Maria Deseada Gutierrez Pascual

INDOOR LOCATION SYSTEMS BASED ON ZIGBEE NETWORKS

Bachelor's Thesis
Information Technology

April 2012

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	Date of the bachelor's thesis May 12, 2012
Author(s) Maria Deseada Gutierrez Pascual	**Degree programme and option** Information Technology Telecommunications

Name of the bachelor's thesis

Indoor Location Systems (ILS) based on ZigBee Networks

Abstract

In the last few years, Indoor Location Systems have been a major investigation branch in the Research and Development area. They got an special importance after the popularization of GPS which allows Outdoor Location with an accurate precision.

However, GPS is not operative inside buildings where mostly human activities occur, as there is no direct line-of-sight between the antenna and satellites, the signals suffer from reflections, scattering and diffractions.

The aim of this thesis is to implement an Indoor Location System able to locate a person inside a room, flat or building. Moreover, the ILS must be a low cost and low consumption system so that it can be a considered market option. It also should be open-hardware and open-source and easy to install what will make the project to be spreaded and improved.

For this goal, the project has two different branches of research: low consumption and low cost hardware.

This ILS has been implemented using RSSI information. Basing it only on RSSI values from one slave will give too much incertainty, due to reflections and diffractions. In this project we have improved the precision using trilateration with three slaves. Moreover, accurating the RSSI measurements from each slave have been done taking multiple samples in consecutive moments and using statistics models to processing them.

In addition, fingerprinting has been implemented with a database where some measurements will be added, so that errors made by attenuations or wall absorption can be compensated.

Subject headings, (keywords)

Indoor Location System, Indoor Positioning System, ZigBee, Pinguino, microcontrollers, Python, fingerprinting, trilateration, triangulation, RSSI, time of flight.

Pages	**Language**	**URN**
45	English	

Remarks, notes on appendices

The bachelor's thesis is published under GPL license.

Mikkeli University of Applied Sciences.

Tutor	**Employer of the bachelor's thesis**
Osmo Ojamies	

Contents

List of Figures

List of Tables

1. INTRODUCTION

In the last few years, Indoor Location Systems have been a major investigation branch in the Research and Development area. They got an special importance after the popularization of GPS, caused by a good reduction in the cost of devices needed for its implementation, which allows Outdoor Location with an accurate precision and it is used nowadays in most mobile applications and tracking systems. However, GPS is not operative inside buildings where mostly human activities occur, as there is no direct line-of-sight between the antenna and satellites, the signals suffer from reflections, scattering and diffractions.

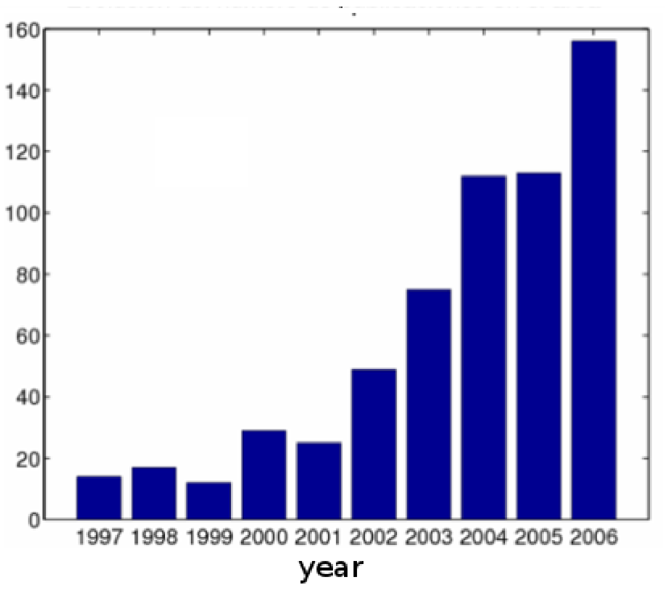


Figure 1: Increase of number of publications in the ILS area. Source [5]

This problem made engineers research other possibilities based on different Wireless networks, because society have realized the comfort and advantages of non wired devices

- PAN (Personal Area Networks): Bluetooth, wireless mouses and keyboards.
- LAN (Local Area Networks: WiFi.
- Mobile systems: GPRS, 4th generation mobiles (all IP).

