A bottom up sensor testbed

Sergio Almendros Díaz

TFG UPF / YEAR 2014

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

Acknowledgments

Abstract

This project aims to deploy sensor node network into the guifi network, an open, free, and neutral network, and the visualization of this data in an intuitive way that helps final users to understand the meaning of these data without any previous scientific knowledge.

The main objective is to give citizens environmental conscience so that they are aware of their surroundings and could act accordingly if anything stands.

This way anybody can act consequently if the sensors detect harmful situations, such as, not going to a park where the gas sensor detects a bad air quality.

From this point, arduino Yun is the best choice to be the main component of the sensor node, and through a series of sensors attached to it and the Internet connection that provides guifi, the node can communicate with a storage platform of sensor data, in this case, opencities. Finally, an android application downloads the sensor data from the servers of opencities and display them on a map.

Resum

Cuando el Abstract esté perfecto, lo traduciré al catalan.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

The development of this project has been divided into three phases: collect data from environmental sensors, display it to end users in an intuitive way, and make a testbed to demonstrate if it had a good or bad performance.

As a sensor node has been used an Arduino[Arduino, 2014] 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[1] nodes and send the sensor data to a sensor platform, like opencities[Opencities, 2014].

The Bottom-up[2] pattern has been 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 being provide and use by the end users.

In this project a Android application has been made to visualize the sensor data and make it more accessible to the end users.

A sensor testbed has been made to gather sensor data, and test the technologies used as nodes to see if they are the best option.

This project is an easy way to understand the importance of sensor networks and how they can help us to take conscience about our environment.

In the following chapters it has been explained the state of sensor networks nowadays (Chapter 2), which technologies had been used (Chapter 3), and how the project had been done (Chapter 4).

There has been a testbed deployed and some data analysis (Chapter 5).

Finally, the conclusions (Chapter 6) and future work (Chapter 7).

[1] `http://guifi.net/`

[2] `http://bubforeurope.net`

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 a few projects on sensor networks deployed has been introduced:

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 [smart city, 2014].

2.2.2 Santander smart city

Santander has his own sensor network testbed for environmental monitoring, outdoor parking area management, or traffic intensity monitoring.

[santander, 2014]

2.3 Companies

In the following points some companies that are in the business of sensor networks have been described.

2.3.1 Smartcitizen

Smart Citizen[1] is platform that allows a user to buy a node based on Arduino to monitor the environment as it can be seen in the figure 2.1, upload this data to their own database to anybody can see it.

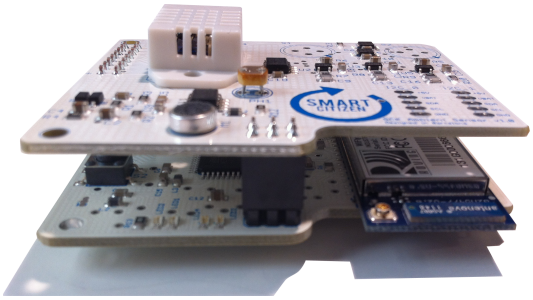


Figure 2.1: Smart Citizen Node.

2.3.2 Libelium

This company is an Internet of things platform provider[2], which supply an open source sensor platform for the Internet of things. They have a variety of products, here are some interesting ones:

[1] http://www.smartcitizen.me/
[2] http://www.libelium.com/

- e-Health: A sensor shield for Arduino and Raspberry Pi for body monitoring: pulse, oxygen in blood, airflow, body temperature, electrocardiogram, glucometer, galvanic skin response, blood pressure, patient position, and muscle/eletromyography sensor (Figure 2.2).

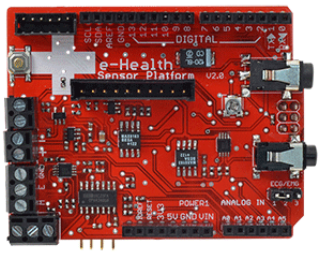


Figure 2.2: Libelium ehealth.

- Waspmote: A sensor node where we can attach more than 60 sensors, solar powered, and though for adjust into street light posts. (Figure 2.3).

Figure 2.3: Libelium Waspmote.

- Smart Water: Is a wireless sensor platform for water quality monitoring, it also provides real-time data. (Figure 2.4).

2.4 Opendata services

The sensor networks are useless if the data is not stored, although the data could be saved in the device, it would be too expensive to collect it. That is why an opendata service is used.

Figure 2.4: Libelium Smart Water.

2.4.1 Opencities

Opencities[Opencities, 2014] is a platform to browse, visualize, and download open data from different participants. They a very simple API to upload and download data, a web page to visualize the stored data, and the possibility that the project can contribute to their system, and so they to the project.

2.4.2 Xively

Xively[3] offers an Internet of Things platform as a service, basicaly it let you store sensor data, download it, and visualize it in graphics.

2.4.3 Sentilo

Sentilo[4] is a open source platform for Smart Cities, it allows the user to use their own service to store the data, but not many. Their goal is that anyone who wants the sentilo platform will have to installed it in their server, and then use it. It also provides a interface to show the data.

2.5 Sensor boards

In this section it is discussed some of the options for a sensor node.

2.5.1 Arduino YUN

The Arduino YUN is a microcontroller board with two processors as it can be seen in the figure 2.5, an ATmega32u4 (Arduino), and an Atheros AR9331 (For a Linux distribution named OpenWrt-Yun). It also have a Ethernet and WiFi module, a

[3] `https://xively.com`
[4] `http://www.sentilo.io`

USB-A port, a micro-SD card slot, 20 digital input/ouput ping, 16 MHz crystal oscillator, 16 MHz crystal oscillator, ICSP header, and 3 reset buttons.

Figure 2.5: Arduino YUN.

The great thing about the arduino YUN is that the processor for the arduino sketches can communicate with the Linux processor through the bridge library, which allows you to write python scripts and execute them. In the figure 2.6 it can be seen.

A power over ethernet module (PoE) can be attach to the arduino, which is perfect for the guifi network, as their nodes are PoE.

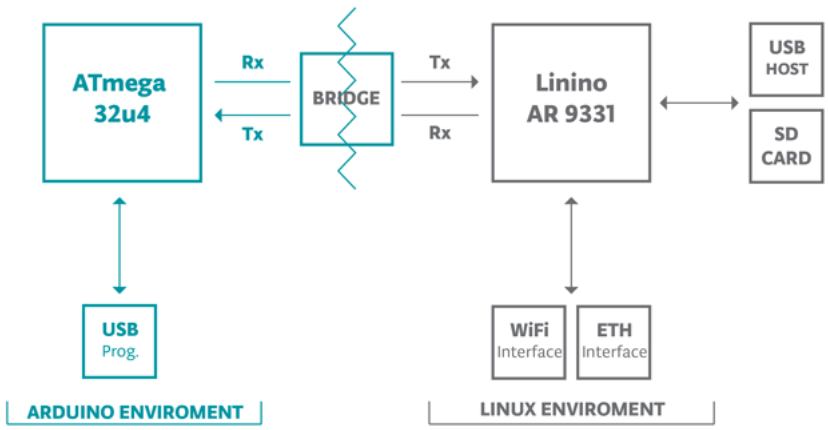


Figure 2.6: Arduino YUN Bridge.

2.5.2 Raspberry Pi

Raspberry Pi[5] is a single-board computer, they have two models, A and B. The model B is more appropiate for this project because it has an Ethernet controller. It is composed by an HDMI Micro USB, and USB 2.0 connector, an SD card slot, Input/Output (GPIO) pins, an RCA connector, an audio jack, an Ethernet controller, and a Broadcom BCM2835 processor.

It is very easy to attach sensors to it, and it supports Linux environment like raspbian. In the figure 2.7 we can see the raspberry pi B.

Figure 2.7: Raspberry Pi model B.

2.5.3 Picoboard

Puede que cambie Picoboard por BeagleBone

PicoBoard[6] is a board to interact with the world, it can be programmed by Scratch projects. It is less flexible than the others, it is composed by a button, a light and sound sensor, a slider, and alligator clips which can be use to build custom sensors. This board can be seen in the figure 2.8.

