Universitat Pompeu Fabra

Bachelor Thesis

Open Sensor Network

Author:
Alejandro Andreu Isábal

Supervisor:
Dr. Jaume Barceló

A thesis submitted in fulfilment of the requirements
for the degree of Graduate in Telematic Engineering

in the

Research Group Name
Department or School Name

May 2013

Declaration of Authorship

I, Alejandro Andreu Isábal, declare that this thesis titled, 'Open Sensor Network' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

UNIVERSITAT POMPEU FABRA

Abstract

Escola Superior Politècnica
Department or School Name

Graduate in Telematic Engineering

Open Sensor Network

by Alejandro Andreu Isábal

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

Abbreviations

ACK	**ACK**nowledgement
ADC	**A**nalog to **D**igital **C**onverter
BuB	**B**ottom-**u**p **B**roadband
DIO	**D**igital **I**nput **O**utput
ICT	**I**nformation and **C**ommunications **T**echnology
ISM	**I**ndustrial **S**cientific and **M**edical
LAN	**L**ocal **A**rea **N**etwork
LoS	**L**ine **o**f **S**ight
PAN	**P**ersonal **A**rea **N**etwork
RF	**R**adio **F**requency

For/Dedicated to/To my...

Chapter 1

Introduction

During the last years we, the Internet users, just had one chance to know how things are managed, from top to bottom. That is, telecom operators reserve some resources (optic fiber, certain bandwidth, etc.) for each one of their clients and charge them for this service. In a top-down approach the consumer remains completely passive and has to solemnly accept what the telco dictates. Now, a new model pretends to turn this trend upside down.

This new way to do things is called *bottom-up broadband*, and is also how this project is posed. The very same users that were before passive will become very active, helping not only by designing the network but also deploying and maintaining it, thus participating in every step of the system lifecycle. Hence, without a central authority the usufructuaries are the only ones that conform this kind of networks.

Bottom-up broadband (BuB from now on) schemes have several important advantages over those that follow a conventional top-down approach, such as: easier and faster setup due to the lack of a central authority (as it happens in peer-to-peer networks), it can be adapted to anyone's needs since they are the caretakers of the system, and could also become the solution to those that live in an area that is not economically attractive to regular ISPs[].

As for disadvantages, a BuB network creation can be very time consuming, since users participate in every single step of this endeavor[].

Sensor networks are very important nowadays and its objective is to gather data.

> Data itself isn't good nor bad. Data just represents the surrounding reality. The more data we may access, the more accurate model we may create of the

reality, thereby also define our actions in ways that are maximally beneficial to our aims[].

But, does it make sense building such a sytem under a BuB model? The answer is yes, it does. As it happened with traditional telecom operators, the information that is obtained through sensor networks is kept by the agencies that own them without even making a public API to "play" with this data.

Therefore the main goal of this project is to design and deploy a sensor network that gathers real-time information and that enables developers to create applications that will ultimatelly help the citizenship improve their daily lives. Intrinsically this can be divided in more specific objectives, such as:

- Allow citizens, individuals belonging an organization or even enterprises to connect scattered sensor nodes.
- Collect different kinds of information and transmit it to the Internet.
- Samples (values of sensors) must be gathered ofen enough to be almost real-time.
- Use of open technologies to allow easier replication and modification as well as reducing final costs.
- The project shall become a tool so anyone that needs or wants to deploy a sensor network can do it as soon as possible.

Chapter 2 (2) takes a look at the state of the art. That is, why are sensor network important nowadays and what has been done until now regarding commercial and open solutions.

Chapter 3 (3) shows what technologies have been used to complete this endeavor. A brief description about each element is attached so the reader can replicate more easily the network and have some details at first glance.

Chapter 4 (4) focuses on the way this pilot has been completed. That is, the methodology that has been followed, as well as how problems have been confronted.

In Chapter 5 (5) we can see how is each node and the designed and also how programs and scripts work, mainly through flow diagrams.

Chapter 2

State Of The Art

Sensor networks, as many other technological advances see their origin on military research. They date back to the early 60's during the Cold War, when the United States deployed an underwater system to detect Soviet submarines called SOSUS (sound surveillance system). However, it is not until the beginnings of the 21st century that more applications were found beyond warfare. The main causes for that to happen is that the cost and the size of the components did decrease.

Also, new sets of wireless standars did see the light. IEEE 802.15 allows to create networks in a short radius called WPANs[1] with a low bitrate and incredibly extending battery lifetimes. On the other hand we have IEEE 802.11, which enables wireless communications to experiment similar bitrates to those obtained in a wired network.

One of the motivations to develop an open sensor network is that normally those who are the owners of the information keep it to themselves, a situation on which nothing is given back to society. Thus social, cultural and urban development gets narrowed[1].

Also, deploying sensor networks significantly contributes to the growth of smart cities ICT[2] structures which, at the same time, makes urban areas thrive[2].

There are already some initiatives that make use of wireless sensor networks. The word "initiative" is not intended to refer just to companies but also to organizations and individuals. At the time of writing, we can distinguish between two main kinds of sensor networks: company-driven and community-driven networks, depending on who shapes the system.

[1] Wireless personal area networks, defined in IEEE 802.15, refer to networks where devices are a few meters away from each other.

[2] Information and communications technology, expand.

These networks can generate a big amount of data creating the necessity of storing this information and making it always available for further usage. There are already some websites with the only objective of storing this information and providing beautiful visualization tools.