Indoor Location Systems (ILS) also called by Local Positioning Systems (LPS) are location systems created to work in local and indoor environments. These systems are presented in different investigation branches, based on WiFi or Bluetooth. However, none of them has been as successful as expected and it is due to its extreme cost or its short signal reach.

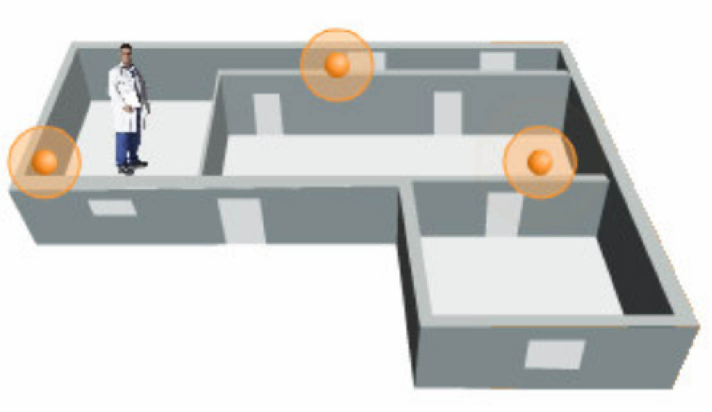


Figure 2: Indoor Location System. Source [5]

Lots of applications are appearing with the development of this systems:

- Rescue and emergency systems: elderlies living alone, i.e.
- Shopping trolley tracking for marketing purposes.
- Human interaction with home automation. For example, lights getting on while the user enters one room.
- Robots guiding.
- Automatic wheelchairs.

The aim of this thesis is to implement a Indoor Location System able to locate a person inside a room, flat or building and tracking him/her. Moreover, the ILS must be a low cost and low consumption system so that it can be a considered market option. It also should be open-hardware and open-source and easy to install what will make the project to be spreaded and improved.
For this goal, the project will have different branchs of research: low consumption and low cost hardware.

2. UNDERSTANDING ZigBee

2.1. Definition and Characteristics

2.1.1. Definition

ZigBee is an specification of a joint of high level wireless communication protocols based on the wireless personal area network (PAN) standard IEEE 802.15.4. Its goal is the applications that require reliable communications, due to mesh topology, with low data transmission rate and long live batteries.

ZigBee can be used in several types of applications as automation and security control, control of end devices as mouse or keyboards, remote control of electronic devices, monitoring pacients or elderlies but the main and most successful is home automation.

Figure 3: ZigBee main applications. Source [6]

The characteristics that make it so suitable for these purposes are:

- Low-cost.

  ZigBee devices are cheap as they do not need a high data rate and the microprocessor required for ZigBee devices is quite simple, due to the small size of the ZigBee MTU (Maximum Transmission Unit).

- Mesh topology.

  Provides a higher reliability because multiple transmission paths exist. This allow some nodes of the network to be asleep while others take the control of the propagation and avoid a whole network to block if ones node gets down.

- Low power consumption.

  As multiple nodes can be asleep until they receive some information, they do not consume too much power and the batteries can live even for 5 years.

2.1.2. Main Characteristics

- It operates in the industrial, scientific and medical (ISM) bands: it can operate globally in the 2.4 GHz frequency, but also in 868 MHz (Europe) and 915 MHz (USA).
- Its data rate is 250 kbps at 2.4 GHz, 20 kbps at 868 MHz and 40 kbps at 915 MHz.
- Its reach range is from 10 m to 1000 m.
- It operates over 16 channels in 2.4 GHz and over 11 channels in 868 and 915 MHz.
- A ZigBee network can have a maximum of 255 nodes, which mostly of time are asleep.