[5] http://www.raspberrypi.org/
[6] http://www.picocricket.com/picoboard.html

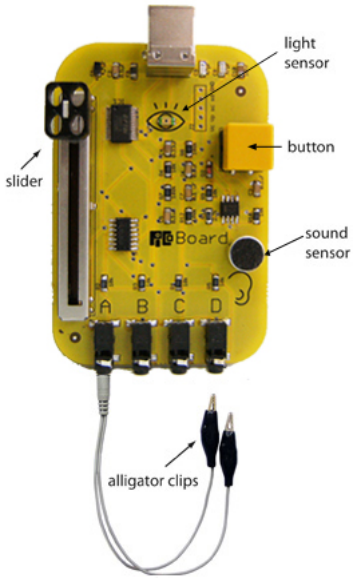


Figure 2.8: Picoboard.

Chapter 3

TECHNOLOGIES

This chapter has been focused in the techonologies used to develop this project.

3.1 Arduino vs Raspberry PI

For this project, it had been narrowed down the sensor node posibilities to an Arduino Yun (Yun) or a Raspberry Pi model B (RPiB).

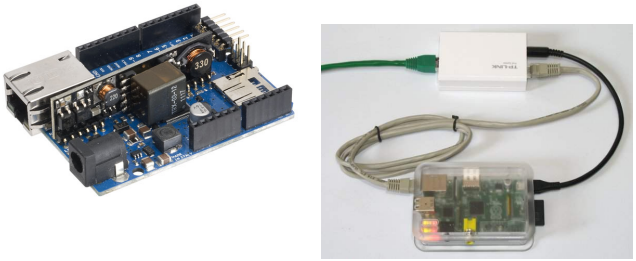
The Yun is a microcontroller while the RPiB is a full computer, which makes it more powerful, but this project does not require too much, so both Yun and RPiB serve.

The two of them had a linux environment, but with the Yun, the normal way to interact is by an arduino sketch. So RPiB will consider that the developer has some linux knowledge, while the Yun is better for beginners, the arduino IDE provides a variety of programs that help to start.

The sensor board is essentially the sensor node, so the size is very important, the smaller the better. In this case, the Yun is smaller. On the other hand, the sensor node will be attach on guifi nodes, which are Power over Ethernet (PoE). For the Yun there is the posibility of a PoE module which it is not available yet, but will be. The figure 3.1a shows and arduino ethernet (similar to the Yun) with a PoE module and does not makes it bigger, and with the RPiB, the module makes it bigger, as it can be seen in figure 3.1b.

Another advantage that YUN offers is the integrated WiFi module, since a person who does not have a guifi node can connect to one, and thus have access to opencities, and of course, to stored data.

Finally, the Arduino Yun had been used instead of the Raspberry Pi basically because of the size that will become the sensor node, but if, in the future was necesary a more powerful procesor, it will be better use a Raspberry Pi.



(a) Arduino Ethernet PoE (b) Raspberry Pi (B) PoE

Figure 3.1: Sensor boards

3.2 Sensors

The goal is to analyze the environment around us, for that purpose, sensors are very useful. It has been decided for the use of environmental low cost sensors that are easily accessible and usable. These sensors measure the aspects that may be more useful for citizens: temperature, light, noise, humidity, and air quality.

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[1].

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. [Instruments, 2013]

3.2.2 Light Dependent Resistor (LDR)

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

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

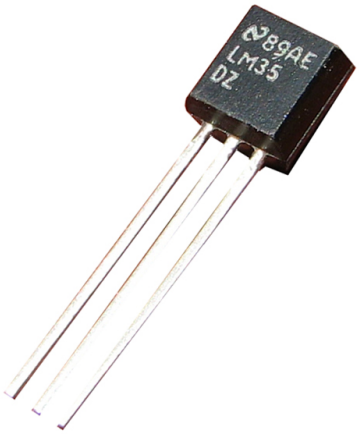


Figure 3.2: LM35 sensor

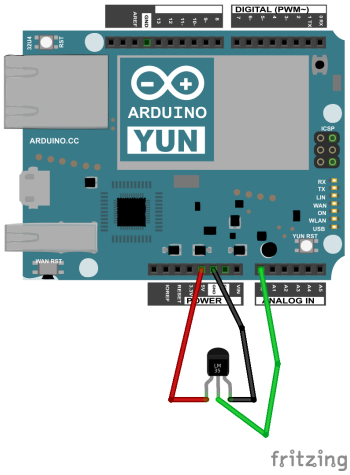


Figure 3.3: Temperature Sensor Breadboard

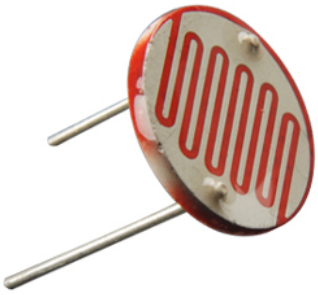


Figure 3.4: LDR sensor

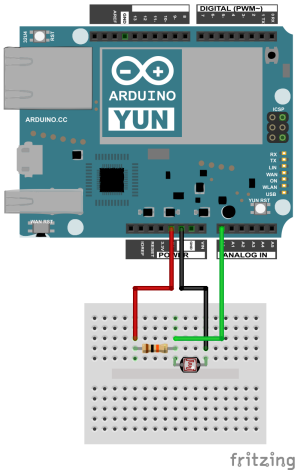


Figure 3.5: Light Sensor Breadboard

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 [Figure 3.6] and [Figure 3.7]. The code to read the noise values is the same for both. In the figure 3.8 it can be seen the way to connect them to the arduino. [Emartee, 2013]

Figure 3.6: Mini Sound Sensor

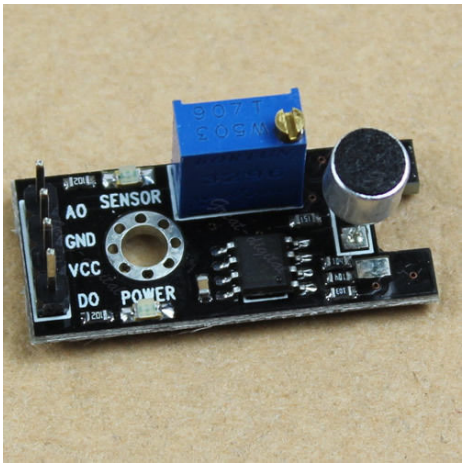


Figure 3.7: Analog noise sensor

3.2.4 Aosong DHT22 and DHT11: Humidity

DHT22 [Figure 3.9] and DHT11 [Figure 3.11] 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[2]. The arduino and the humidity sensor has to be connected as shown in the figure 3.10.

At the time the testbed started, there was only 1 DHT22 sensor, so a DHT11 sensor has been used instead of the DHT22, which change the breadboard a little bit as shown in the figure 3.12. [Aosong Electronics Co., 2013]

3.2.5 MQ135: Gas sensor

This is a gas sensor [Figure 3.13], and is used to measure air quality. This sensor does not have a figure in fritzing, so gas sensor that has the same output has been used, and in the figure 3.14 it can be seen how to connect it to the arduino.

The Mq135 sensor reacts to the concentration of the following gases: NH3, NOx, alcohol, Benzene, smoke, CO2, etc.. **??**.

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

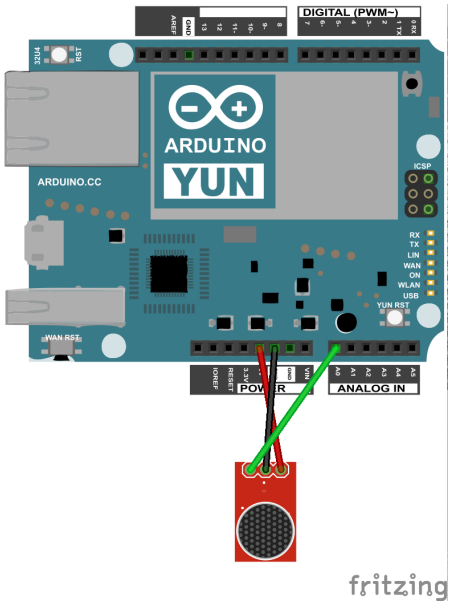


Figure 3.8: Noise Sensor Breadboard.