2.1 Company-driven sensor networks

These kind of systems work normally in an opaque or translucent way but with the advantage that they have very clear objectives. Also, they usually have more resources, as making money is the main motivation.

expandir

2.1.1 Libelium

This is one of the biggest companies in the world built around wireless sensor networks. It offers the mechanisms and tools to deploy and build systems around the Internet of Things, smart cities and M2M[3] communications.

FIGURE 2.1: Libelium logo.

The majority of the products they sell are focused on one specific application, such as waste management, structural health, etc. These are intended to be bought by system integrators for end users. However, they also offer the so-called "Waspmote", which is a sensor device for developers that can be freely customized and reprogrammed, since this is an open source product.

Their products are being widely used across more than 75 countries and they are definitely one of the leaders of the wireless sensor network industry.

2.2 Community-led sensor networks

Projects that are driven by the community give all the decision making power as well as resources to the community. The individuals that conform the

[3]Machine-to-machine communications are established between two devices.

community are very passionate about what they do and the workflow is highly transparent.

Community-driven projects give more control to the individual user. *expandir*

2.2.1 Air Quality Egg (AQE)

This is a sensor network that aims, as its own name indicates, to measure the air quality. This is through NO_2 and CO levels.

Each user is supposed to connect their egg to their local network via an Ethernet inferface. Then, a bunch of outdoor sensors are placed outside and communicate their readings to the base station (the egg) wirelessly through a radio frequency transmitter. Finally the data is sent in real time to Cosm[4], an open data portal that will be described in the next section.

It is worth mentioning that the AQE project is completely open, hence anyone can improve the platform as well as building his own egg from scratch without having to actually buy one. All the information related to the hardware, software and sensor calibration can be found in their wiki.

2.2.2 Smart Citizen

Although this is a very young project (still in beta stage) it intends to create the biggest community around social sensing. It was initially crowdfounded in 2012 through a Goteo[5] campaign and they are planning on going to Kickstarter soon.

The Smart Citizen platform allows its users to precisely geolocate their data and see other users' information. There is also a very big emphasis in data sociability, since every value or datastream[6] can be shared through any social networking site or even inside the same web application.

Openness is as well one of their main values, since every piece of code (including the website) is open source licensed.

[4] http://www.cosm.com

[5] Goteo is a Spanish social network that helps crowdfund open projects that result in a society improvement.

[6] Set of values that represent an individual sensor.

2.3 Open data services

All gathered information must be stored in some place, and this is where open data portals (also called Internet of Things clouds) come in play.

These websites provide users with an open API[7] so they can upload new values, create new feeds[8], retrieve the data end even create customized triggers, such as sending a push notification to a smartphone or even "tweeting" something. This way, we cannot only sense but *act* to certain kinds of events.

Because data by itself is usually worthless, one of their most important features are data visualization tools. They allow us to easily detect patterns and also even correlate certain factors.

Good examples of these sites are, as mentioned before, Cosm and Sen.se. Both are free to use and very easy to interact with (mainly through simple HTTP packets).

In case we want to host our own cloud for the Internet of Things, there is also a great solution called Nimbits. It is open source software and anyone can install it in its own server. It is very easy to fetch and push data to the platform, since it has a RESTful API.

[7] Explicar API?

[8] Representation of an environment. A feed can be a museum hall, where presence and noise levels are measured.

Chapter 3

Technologies

Three essential blocks form a sensor network. Namely sensors, processors and communication devices[3]. In the next sections all of them will be explained and in the next chapter I will show how are these related. There is even a case where a device (Digi XBee®) fulfills two of these roles.

3.1 Sensors

We now live in a world where we hear a lot the word "sensor", but what is exactly a sensor? *A sensor is a converter that measures a physical quantity and converts it into a signal which can be read by an observer or by an instrument.*[4]

This definition might seem (and is) quite simple, but complexity resides on calibration and coming up with actual useful applications. Since every sensor from the same family is equal in terms of design but different in reality (due to small random variations during the fabrication process) output has to be adjusted to agree with a given standard. When it comes to applications, RFID tags can be used to determine wether a book is on the right spot in a library or not, or with a very intense light beam we can detect how the blood flows through a vein thus succesfully sensing heart rate. These are just two imaginative uses for nowaday sensors.

Right now, only environmental factors have been measured since those have been tested over and over for the last years and they serve as a proof of concept for this network.

3.1.1 Aosong DHT22

The DHT22 is a low cost humidity and temperature sensor designed by Aosong Electronics, a Chinese corporation[1]. This is a digital sensor, which means that the output is represented in the form of bits, thus requiring some amount of computational power to "interpret" the results. The output format is precisely described in the datasheet of this product, and luckily there already are some library implementations to work with the DHT22. Then, this device will only work with platforms that allow digital input, such Arduino.

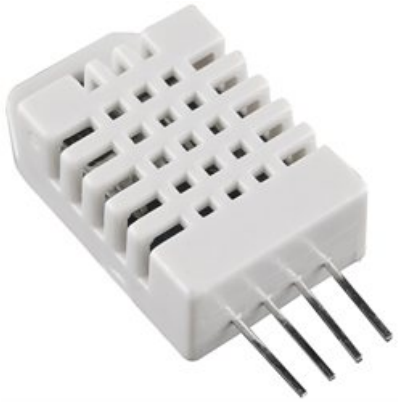

FIGURE 3.1: DHT22 humidity and temperature sensor.