2.2. Architecture

The IEEE 802.15.4 standard defines an architecure based on the OSI reference model, as it can seen in the next figure.

This architecture defines the physical layer and the data link layer, divided in a Medium Access Control sublayer (MAC) and a Logical Link Control sublayer (LLC).

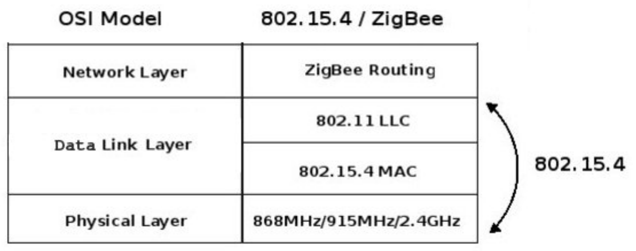


Figure 4: IEEE 802.25.4 architecture

2.2.1. Physical Layer

The IEEE 802.15.4 offers two options of physical layer depending on the using frequency. Both are based in direct sequence spread spectrum (DSSS) methods easy to implement digitally and also share the packets structure. The physical layer at 2.4 GHz allows a higher data rate due to a bigger order in the modulation. As the 2.4 GHz frequency is used globally, we are going to focus the study of the physical layer on it. The physical layer at this frequency has 16 channels between 2.4 and 2.4835GHz with a wide space between channels (5 MHz) facilitating the filtering in the transmission and reception. As this frequency band is free and used by most domestic devices, the physical layer has some functions, as received energy detection or link quality indicators that are used to choose a channel or change it in case of interference.

Physical	Band	Binary Rate	Modulation
868/915 MHz PHY	868.0-868.6 MHz	20 kbps	BPSK
868/915 MHz PHY	902.0-928 MHz	40 kbps	BPSK
2.4 GHz PHY	2.4-2.4835 GHz	250 kbps	16-ary orthogonal

Table 1: Physical layers characteristics

The physical layer at 2.4 GHz uses a semi-orthogonal modulation technique. The binary data are joint in symbols of 4 bits and each symbol corresponds to one of the 16 sequences of the code. The sequences are concatenated until they are succeeded symbols and the sequence is modulated using MSK (Minimum ShiftKeying).

2.2.2. Data Link Layer

The IEEE 802 divides the data link layer in two sublayers: Medium Access Control sublayer and Logical Link Control sublayer. LLC sublayer is common to all 802 standards while MAC sublayer depends on the hardware and changes according to the physical layer.

In ZigBee the medium is shared by all the users, so a MAC protocol is able to control the transmission through the channel and avoid possibles collisions. Moreover, in wireless networks it is not possible to listen and transmit at the same time. Therefore, ZigBee uses CSMA-CA (Carrier Sense Multiple Access Collision Avoidance). Each device, before transmiting senses the channel to see if it is idle, if the device finds that the channel is been used for another node, it waits for a random backoff time and after that time check again the medium. If now the channel is free and RTS/CTS Exchange is not implemented, the device transmits its data.
This protocol has a problem called ḧidden terminal problem ·̈ It occurs when a device is so far from another one that they are not able to sense each other producing a collision if both transmit at the same time. For avoiding this problem there is the RTS/CTS Exchange where the device, before transmitting, requests for the medium (RTS) and if it receives an acknowledge called CTS, then it can proceed to transmit avoiding collisions.