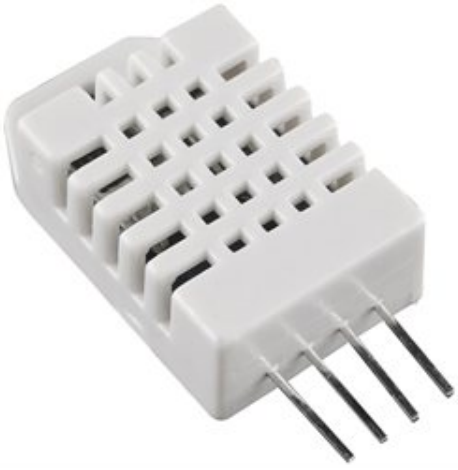


Figure 3.9: DHT22 sensor

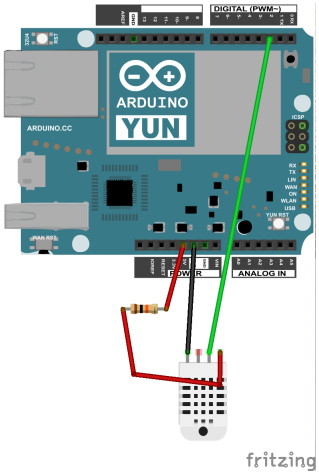


Figure 3.10: DHT22 Breadboard

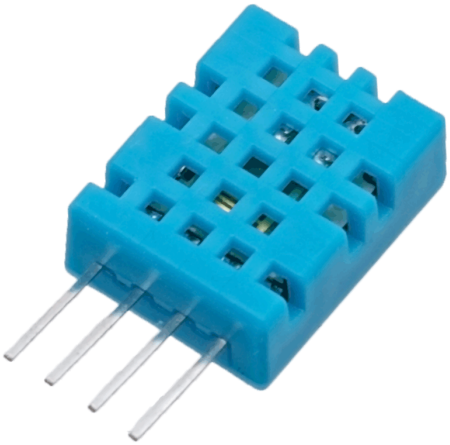


Figure 3.11: DHT11 sensor

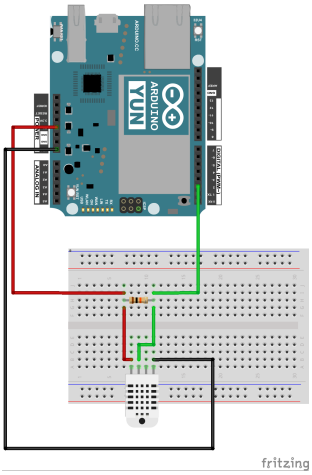


Figure 3.12: DHT11 Breadboard

Figure 3.13: MQ135 Air Quality sensor

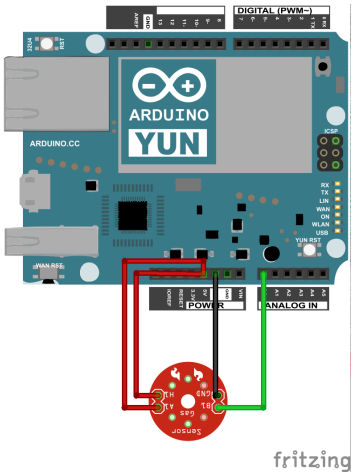


Figure 3.14: Gas Sensor Breadboard

3.2.6 BreadBoard with all the sensors

In the figure 3.15 it can be seen the final prototype with the DHT22 sensor, and in the figure 3.16 with the DHT11 sensor.

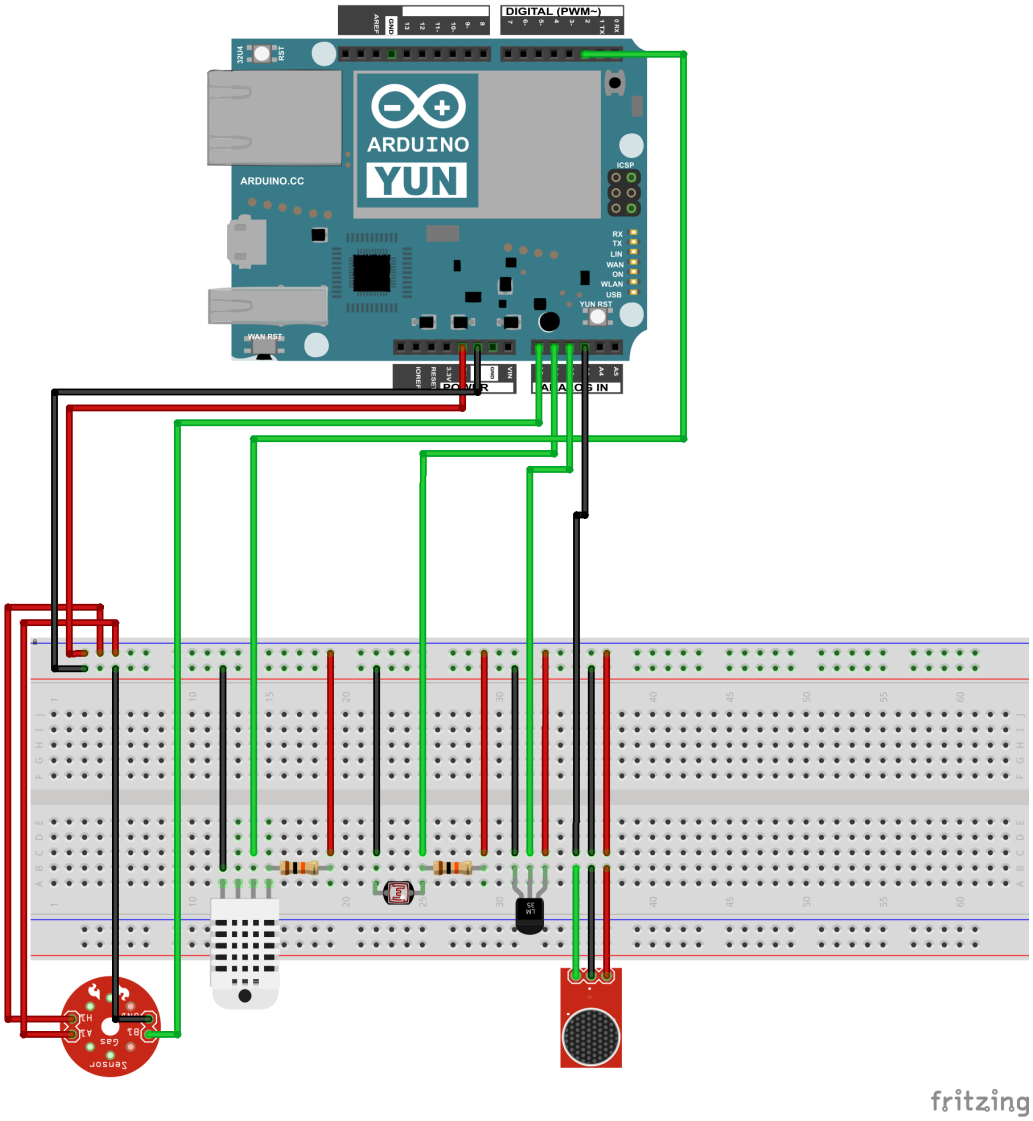


Figure 3.15: Sensor node Prototype with DHT22.