3.1.2 Emartee Mini Sound Sensor

Manufactured by Emartee, can also be found by the name of "Emartee part number 4021", and can be used to measure noise levels among other uses. Essentially, it consists on a microphone with a built-in amplifier onto a breakout board, which can be useful to work directly with perfboards or breadboards (both construction bases for rapid prototyping of electronic circuits).

The output signal is analog and is increased by a factor that allows an Arduino or any device with analog I/O pins to detect it easily[5]. Its operating voltage is 5V.

3.1.3 Sharp GP2Y1010AU0F

This is an inexpensive optical dust sensor, used to measure air quality. It is made out of an infrarred emitting diode which, with a well positioned phototransistor can measure the reflected IR rays thus detecting dust levels

[1]Its datasheet can be found at: http://www.adafruit.com/datasheets/DHT22.pdf

in the air[6]. This device, which can be powered with up to 7V gives an output voltage (analog) proportional to dust density in the air. Some of its applications are air monitoring and air conditioning.

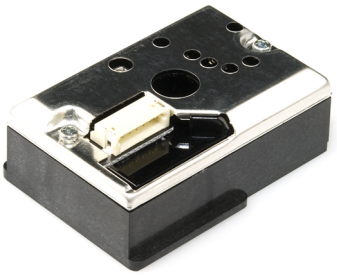

FIGURE 3.2: Sharp GP2Y1010AU0F optical dust sensor.

Surprisingly, this detector, which is priced at the time of writing about $12, gives very precise results, similar to those offered by an expensive laser particle counter.[7]

3.1.4 LM35

3.2 Digi XBee® Wireless RF Module

These radio modules are based on the IEEE 802.14.4 standard and provide an unexpensive, low power, low rate communication. They mainly use ISM[2] bands.

There are two versions of these modules, named "Series 1" and "Series 2". The older one (Series 1) implements the previously mentioned IEEE 802.15.4 standard which enables the network to follow point to point topologies. On the other hand however, the latter implements a standard specification called *ZigBee*. This protocol, although more complex has mesh networking capabilities which can be a key feature in some sensor networks.

Despite their size provide us with many interesting features, such as 128-bit encryption, over-the-air configuration and several pins that enables the XBee to read analog values as well as working with digital input and output[8].

This is why for the sake of this project "Series 2" has been chosen. It is worth saying that each of the two versions can transmit with different power levels

[2]Descripción de ISM.

FIGURE 3.3: Digi XBee® Wireless RF Module

thus varying the effective communication range[9]. More detailed information can be found in the table below.

Version	Power	Indoor range	LoS range[1]
Series 1	1mW	30m	100m
Series 1 PRO	63mW[2]	90m	1600m
Series 2	2mW	40m	120m
Series 2 PRO	63mW[2]	90m	1500m

TABLE 3.1: Comparison of different versions of XBee®.

Also, if extra range is needed (up to 40km in line of sight) there are also XBee® devices that transmit in lower ISM bands (900 and 868 MHz). However, when transmitting in these frequencies neither ZigBee nor IEEE 802.15.4 can be used. DigiMesh™ networking protocol is the only option and it is property of Digi International Inc.

3.3 Arduino

Arduino is the leading protopying platform nowadays. It is completely open source including the schematics of the hardware itself, which is a single-board microcontroller. Anyone can program the board through a programming language very similar to C/C++ and based on Wiring. To upload a sketch (a program) to the microcontroller they also have developed an Arduino IDE based on Processing.

[1]LoS range refers to line of sight range, where a straight line can be drawn from the transmitter to the receiver. In this situation, there are no obstacles between them and better bitrates and/or ranges can be achieved.

[2]This output power can be obtained using high gain antennas.

The amount of projects related to this platform is incredibly big, and it has gained huge popularity amongst designers, hackers, programmers and hobbyists these past years. It offers several advantages over similar devices, because it is really cheap, cross-platform and has every benefit inherent to the open source initiative. Also, like other open projects Arduino comes in many "flavours" depending on the characteristics of the project.

As it can be seen on the next picture, the board has many input/output pins that are compatible with analog and digital values. It's not just that but also it can establish a serial communication with a computer so interaction between programs and the platform can take place.

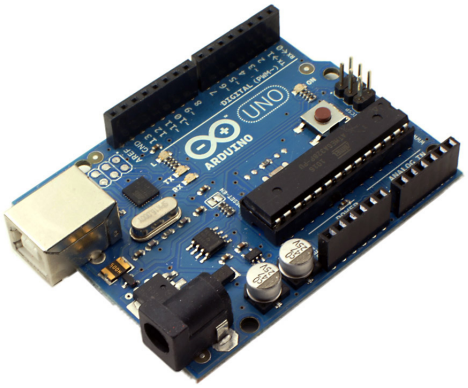

FIGURE 3.4: Arduino UNO prototyping platform.

Arduino UNO is the one "flavour" that has been chosen to perform this project, since it is cheap and also the most common. That means all shields[3] work by default on it and the community it has is the biggest. In particular, this model has the following features[10]:

3.4 Raspberry Pi

This device is an inexpensive GNU/Linux box that follows an ARM architecture that acts as the sink of the network. There are two models of this device: one that has 256MB of RAM (known as model A), and another one that has 512MB of such memory (model B). As for the processor it utilizes a 700MHz Broadcom SOC. The operating system is directly loaded from an SD.