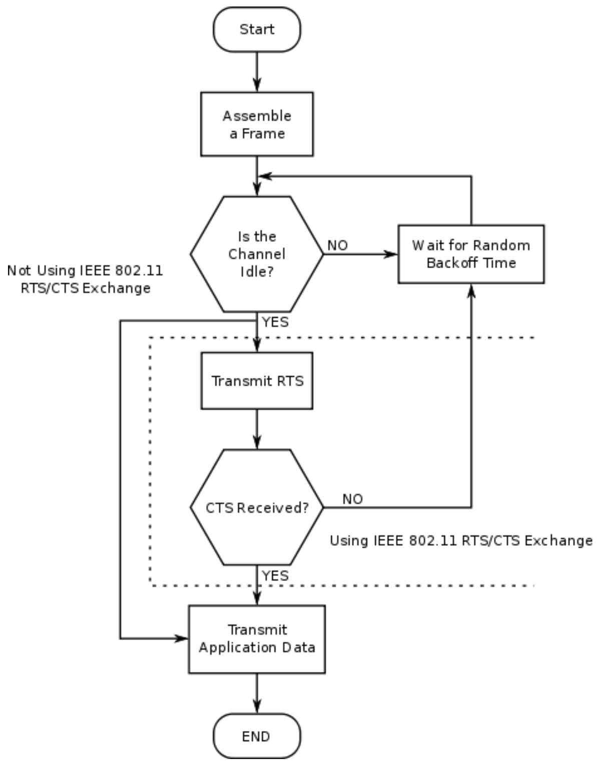

Start
Assemble a Frame
Is the Channel Idle?
NO
Wait for Random Backoff Time
Not Using IEEE 802.11 RTS/CTS Exchange
YES
Transmit RTS
CTS Received?
NO
Using IEEE 802.11 RTS/CTS Exchange
YES
Transmit Application Data
END

Figure 5: CSMA/CA diagram. Source [7]

2.2.3. Network Layer

Devices

Three different ZigBee devices can be defined according to its role in the network:

ZigBee Coordinator, ZC

It is the most complete device. It must exist one in each network and its functions are controlling the network and the paths that the dispositives have to follow to connect between them.

ZigBee Router, ZR

It interconnects devices separated in the network topology. Moreover, it offers an application level for the execution of user code.

Zigbee End Device, ZED

It has the necessary functionality to communicate with its parent node (ZC or ZR) but it can not transmit information to other devices. Therefore, this kind of nodes are sleeping mostly of time, keeping their batteries longer. Moreover, they do not need high requirements of memory what makes them cheaper.

According to their functionality, they can be devided as:

Full Functionality Devices (FFD)

Also known as active nodes. They are able to receive messages in 802.15.4 format. Thanks to the additional memory and computational capacity, it can work as Coordinator or ZigBee Router.

Reduced Functionality Devices (RFD)

Also known as pasive nodes. They have limited capacity and functionality with the aim of getting a low cost and high simplicity. Basically, they are sensors of the network.

Topologies

ZigBee allows three different topologies:

- Star topology: the Coordinator is in the middle and all the rest of devices are connected to it. If the coordinator blocks, no node is able to continue communicating with another node.
- Tree topology: the Coordinator is the root of the tree. If the coordinator blocks, the network will be divided in two subreds, but still some nodes can communicate with others.
- Mesh topology: at least one node has more than two connections. If one node blocks, the communication between the rest of the nodes can still happen due to the existence of multiple paths. The coordinator is the responsible of the management of paths.

In the next figure, we can visualize the different components in a ZigBee network.