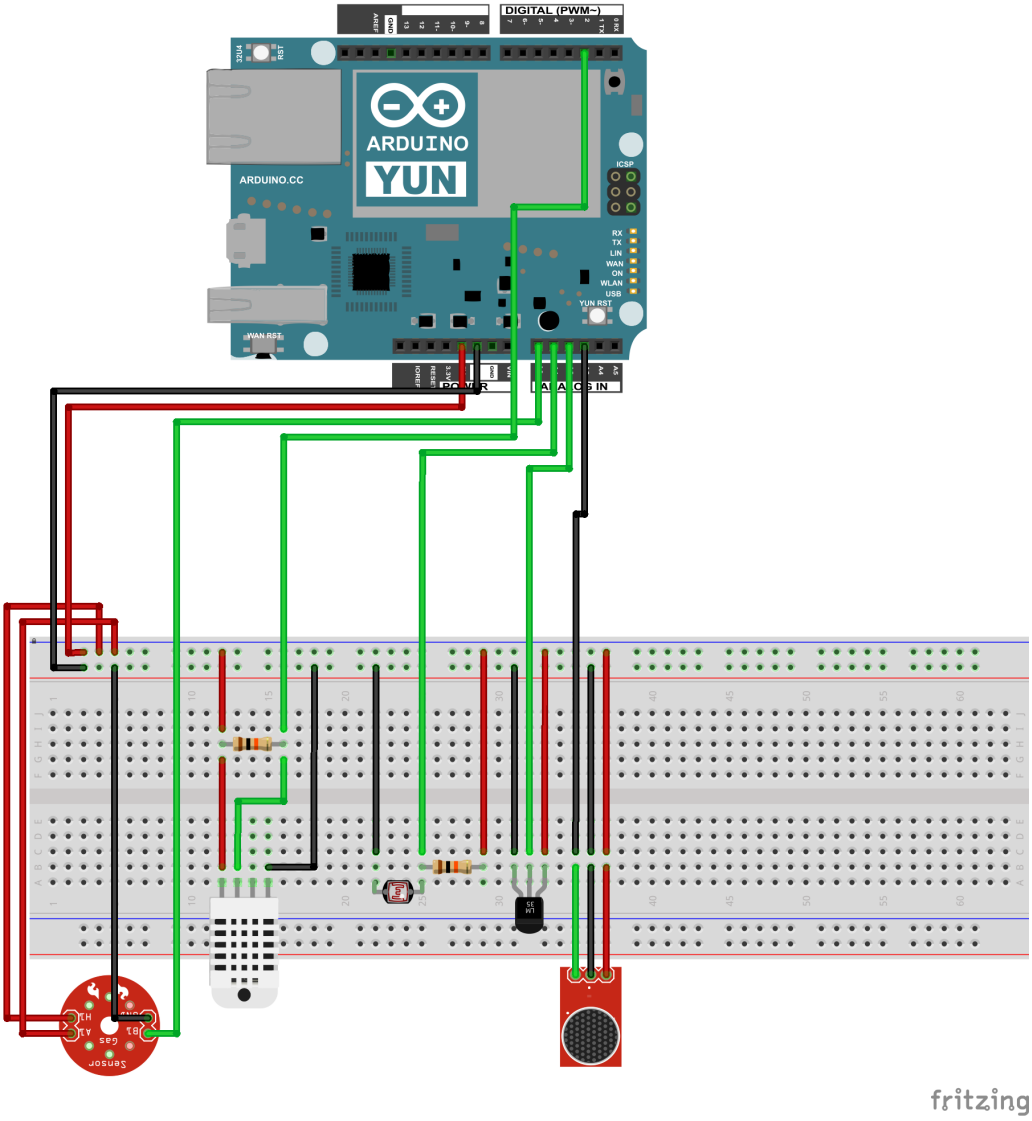


Figure 3.16: Sensor node Prototype with DHT11.

3.3 Upload sensor data

A main part of the project is upload the data from the sensors to a platform so that everyone can access them.

The problem is that is needed to upload five sensor values, consequently the GeoJSON message (Explained in section 3.3.1) takes too much memory, so the Arduino sketch can not handle it.

For this type of operations is possible to run a script in the linux environment through the bridge library2.6.

This being the path that is followed, there are several options for developing this script, for example, programmed in Java, C / C + +, python, etc..

Python has been used because of the following reasons: fast and perfect for prototyping programming, the code is shorter and therefore, easier to understand, especially if you want to improve in the future.

3.3.1 GeoJSON

A GeoJSON[3] is a format for encoding a variety of geographic data structures. The GeoJSON that has been used is a collection of features, every feature contains a geometry object, in our case, a "point" with the longitud and latitud of the sensor node, and some properties required: ID, name, datasetID, datasetName, address, description, timestamp, value of the sensor, and unit.

In the figure 3.3.1 it can be see an example of a GeoJSON message that is used in this project.

3.4 Community network

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

Guifi is a network where the people that are interested put their effort in creating an infrastructure which provides acces to the Internet at a fair price.

Because of their philosophy of participation, Guifi is the perfect network for the deployment of this sensor nodes. Also their networks is large enough to become a useful sensor network, as it can be seen in the figure 3.18.

[3] `http://geojson.org/`
[4] `https://www.guifi.net/`

```
{
    "type": "FeatureCollection",
    "name": "dummy",
    "timeStamp": "2014-06-12T08:54:59.424Z",
    "features": [
        {
            "type": "Feature",
            "tags": [
                "red",
                "tall",
                "cheap",
                "upf"
            ],
            "geometry": {
                "type": "Point",
                "coordinates": [
                    2.18946,
                    41.403809
                ]
            }
        }]
}
```


Figure 3.17: GeoJSON message.

Figure 3.18: Guifi sensor map.

3.5 Storage Resource Broker

Opencities[Opencities, 2014] is the opendata services it has been chosen, the reasons opencities has been chosen are:

- The developers are at UPF, this helps because the platform can be tested, and because they can help solve problems and make these part of the project more easy.
- Easy API to upload and download the data.

The role of Opencities is to be the intermediary between data creators and those who want to see them. In Opencities, the sensor data is uploaded into a dataset. Every user have an unique API key, and can have one or more datasets. To distinguish, the datasets have a unique ID.

3.6 Visualization platform

For the purpose of the sensor data visualization there are several options, in plain text, in a map, in graphics, etc. In this project is used a map to display the data. This can be done on a web page, in a mobile application, tablet application, etc.

Since the goal of the application is that a user consult it for an small period of time, the best option is a mobile application. Additionally, almost everybody has an smart phone.

From this step forward it has to be chosen in which mobile operationg system it should be developed the application. There are three options, iOS, Android, or Windows Phone. There is another option that is to developed the application in html5, javascript, and CSS, and then compiled it with phonegap, and that will return the application for the three operating system. But that was too much of a challenge, so it was decided to code an application directly to one operating system.

Having decided that the application it has to be for only one operating system, it is more logical to chose the operating system with more market share, to reach as many people as possible.

Android is a mobile operating system from Google, it runs on smartphones, and applications are programmed in Java. Java is a computer programming language that is object-oriented.

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

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.

To make the project easier to understand an image has been made (Figure 4.1.

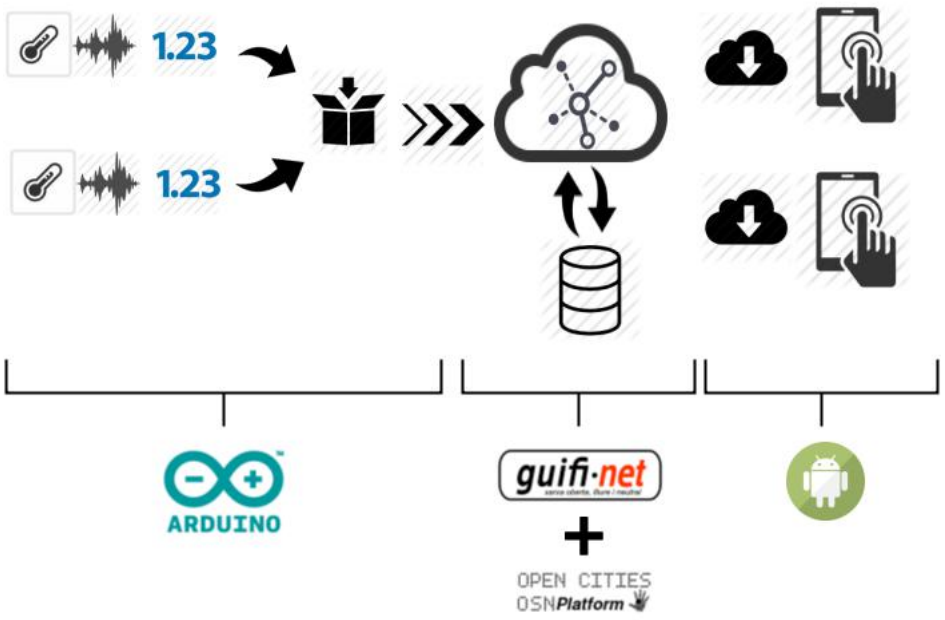


Figure 4.1: reportGeneralView.