[3] A shield is another board plugged on top of the Arduino to extend its functionalities.

Feature	Value
Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

TABLE 3.2: Characteristics of the Arduino UNO.

Currently it supports many popular distributions, such as: Arch GNU/Linux, Raspbian (a fork of Debian specifically designed to run on the Raspberry Pi), NetBSD, etc.

This is an optional part of the network since an XBee can be attached to any device that speaks serial. Nonetheless it is small and low-power, providing the necessary modularity for this kind of systems.

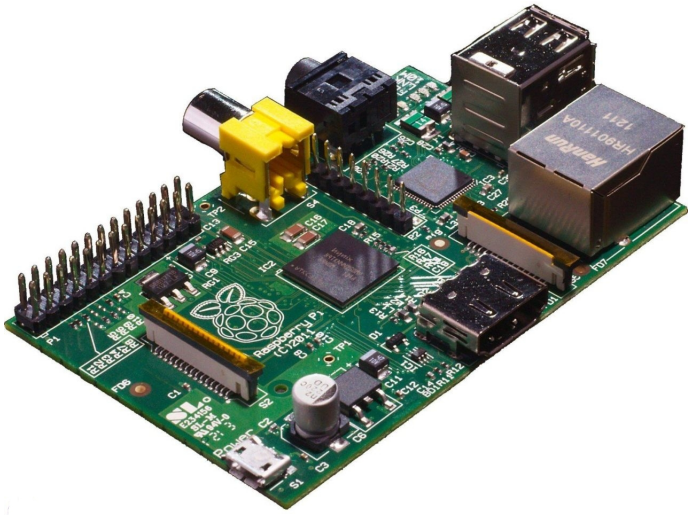

FIGURE 3.5: Raspberry Pi

3.5 D-Link DWA-123

The Raspberry Pi, as stated before, has an ethernet interface, but in case such option cannot be used, a Wi-Fi dongle is the best solution. This model

in particular is works out-of-the box with the Raspberry Pi and is cheap as well, supporting 802.11b/g/n.

3.6 Arch GNU/Linux ARM

Explicar.

Chapter 4

Methodology

The first step I took to complete this project was having a deep look at the state of the art. There are a lot of sensor network designs and some are as well open sourced, but the majority of them require either advanced knowledge on PCB fabrication or are focused on just one particular area (they aim to solve just one problem). Thus developing a system which is multipurpose and uses well-known technologies for rapid deployment are some of the key requirements that this network should meet (apart from the initial objectives).

Once all initial requirements are identified I had to choose one appropriate life cycle for the project. The pilot scope was not strictly constrained thus changes shall be handled in some way. Consequently, I chose an agile[] approach.

Agile management is a special case of iterative management, driven by changes[**?**]. Development of small modules is the usual thing, with deadlines every two or four weeks. Also, stakeholders are highly involved which is very related to the approach we followed all the components of BuB4EU. Each month there was a scheduled workshop where every participant informed the rest of the team about his/her last advancements. This method is very useful for getting constant feedback hence improving the overall quality of the project.

At the same time, the pilot followed an open development model, since it was (and still is) available on GitHub from the beginning. Open projects present several advantages over closed ones[], such as: ...

Chapter 5

Open Sensor Network

The result of working on this pilot is a set of tools (e.g. server script, schematics, Arduino program, etc.) that can be used to easily deploy a sensor network with well-tested and mature solutions that automatically uploads the information to the Internet. The two main features that define this network are:

- Ease of deployment — The server runs a general purpose Python script that doesn't care of the actual information it receives. Instead, it just uploads data to the selected webpages. If required, it can do some preossing in case the information that arrives is in a very raw form or it is mandatory to process it because of legal problems (such as photos, health care information, etc.).
- Flexibility — The system is prepared to transmit heterogeneous information. Each node is able to transmit different information and the sink will decode it anyway. This allows a community to gather what each individual wants or to achieve better granularity where needed. That is, someone might be interested in measuring temperature every two blocks and humidity every four blocks.

5.1 Network topology

Network topology refers to the way nodes that conform the system are arranged, thus it is clear that this factor will determine very important components about the network, such as reliability, modularity, fault occurence, etc. The use of Digi Xbee® RF modules enables the network to be configured in any kind of topology, from a simple ring to a complex mesh (networks where a packet can follow more than one path to reach its destination).

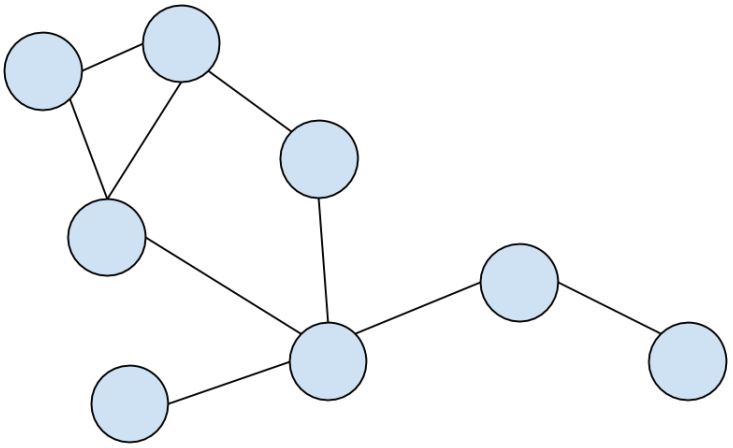

FIGURE 5.1: An example of a mesh network.