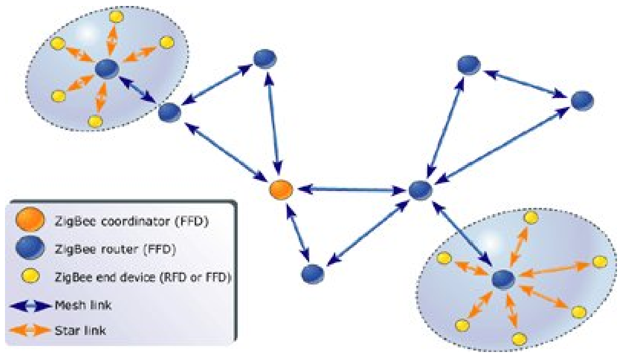
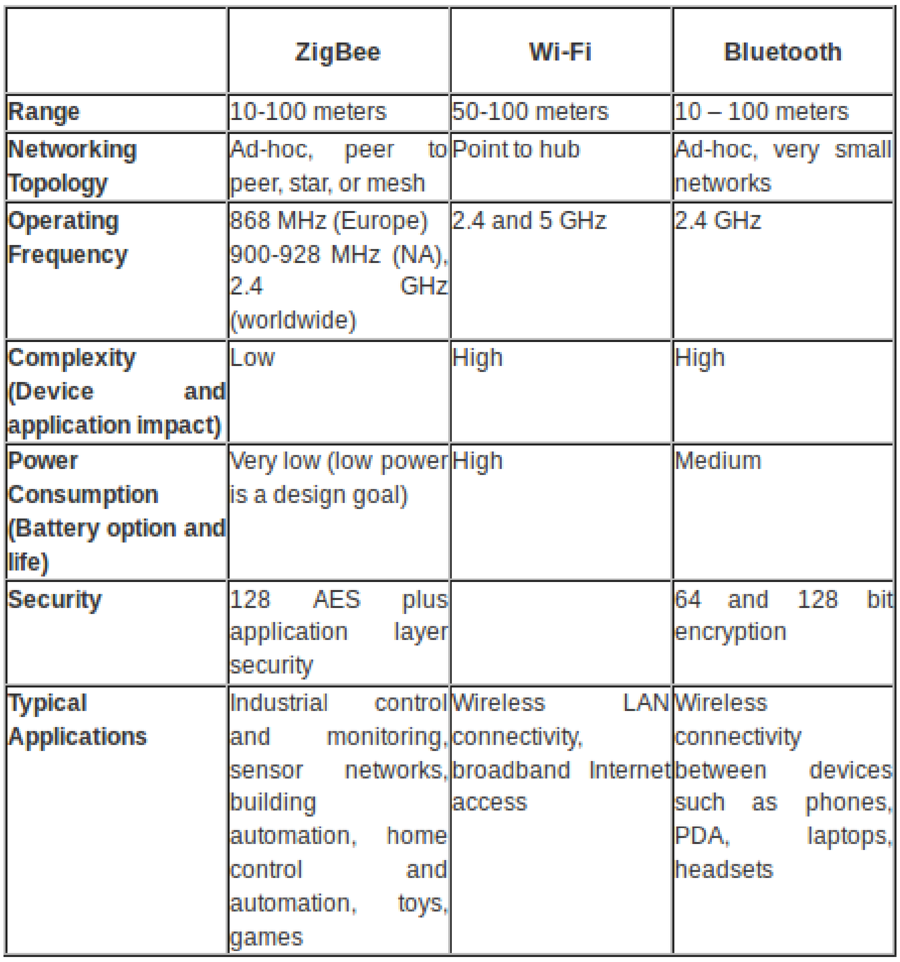


Figure 6: ZigBee networks: topologies and devices. Source [8]

2.3. ZigBee and other RF technologies

The technologies to be considered for this Home Area Networks (HAN) would be ZigBee (based on IEEE 802.15.4), Bluetooth (based on IEEE 802.15.1) and WiFi (based on IEEE 802.11). Other RF technologies are available but only these three have the technical maturity for this kind of systems and low cost required for consumer products. In the next figure, a table with the main characteristics of each technology can be seen.