4.1 Arduino Development

Two scripts has been needed, 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 GeoJSON message to opencities is too big. That is why a python script has been used, called first by the arduino sketch.

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. Finally the python script have to create a GeoJSON and send it to opencities.

4.1.1 Collect sensor data

To collect almost all the data the skecth does not need to include any libray, except for the humidity sensors (DHT22 and DHT11) which need an external library[1].

The following libraries has been needed to develope the code: FileIO, Process library, Bridge.

This sketch is code in a very simple way, the setup function will initialize the Bridge library to communicate with the linux environment, the Serial library for debugging purposes, and the FileSystem to log the process, and the loop function. The loop will call three functions: readSensors, readFile, and executePythonScript.

- **readSensors:** It call 5 different functions to read every sensor, the goal of doing a separate function is to make the code more clear.
- **readFile:** This function log the process, saves the ID of the message, the sensor values, and a timestamp.
- **executePythonScript:** The script in python located in the SD card is in charge to create the GeoJSON with all the sensor data and upload it to opencities. This script is called by the arduino sketch.

The figure 4.2 explain how the arduino sketch works.

4.1.2 Communication with opencities

First of all, it is needed to create an acount in opencities, and a dataset to store the sensor data. This process gives all the information needed to upload and download data.

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

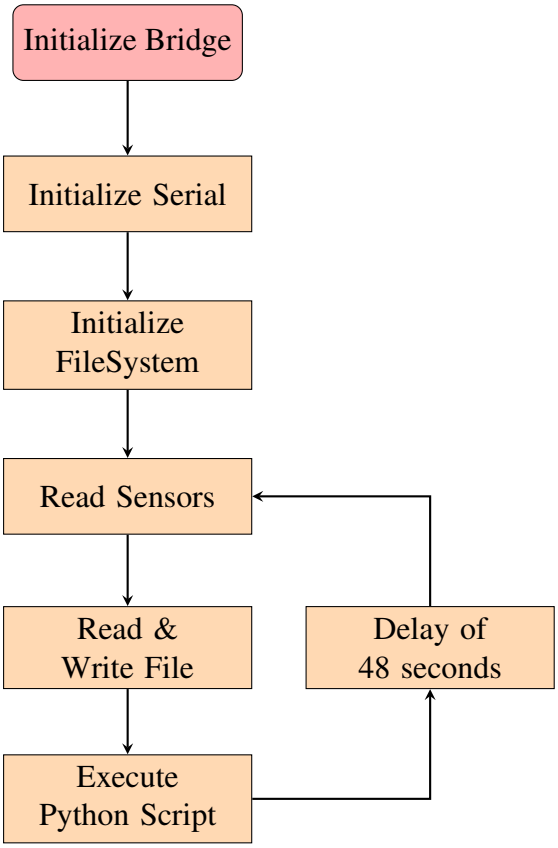


Figure 4.2: Arduino sketch Flow Chart.

The communication with opencities is done by a python script. This are the packages needed into the linux environment of the arduino: distribute, python-openssl, geojson, geopy, and httplib2.

The python script has three process, first it stores all the sensor data pass by the arduino sketch into a class. Then it creates a GeoJSON message with all the data, and all the parameters needed to upload it into opencities. Finally upload the GeoJSON message.

With all this libraries the script can communicate with opencities and store the sensor data recollected by the arduino. The 4.3 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, an android application has been developed which shows the sensor data in a map. The application shows the data in two ways, with markers that show the actual 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 4.4.

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 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 figures 4.4 it can sees the app with the Marker button checked, and without.

4.2.3 Code

First of all, to create this application it has to be used the Google Maps Android API v2[2] for the map view, and the Google Maps Android API utility library[3] for the heatmaps.

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

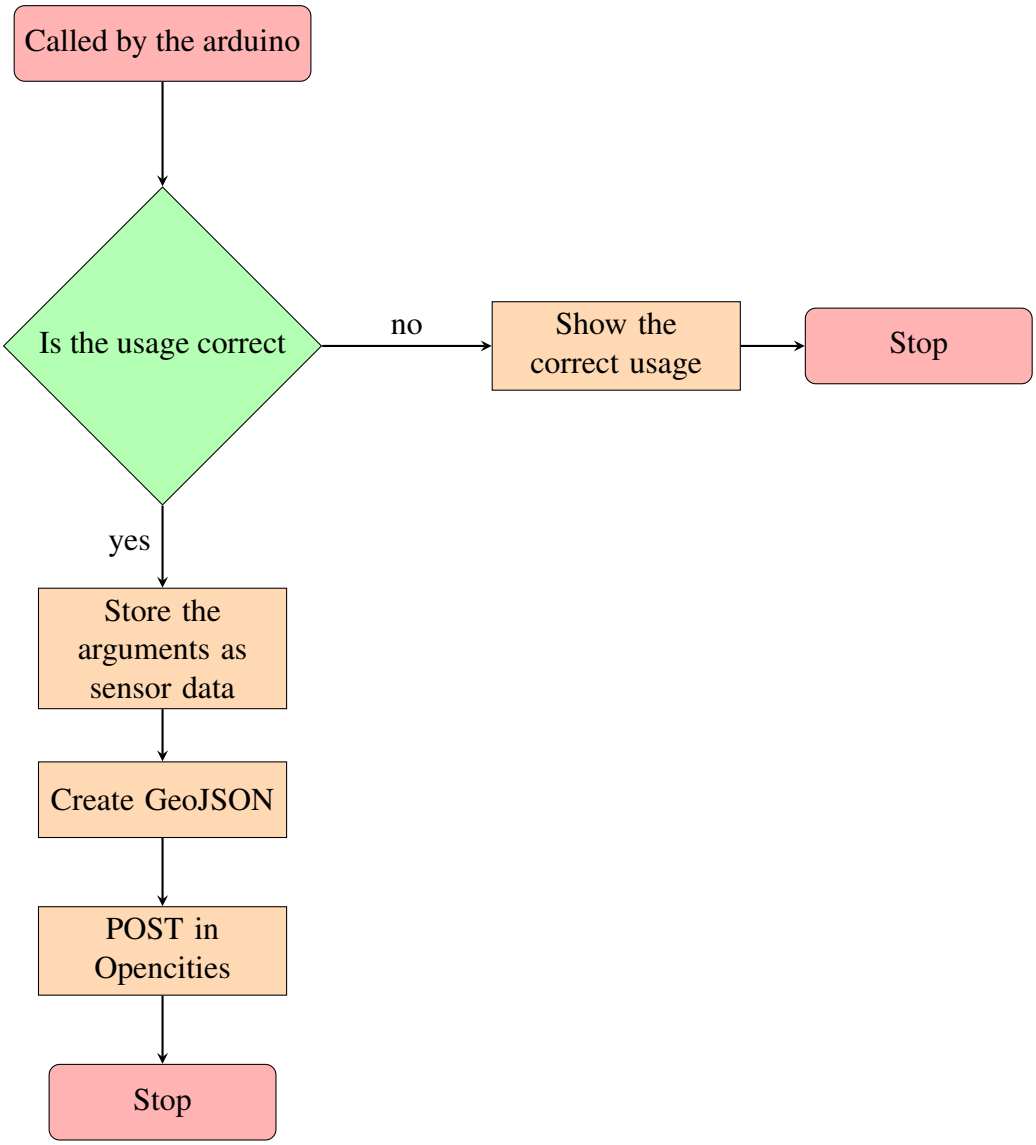


Figure 4.3: Python Script Flow Chart.