As these RF devices only allow one single sink per network, not making use of complex topologies would be an enormous drawback, since the range would then be limited to the sink's range. Luckily, ZigBee specification allows that some nodes act as a relay for other nodes, enabling us to build complex networks. Relay nodes will prolong the network lifetime if the system is battery-powered[11].

A sensor network of this kind will normally be a mesh network, whether it is fully connected (each node is connected to every other one) or partially connected. Mesh networks, as defined in the ZigBee specification have many advantages. They basically enable the system to be:

- Self-healing — Allows the network to operate when a node goes down.
- Self-routing — When a packet is transmitted or forwarded, the route that it follows is created (calculated) locally within every node.
- Self-forming — Nodes that are new to the network create links automatically with the rest of nodes and routes are created dinamically as well.

5.1.1 Device roles

Inside an ZigBee network there are three roles that a node can assume: coordinator, router and end device.

A *coordinator* is the sink of the network, and as stated before, only one is allowed per network. In this case, it will be connected to the Raspberry Pi so it can process and upload all the information to the Internet. A coordinator

stores vital information about the network and it works as the trust center (decides who may or may not join the network) if encryption is enabled[12].

A *router* has the same function has the previously mentioned relay nodes. It can generate and transmit data by itself to other router or a coordinator but it is also able to forward packets from other nodes. If the network is very redundant it should not be a problem having a battery-powered router.

Finally, an *end device* is the least capable device of all. It can only transmit information that will be or will not be forwarded, thus they are always on the edge of the network. Since no other node depends on an end device, they can make use of the *sleep mode*. This mode allos an XBee radio to wake up every certain amout of time thus saving a lot of power.

An example of such a network can be seen in the next figure.

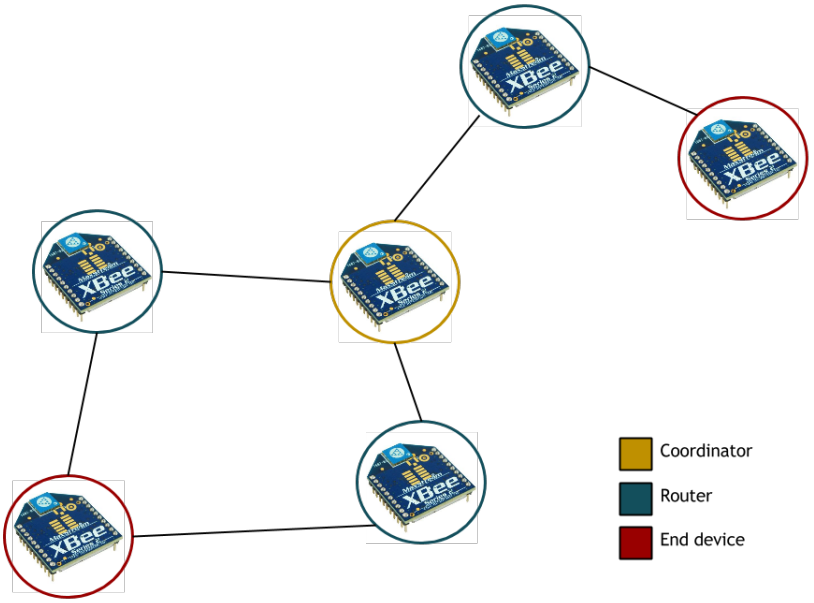

FIGURE 5.2: An example of an XBee network.

5.2 Sensor nodes

A sensor node is an element inside a wireless sensor network that is capable of cathering information, has some processing power and can relay information (if needed) to other nodes in the network[3][13].

In the design of this network two feasible scenarios have been considered, depending on which kind of sensors need to be used (whether they are digital or analogic), and if processing power is required.

In case at least one of those features is needed, an Arduino plus an XBee module are coupled together, with sensors attached to the microcontroller. Otherwise, a standalone XBee is used because it has a built-in ADC hence being able to directly read information from analogic sensors[5].

These two operational modes have been taken in consideration because despite one of them can equate the other's characteristics one can think of some applications where the features of an additional microcontroller are not needed. For instance, a sensor network monitoring temperature in an industrial environment just needs the so-mentioned analog value reading and a transceiver. Although there are two types of nodes, both can be used at the same time inside a given network.

To configure a sensor node one must use the X-CTU software, developed by Digi International. It is intended to be run on Windows, but it works fine on GNU/Linux plus Wine.

5.2.1 Standalone XBee

As mentioned in 3.2, any XBee device has two modes of operation, namely API and transparent.

In AT –transparent– mode, the device only relays serial data until it reaches the network coordinator. This implies this mode is simple and "universal", since a connection can be established with every device that understands serial. However, neither data reception or integrity are not assured (specially when working with the popular 2.4GHz ISM band, which is higly saturated) and eavesdropping can be a real problem since encryption cannot be used.

On the other hand, API mode is slightly more complex than transparent mode. Information is encapsulated in packets, feature that provides, among others, several advantages:

- When the coordinator receives a packet, it immediately transmits an ACK (acknowledgement) packet, indicating the transmission was successful. If such packet is not received then transmitting radio will retry sending it.
- Radios can be re-configured over the air.
- Checksum for data integrity.