	ZigBee	Wi-Fi	Bluetooth
Range	10-100 meters	50-100 meters	10 – 100 meters
Networking Topology	Ad-hoc, peer to peer, star, or mesh	Point to hub	Ad-hoc, very small networks
Operating Frequency	868 MHz (Europe) 900-928 MHz (NA), 2.4 GHz (worldwide)	2.4 and 5 GHz	2.4 GHz
Complexity (Device and application impact)	Low	High	High
Power Consumption (Battery option and life)	Very low (low power is a design goal)	High	Medium
Security	128 AES plus application layer security		64 and 128 bit encryption
Typical Applications	Industrial control and monitoring, sensor networks, building automation, home control and automation, toys, games	Wireless LAN connectivity, broadband Internet access	Wireless connectivity between devices such as phones, PDA, laptops, headsets

Figure 7: Main comparison: Bluetooth, ZigBee and WiFi. Source [9]

2.3.1. ZigBee vs. Bluetooth

Bluetooth is a standard and a communications protocol based on IEEE 802.15.1. Examining these two technologies, both are PAN protocols but they have different ranges and applications field. A Zigbee network can have until 255 nodes in one network, while Bluetooth is only able of having 8 nodes. However, Bluetooth has a higher data rate of 1 Mbps. Bluetooth has been developed for very short- range and is more suitable for applications with a middle data transmitting rate and non-stop services as file transfering and sound transmission.

ZigBee has been implemented in applications of low data transmission rate. End devices are not transmitting or reciving all the time what saves a lot of battery. For example, if a sensor transmits once each minute to inform about its state, a

Bluetooth device battery will last 100 days and ZigBee devices battery will last 9.8 years. As it can seen, the high data rate in Bluetooth is not needed for this kind of systems and the power saving with ZigBee is really significant.

2.3.2. ZigBee vs. WiFi

WiFi is a trademark of WiFi Alliance based on the IEEE 802.11 standard.
Indoor Location System does not require a big data rate. Order of tens of Kbps is enough and both technologies can accomplish it. Moreover, both provide similar power output sufficient for giving a good range for the system.
Going deeper on this comparison, some facts can be found:

- Star Network Topology vS. Mesh Technology.
  ZigBee usually is based in mesh networks, so it uses mesh technology what provides a higher reliability because multiple transmission paths exist. This allow some nodes of the network to be asleep while others take the control of the propagation and avoid a whole network to block if ones node gets down. WiFi uses the star topology (as we have an access point or router and all the end devices come from it). Its main disadvantage is that, if there is no redundancy, a significant part of the network can be blocked if one network device get collapsed. In addition, WiFi devices have a beacon interval each 100 milliseconds, so these devices can not get asleep permanently.

- Cost of devices.
  ZigBee devices are always cheaper than WiFi ones, at least a 20% and moreover, the microprocessor required for ZigBee devices can be less complex compared to Wi-Fi, what in massive production means savings of millions.

- Power consumption.
  Apart from taking into account the cost and performance aspects, power consumption is a major factor to be considered. General Electric (GE) published a research study [10] comparing the energy efficiency between this both technologies and their application in home networks (HAN). According to this publication, ZigBee is more efficient and therefore, it is the most

suitable technology and really well adapted for sensor networks. These kind of networks should limite consume peaks, allow the integration of renewable energy and reduce the consume of each device that compone these smart networks. In the next figure, it can be seen how the difference between consumptions is significant.