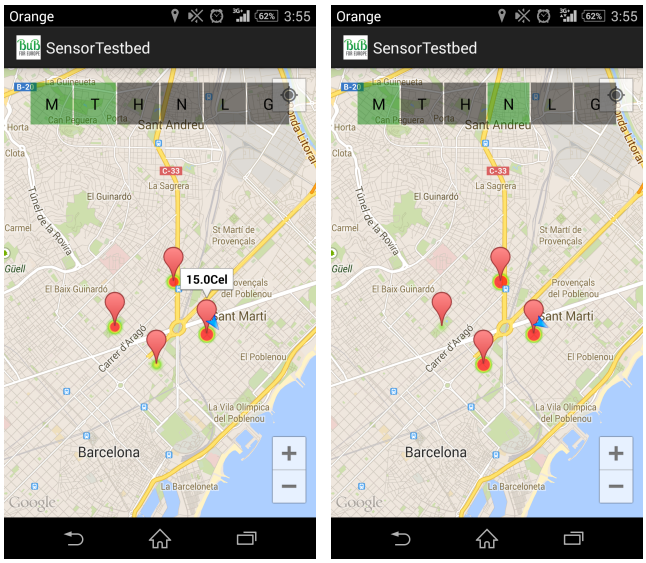


Figure 4.4: App Screenshots

In this application each class has a defined function, starting with MainActivity, which is the one that intialize the other clases, calls all the functions needed in the other classes to get the application started.

It also has all the code to interact with the interface. For example, if the user clicks on the marker button, this class has to call all the necessary functions and display all the markers into the map.

Next comes the dataBase class, which, as the name suggest, is the class that stores and processes the data downloaded from opencities. This is a Singleton class, which means that there will only be one instance of this class and any instance of any class could access to the data stored on it.

To stored the data from opencities, there has been created a set of classes equivalent with the GeoJSON message downloaded to avoid confusion. At the end, the data is stored in features, which is composed primarly of Geometry and Properties.

Then a class called HttpAsyncTask that handles the communication with opencities and, using an instance of dataBase, stores the downloaded data.

Finally, a GPSTracker class is responsible for obtaining user location to place the map on the user location, because it will be the place that the users want to see.

The class diagram, in a reduce way, can be seen in the figure **??**

The explanation of the code is in the following flow chart 4.6:

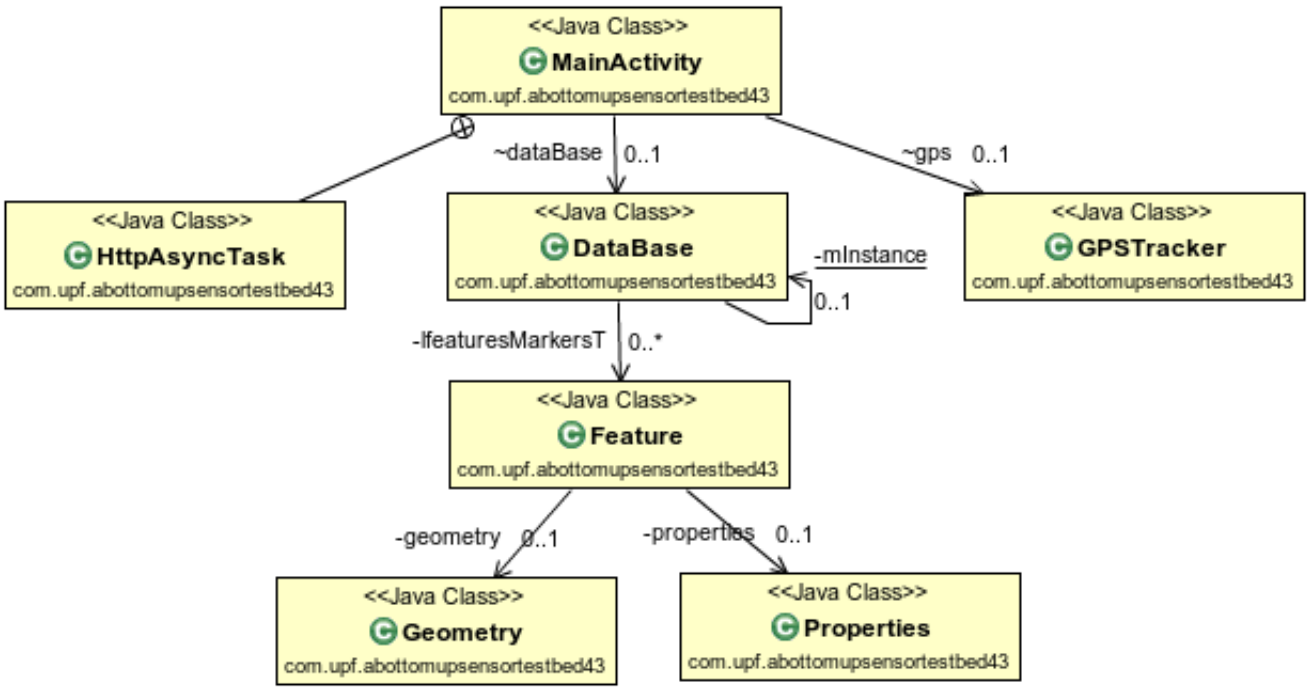


Figure 4.5: Reduce Class Diagram of the Android App.

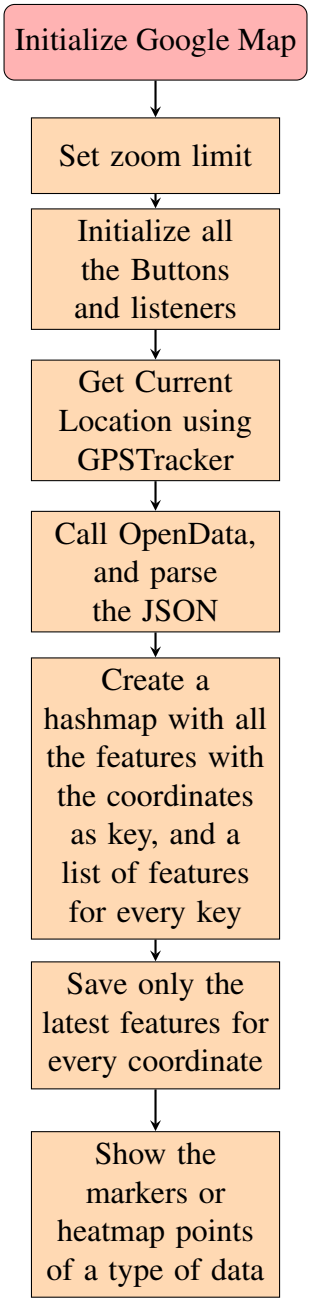


Figure 4.6: Android App Flow Chart.

4.3 Repository

All the code, report, figures, etc has been stored in a public repository, so anyone could see it, downloaded a copy, and change it if necessary. This repository is in

During all the project there have been constant uploads of new information, as it was evolving.

The

Chapter 5

TESTBED DEPLOYMENT AND 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).

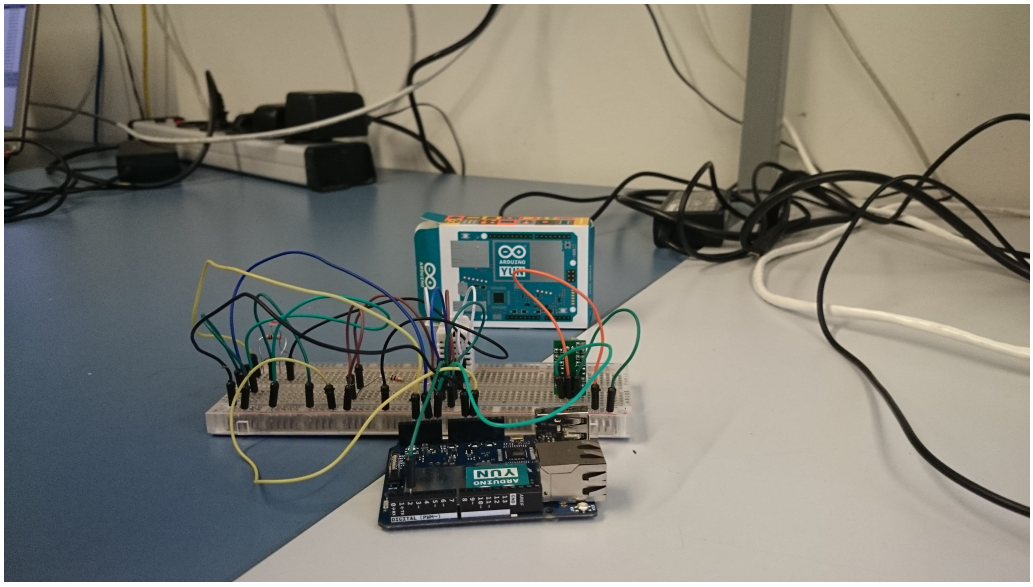


Figure 5.1: TestBed Prototype.

