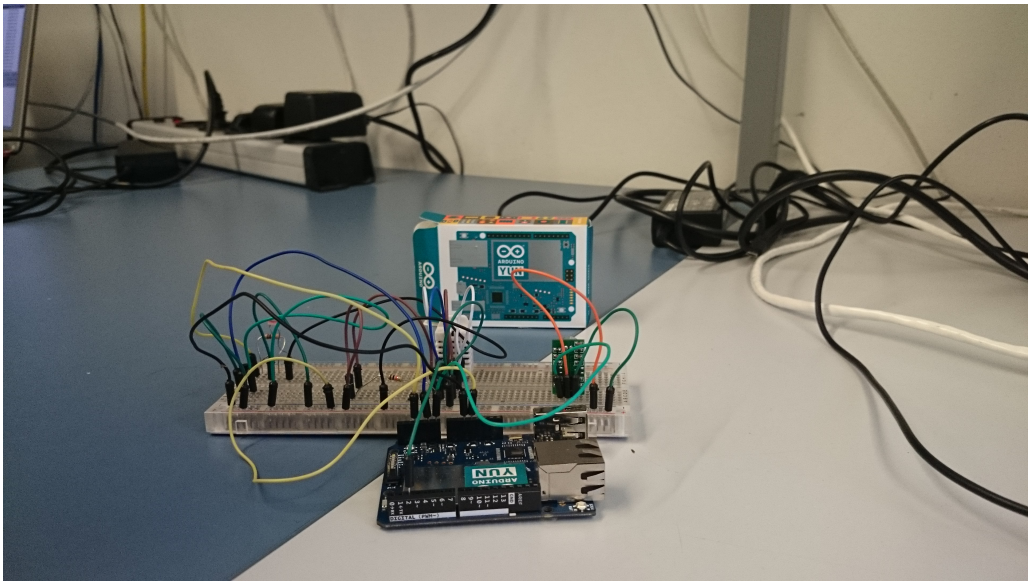


Figure 5.1: TestBed Prototype.

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`

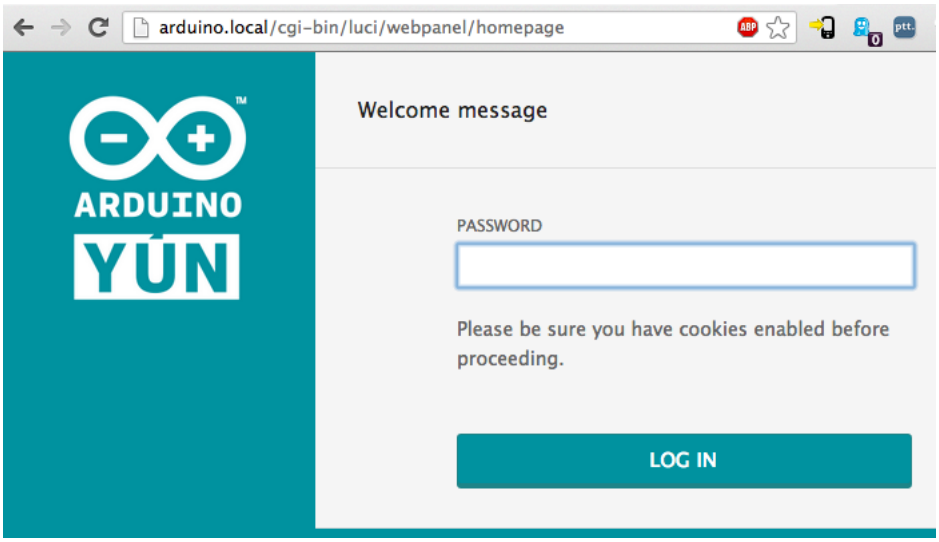


Figure 5.2: Yun web Password.