- Encryption can be used, either using one pre-established key or getting one from the network coordinator.
- I/O samples, which enable us to use all the DIO pins that the module has.

This last feature is the one we will be abusing the most. In the next figure we can observe the pinout of an XBee radio, with all its DIO pins. For analog input, we can only use from `DIO0` (or `DIO0`) to `DIO3` (or `AD3`).

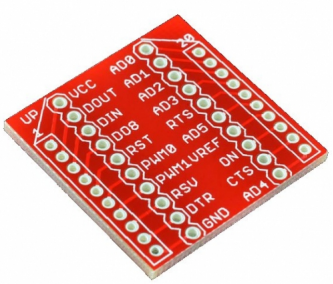

FIGURE 5.3: XBee pinout, seen from a breakout board.

An schematic of this node can be seen below, where only light level and temperature are measured. This particular setup can be useful to prevent fires in forests (although the optimal setup should have a humidity sensor as well).

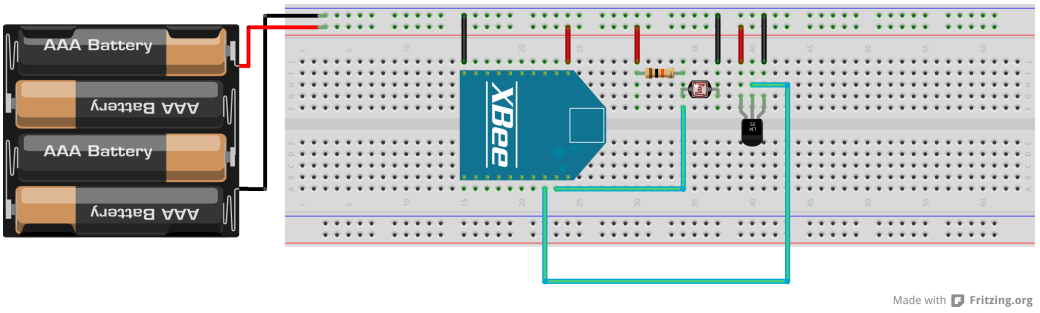

FIGURE 5.4: A standalone XBee node.

Using a battery and a solar panel to power a sensor node like this one, along with *sleep mode* can result in very long lifetimes.

5.2.1.1 Configuring a standalone XBee

To configure a standalone node one must connect it to X-CTU ensure that it is has the same `PANID` that the coordinator (that is, in the same network). Then:

Write down a description on how to configure a node

5.2.2 Arduino-based node

Arduino is capable of executing C code, thus being able to process any kind of information no matter how complex it is (always bearing in mind its hardware limits). To transmit all the information, the RF module attached to it will be always in API mode.

As a proof of concept, the example setup I have worked with has an analog sensor that measures sound levels (3.1.2), another one that measures air quality in terms of fine particles in the air (3.1.3) and a digital sensor that reads humidity and temperature (3.1.1). The elements that conform except the air quality sensor are arranged as follows:

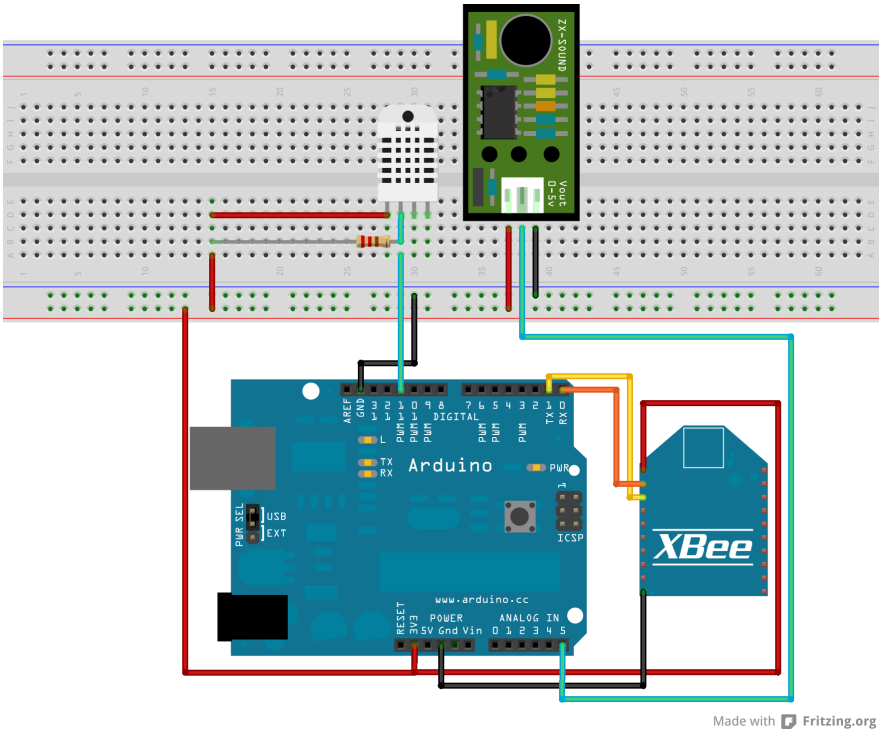

FIGURE 5.5: Sensor node based on Arduino.

This type of node has the same communication features as the standalone XBee. That is, encryption, acknowledgements, etc. In particular are better handled in this case since an Arduino can *react* to these ACKs.