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 Internet connection has to be provide 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 power the arduino YUN.
2. The arduino will create his own WiFi network (ArduinoYun-XXXXXXXXXXXX), and with a computer connect to it.
3. When it is connected to the YUN network, go to a web browser and go to `http://arduino.local` or `192.168.240.1`. Put the password, which is "arduino". as it can be seen in the figure 5.2.

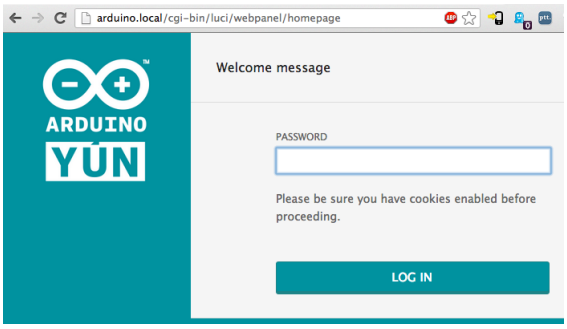


Figure 5.2: Yun web Password.

4. The next page will be an information page, click on the configure button 5.3.

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

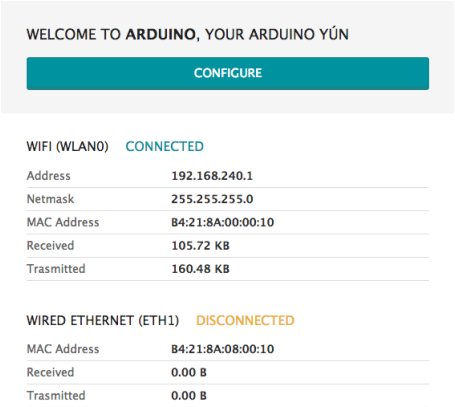


Figure 5.3: Yun web Diagnostic.

5. Give a unique name to the Yun, and the network to connect 5.4.

6. Press the configure & restart button.

7. Finally connect to same network as the Yun is connected.

5.1.2 Install necessary packets

The arduino will run an arduino script and a python script, for python some packets have to be installed.

To install any packet, connect to the arduino (The password has to be "arduino"):

ssh root@X.Y.Z.W

Now 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 create the some directories, so once the ssh command has been done, go to "/mnt/sda1", make the following comands:

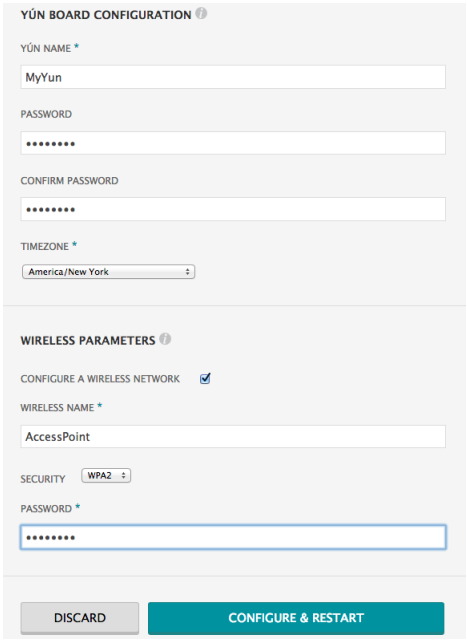


Figure 5.4: Yun web Configuration.

mkdir arduino
cd arduino
mkdir www

Copy the python script "main.py" into the SD-Card. There are two ways to do this, put the SDCard into a computer an save the files in there, or 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 the python step is done, the sensors has to be 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 the IDE: arduino 1.5.5 has to be used. 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 them 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.

In the figure 5.5 there are the nodes deployed, the location try to be a little bit different to get distinc values.

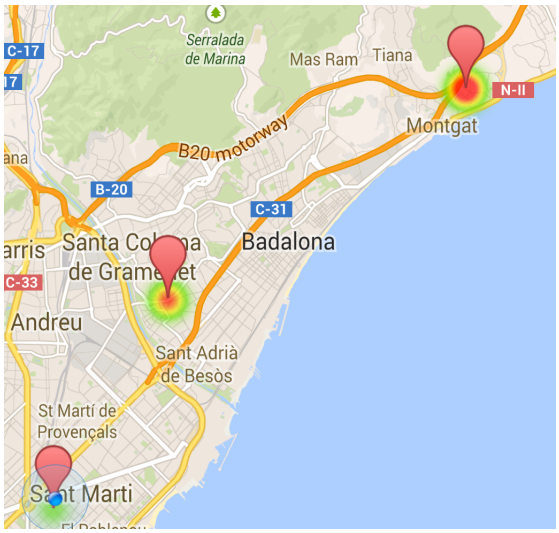


Figure 5.5: Testbed Map.

5.2.1 Results

Thanks to the testbed a sensor network has been created, with only three nodes, for economical reasons, but it has been shown that the network could grow without problems.

The data has been stored on a platform and the application has been able to access them, in this case a problem happened, the application was not designed to so much data, so it needs improvement.

A mobile application has been made, and shows an example that could led to more people to create other applications that work with the sensor data stored in opencities.

All the data created during this testbed has been stored in the SD card of the arduinos, and put it into graphics in the following figures: 5.6, 5.7, 5.8, 5.9, 5.10.

The locations had been chosen to change a little bit the values, for example, the node in Montgat had more relative humidity that the one in Santa Coloma. Or the one in the UPF had more noise that the others.

This graphics had been done with an octave script that analyze the logData file in every sensor node.

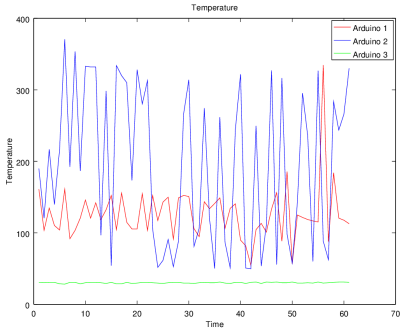


Figure 5.6: Graphic Temperature

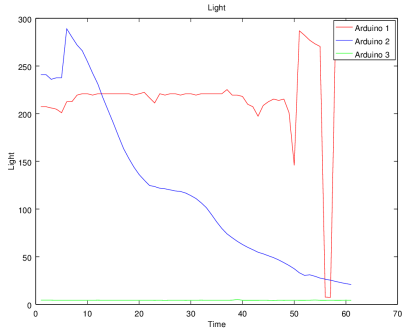


Figure 5.7: Graphic Light

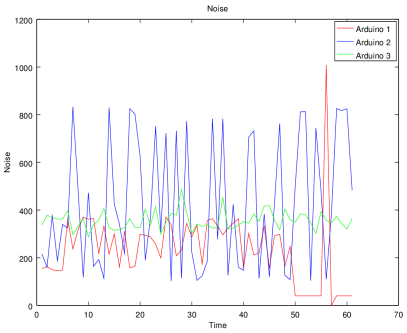


Figure 5.8: Graphic Noise

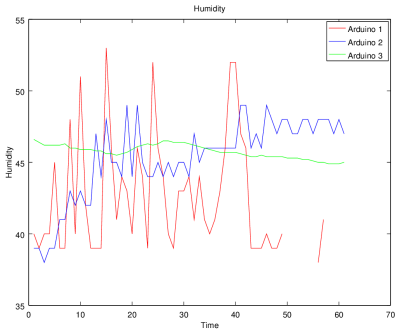


Figure 5.9: Graphic Humidity

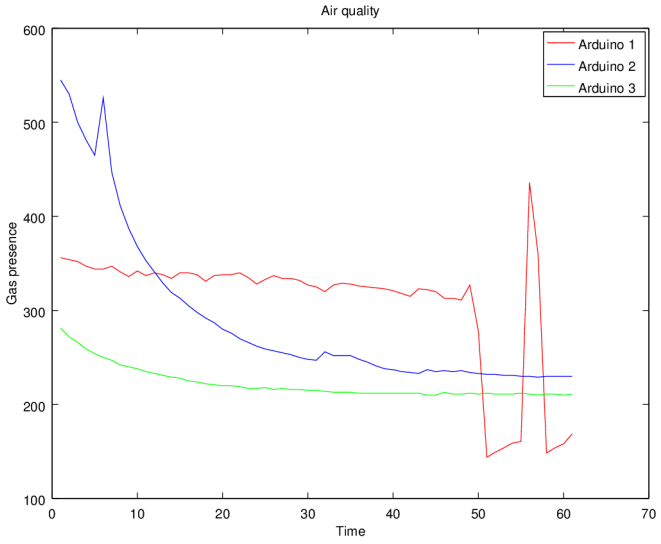


Figure 5.10: Graphic Air Quality.