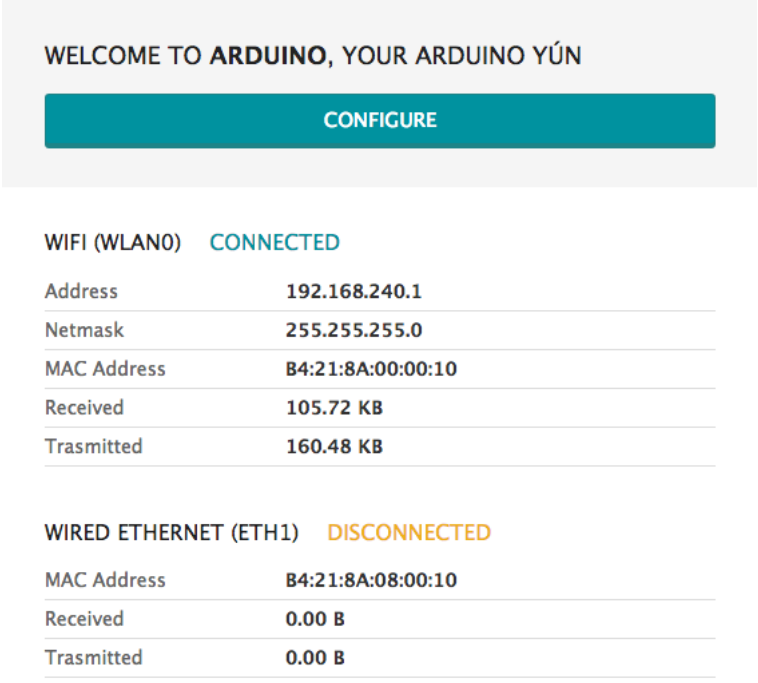


Figure 5.3: Yun web Diagnostic.

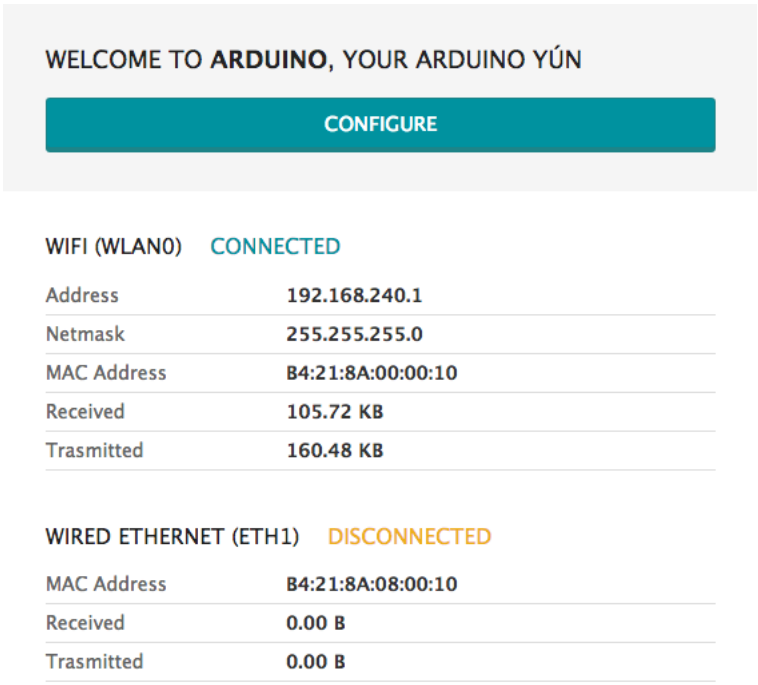


Figure 5.4: Yun web Configuration.

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 3.15 or figure 3.16.

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

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].

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.

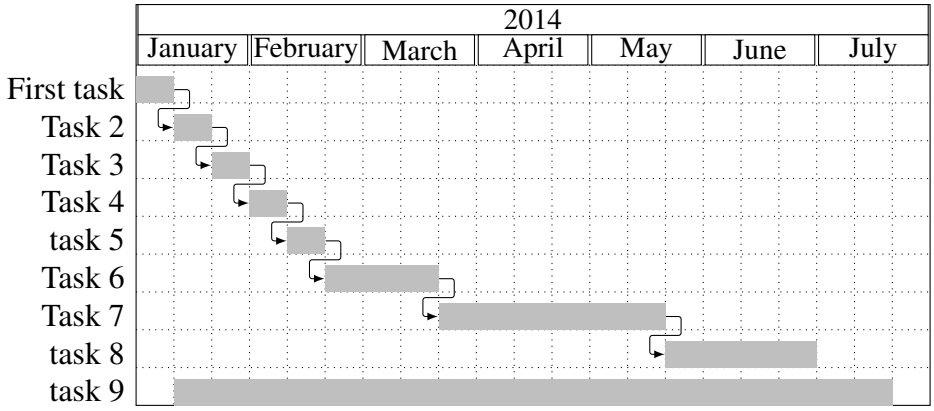


Figure 8.1: Gantt Chart

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