5.2.2.1 Arduino program

The code can be browsed and downloaded via GitHub. There are two versions:

- Exact same code I used to conduct the experiments, so anyone can verify/test the obtained results.
- A skeleton file, very minimalistic and fully commented that can be extended as desired to build a sensor network from scratch.

Arduino syntax is based on Processing[1], thus it follows the same structure. There are two main steps: one called `setup()` and another one called `loop()`.

The `setup()` funcion initializes variables, modules (such as the XBee), libraries and sets pins in specific modes. After this function is successfully it is the turn of `loop()`, a function that, as its own name indicates, is executed over and over again. Thus this structure is where the actions takes place. Also, in order to interact with the XBee module I used the *xbee-arduino* library, a project originally written be Andrew Rapp and hosted on Google Code.

In the following figure (5.6) it is depicted how the program works in more detail:

The step called that involves metadata is because the decoder at the sink must know what data types are going to be sent; it can be two floats and a boolean or just a string. In order to decode these types, the first value that is sent it is always an integer (or an `int`). Its decimal numbers represent the kind of data that is going to be sent. To pack and unpack the values from the binary string, we use table 5.1.

So if for instance, a node wants to transmit two floats and a boolean it can transmit `199` as the first number or also `991`, depending on the actual order of the values. Nonetheless, it is recommended to transmit first the data types that are represented by the smallest numbers so the range of an integer is not exceeded. For more information on supported data types one can visit the reference on the official Arduino webpage[2].

[1] http://processing.org
[2] http://arduino.cc/en/Reference/HomePage

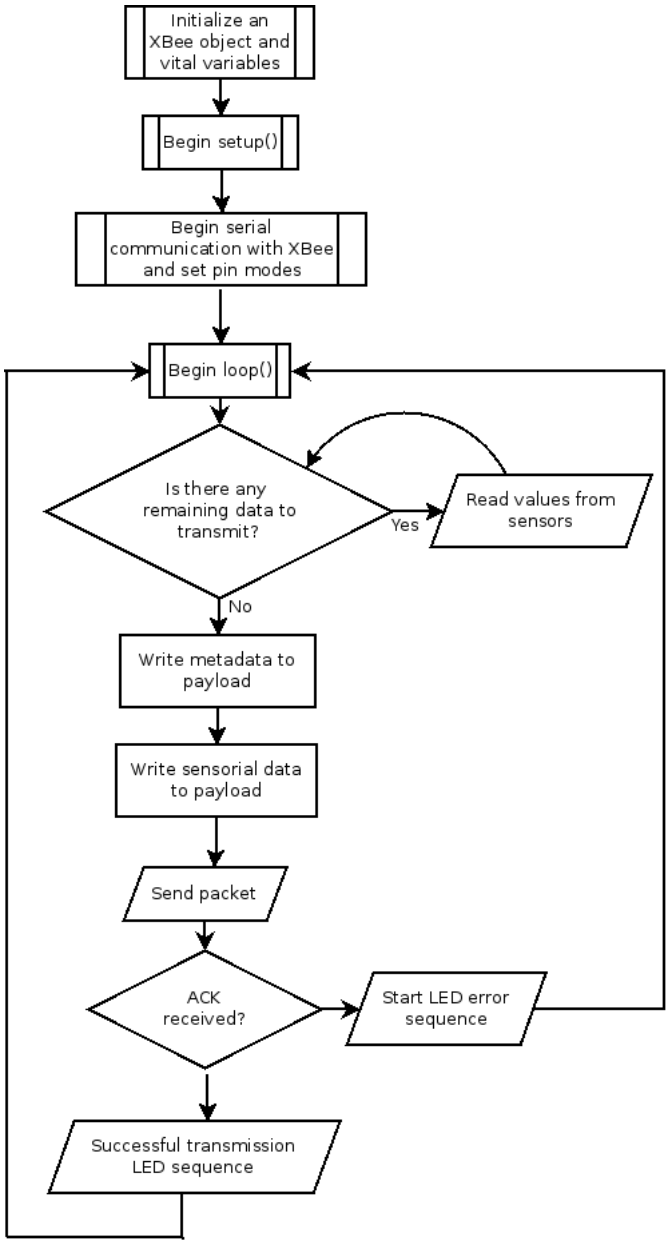

FIGURE 5.6: Flow diagram of the Arduino program.

5.3 Network sink

The sink is where all the information is headed. As stated in the previous section, it is composed of a Raspberry Pi with an XBee module connected to it. The schematic, although simple can be seen in the figure 5.7. Note that in the figure there is not an XBee but a breakout board for it that has a miniUSB interface, very useful for our purposes.

The GNU/Linux distribution that has been chosen to operate in

Number	Data type
1	`boolean`
2	`char`
3	`unsigned char`
4	`int`
5	`unsined int`
6	`long`
7	`unsigned long`
8	`short`
9	`float`
0	`double`

TABLE 5.1: Mapping between numbers and data types.