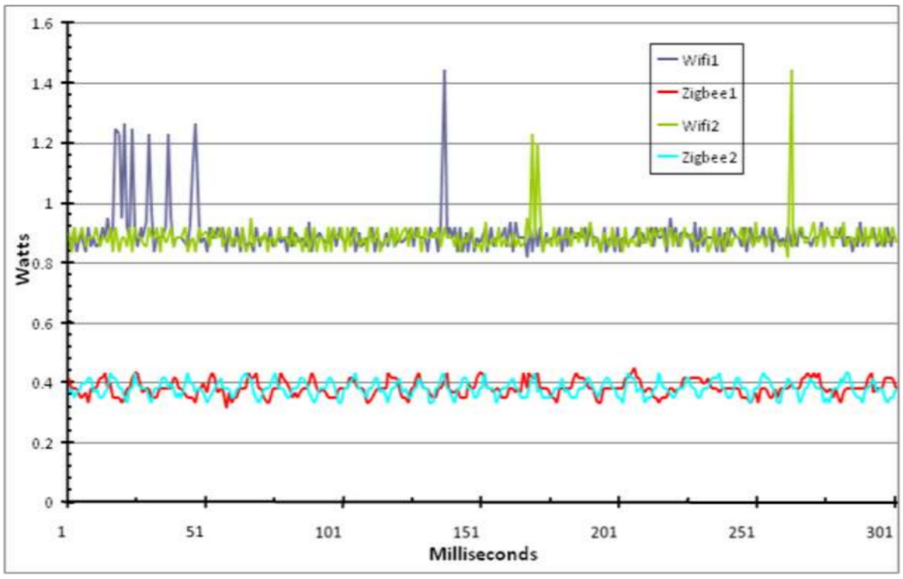


Figure 8: Consumption between ZigBee and WiFi under same circumstances. Source [10]

3. MICROCONTROLLERS RESEARCH

There are a lot of microcontrollers and platforms that could be used for this project. However, we are going to focus our research in Arduino and Pinguino because they present some advantages over the rest of microcontrollers in the market.

- Affordable. These boards are cheaper compared to other microcontrollers platforms. As an example, the most expensive Arduino's board cost 60€.
- Multi-platform. Arduino's and Pinguino's software works in all operating systems: Windows, Macintosh OSX and Linux while most of other microcontrollers have its IDE limited to Windows.
- Easy development environment. The development environment for these two microcontrollers is easy to use for beginners and it is flexible enough for advanced users. There are a lot of libraries just prepared to be used in a pleasant way.
- Open-source. Arduino's and Pinguino's software are published under a open license and they are prepared for being expandable by experienced programmers.
- Open-hardware. The layouts of these microcontrollers are published under a open licence, this means that circuit designers can do their own version of the module, exanding it or improving it. Even some users can build their own board according to the schemes and saving some money.

All these advantages make this two microcontrollers to be easy to program and to get.

3.1. Arduino

Arduino is an open-hardware platform ready to be used and to create prototypes based on easy to use and flexible software and hardware. Arduino can get information from the environment through its pines from a lot of sensors and it can control lights, motors and another devices. It is programmed by Arduino programming language (based in Wiring) and the Arduino IDE (based in Processing).

Arduino boards can be done by yourself (using the CAD designs in the official webpage) or they can be bought and the software can be free downloaded from its webpage: http://arduino.cc/en/Main/Software.
There are a lot of different versions of Arduino boards. The basic board, Duemilanove uses usa Atmel ATmega328. Some other older boards, as Diecimila and the first Duemilanove used Atmel ATmega168, while other boards use ATmega8 (datasheet). Mega Arduino uses ATmega1280.

3.1.1. Arduino Boards

There are a lot of Arduino Boards. The most used and known are the following:

- **Arduino UNO**. It is the latest version of Arduino USB basic board. It connects to the PC using a USB and it can be expandable with a big variety of shields with specific functionalities.It is similar to Duemilanove but has a different USB-to-serial chip the ATMega8U2.
- **Duemilanove**. It is the previous version of Arduino USB basic board. It automatically selects the power supply (from the USB or external source), eliminating the power selection jumper in previous versions.
- **Diecimila**. Its main difference is that it can be reseted from the computer without pressing the physical button in the board. Besides, uses a low voltage regulator which reduces the power consumption.
- **Nano**. It is a compact board designed for using directly in development boards. It can be connected to the PC using a mini-B USB.
- **Mega**. It is the biggest and most powerful Arduino board. Compatible with Duemilanove and Diecimila.
- **LilyPad**. It was designed for applications over clothes. It can be sewn to clothes and it has a stylish design.
- **Fio**. Designed for wireless applications. It includes one shield for XBee and a connector for LiPo batteries.

- **Bluetooth**. It has a bluetooth module that allows to communicate and program without wires.