Chapter 6

CONCLUSIONS

During this project a deployment of a sensor network had been made with optimal results, this has proved that the sensor nodes have properly.

Therefore, it has been shown that anyone can deploy its own network in an inexpensive way, with open source, and easy to use.

A mobile application has been developed to serve as an example, so the citizens who want to create their own find the process easier.

All the code that it has been created during this project is open source and is online, in a Github repository.

In conclusion, the project had satisfied the goals presented at the start, which are share sensor data on an open network, and let the users visualize it.

Chapter 7

FUTURE WORK

This project can be improved in two ways, the sensor node and the android application.

In the case of the sensor node it will be best if a prototype is build for the node to be in the outside, and, also to make the arduino Power over Ethernet.

On the other hand, the mobile application showed some issues, when the data is too big, the process takes a little bit, which could make the users stop using it. But there is also a line that could not be make in this project, and is to show how the data changes during a period of time.

Finally, there has not been any difusion of the project, in the future, the project could be difuse by a webpage or conference on sensors.

Bibliography

[Aosong Electronics Co., 2013] Aosong Electronics Co., L. (2013). Digital-output relative humidity & temperature sensor/module dht22. https://cdn.shopify.com/s/files/1/0045/8932/files/DHT22.pdf?100745. [Online; accessed 10-February-2014].

[Arduino, 2014] Arduino (2014). Arduino. http://arduino.cc/. [Online; accessed 8-January-2014].

[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.

[Emartee, 2013] Emartee (2013). Mini sound sensor - emartee.com. http://www.emartee.com/product/42148/Mini%20Sound%20Sensor%20%20Arduino%20Compatible. [Online; accessed 10-February-2014].

[Instruments, 2013] Instruments, T. (2013). Lm35. www.ti.com/lit/ds/symlink/lm35.pdf. [Online; accessed 8-January-2014].

[Opencities, 2014] Opencities (2014). Opencities. http://opendata.nets.upf.edu. [Online; accessed March-2014].

[santander, 2014] santander, S. (2014). Smart city in santander. http://www.smartsantander.eu/. [Online; accessed 10/02/14].

[smart city, 2014] smart city, A. (2014). Smart city in amsterdam. http://amsterdamsmartcity.com/. [Online; accessed 07/02/14].

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.

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

8.3 Class Diagram

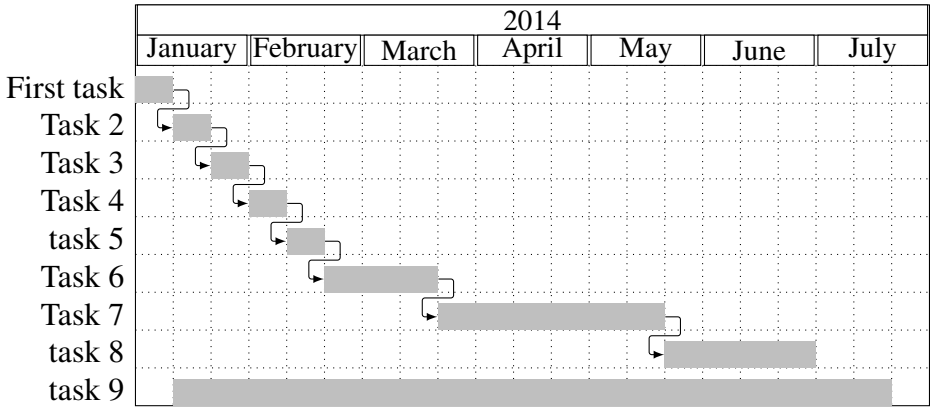


Figure 8.1: Gantt Chart

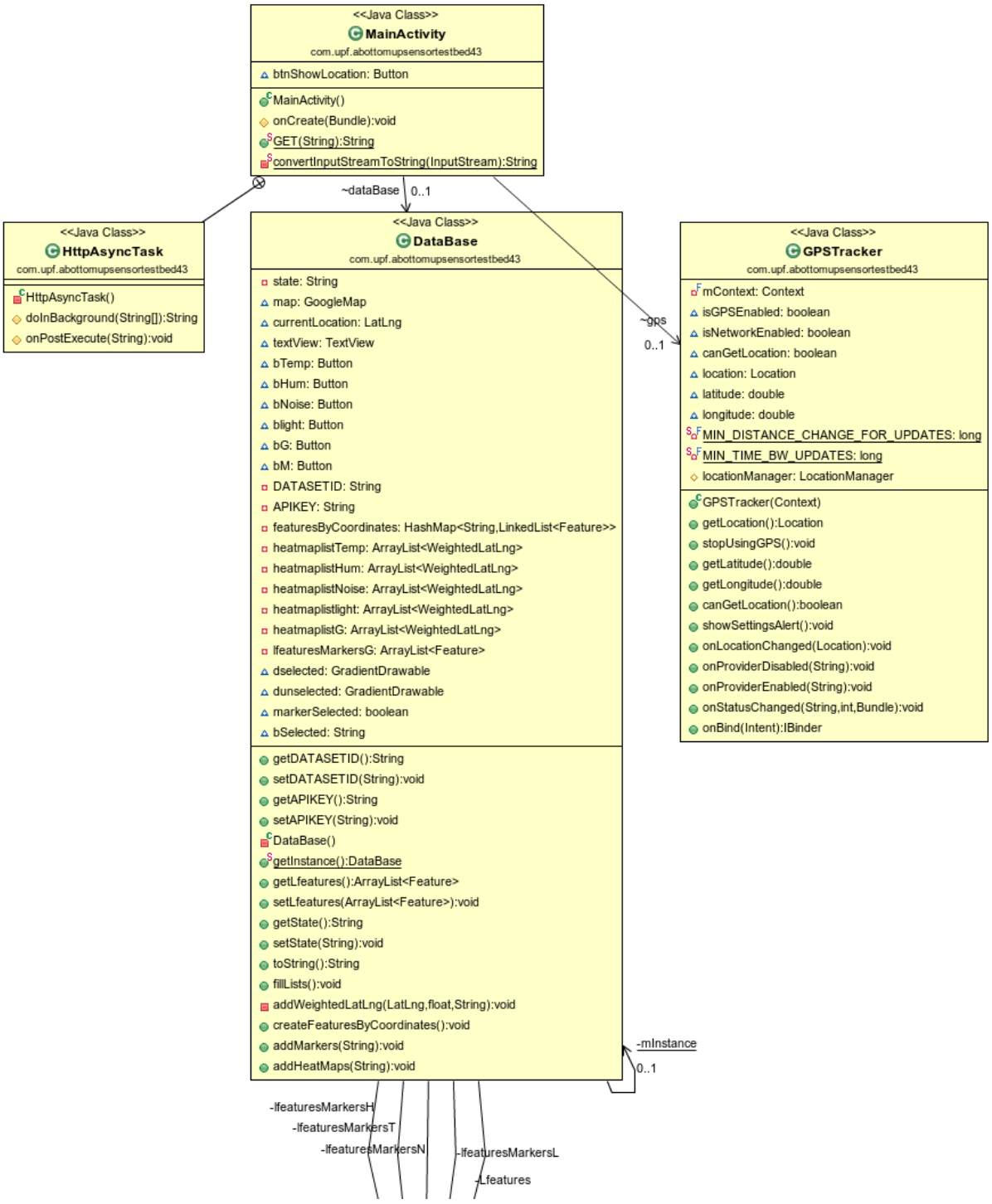


Figure 8.2: Class Diagram of the Android App part 1.

Figure 8.3: Class Diagram of the Android App part 2.