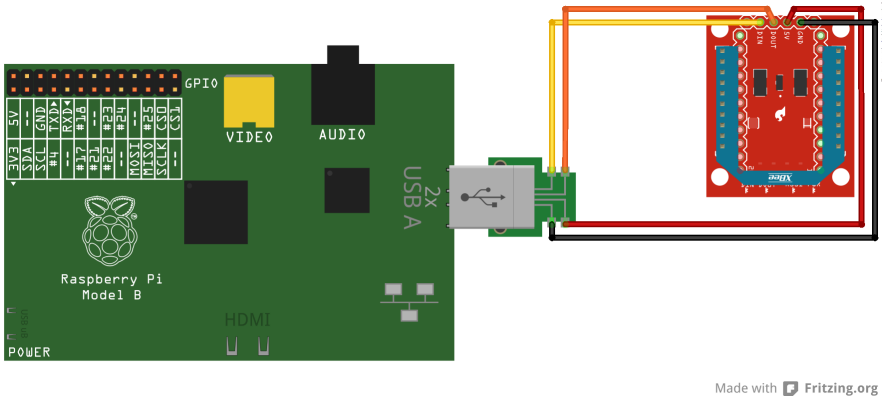

FIGURE 5.7: Schematic of the sink.

this device is Arch GNU/Linux, because of its KISS ("keep it simple, stupid") principle. Also, its repositories are huge because the whole community can contribute to them. Thus it has the *pyserial* library required to receive data from an XBee.

The script, called `server.py` is then run as a daemon which will be always receiving information. What it basically does is run an asynchronous XBee dispatcher, which will create a new thread for every new packet that arrives, thus enabling the sink to process many packets at the same time without blocking the whole script. The program is quite modular, since it allows to upload the information to the places the user wants by just uncommenting certain lines. A more detailed view on how it works can be seen in figure 5.8.

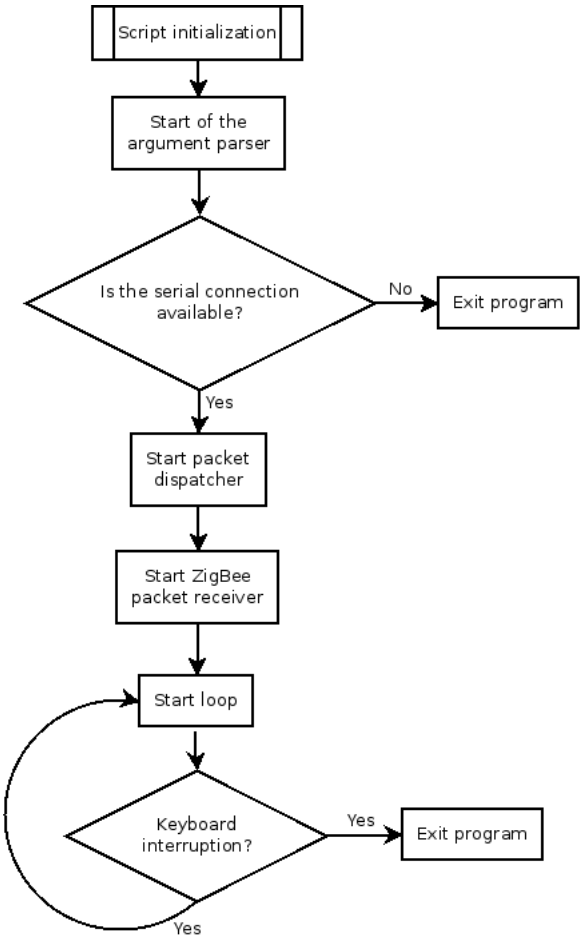

FIGURE 5.8: Flow diagram of the sink script.

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography

[1] Robert G Hollands. Will the real smart city please stand up? intelligent, progressive or entrepreneurial? *City*, 12(3): 303–320, 2008.

[2] Andrea Caragliu, Chiara Del Bo, Peter Nijkamp, et al. *Smart cities in Europe*. Vrije Universiteit, Faculty of Economics and Business Administration, 2009.

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

[4] Wikipedia. Sensor — wikipedia, the free encyclopedia, 2013. URL http://en.wikipedia.org/w/index.php?title=Sensor&oldid=548402365. [Online; accessed 3-April-2013].

[5] Emily Gertz and Patrick Di Justo. *Environmental Monitoring with Arduino: Building Simple Devices to Collect Data about the World Around Us*. Make, 2012.

[6] Sharp. Gp2y1010au0f compact optical dust sensor, 2006. URL http://sharp-world.com/products/device/lineup/data/pdf/datasheet/gp2y1010au_e.pdf. [Online; accessed 6-April-2013].

[7] Chris Nafis. Monitoring your air quality, 2012. URL http://www.howmuchsnow.com/arduino/airquality/. [Online; accessed 4-April-2013].

[8] Digi International. Xbee series 2 oem rf modules, 2007. URL ftp://ftp1.digi.com/support/documentation/90000866_A.pdf. [Online; accessed 6-April-2013].

[9] Robert Faludi. *Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing*. O'Reilly Media, Incorporated, 2010.

[10] Arduino. Arduino - arduinoboarduno, 2013. URL http://arduino.cc/en/Main/arduinoBoardUno. [Online; accessed 6-April-2013].

[11] Y Thomas Hou, Yi Shi, Hanif D Sherali, and Scott F Midkiff. On energy provisioning and relay node placement for wireless sensor networks. *Wireless Communications, IEEE Transactions on*, 4(5):2579–2590, 2005.

[12] David Gascon Alberto Bielsa. Wireless sensor networks research group, 2010. URL http://sensor-networks.org/index.php?page=1010510536. [Online; accessed 20-May-2013].

[13] Wikipedia. Sensor node — wikipedia, the free encyclopedia, 2013. URL http://en.wikipedia.org/w/index.php?title=Sensor_node&oldid=544833369. [Online; accessed 21-May-2013].