- **Mini**. It is the smallest Arduino board. It is suitable for projects where the space is limited.

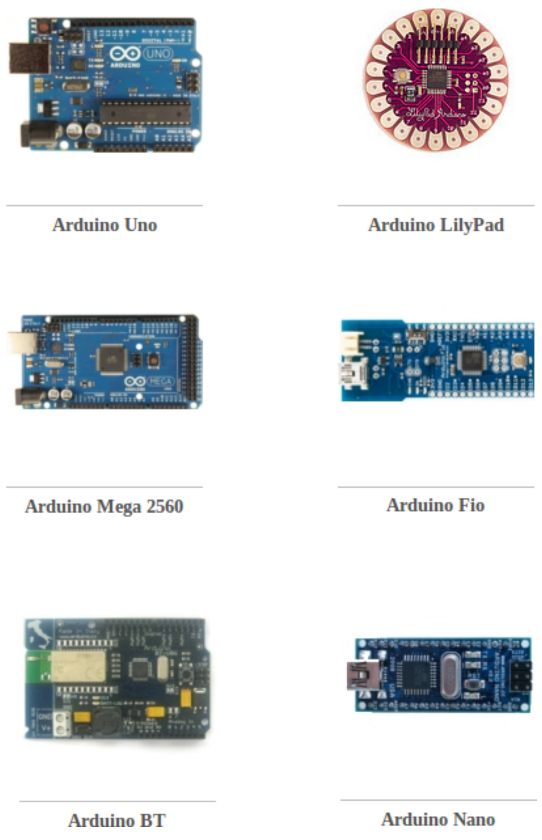


Figure 9: Arduino boards. Source [12]

3.1.2. Shields

Shields are boards that are put on the Arduino boards and expand a new functionality to be controlled with Arduino.

- Xbee shield. It allows to connect through ZigBee several Arduinos with a distance of 100 m using the module Maxstream Xbee Zigbee.

- Motor shield. It allows Arduino to control DC motors, servos and read encoders.

- Ethernet shield. It allows an Arduino to connect to an Ethernet network and have access to Internet.
- Proto shield. Provides space for building circuits.

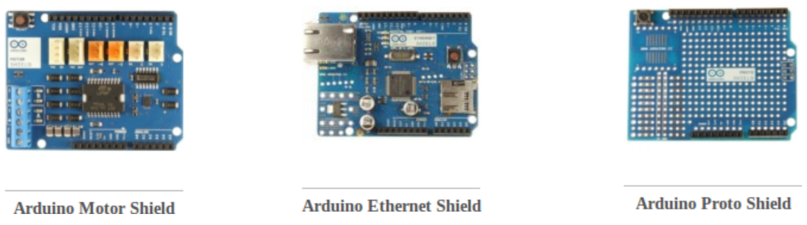


Figure 10: Some Arduino shields. Source [12]

3.2. Pinguino

Pinguino is an Arduino-like board based on Microchip 8-bit or 32-bit PIC micro-controllers. It is a platform open-source and open-hardware and it has also its own IDE.
The IDE of Pinguino is built with Python and available for all operating systems. An integrated preprocessor translates specific Arduino instructions directly into C. This preprocessor reduces the code length and the execution speed.

```
Welcome to Pinguino IDE x.2
```

Figure 11: Pinguino IDE. Source [13]

As Arduino boards, Pinguino can be done by yourself (using the CAD designs in the official webpage) or they can be bought and the software can be free downloaded from its webpage: http://www.pinguino.cc/download.php Finally, compatibility between Arduino and Pinguino has been tested and it is about 90% with some libraries still under development.

3.2.1. Pinguino boards

There are several Pinguino Boards. The most used and known are the following:

- **Kidule Pinguino**. This version is built with a Microchip PIC18F2550 chip. It works with a 20 MHz crystal and is USB 2.0 compatible. This board can be powered by the USB connector but also from an external power supply.
- **PICUNO-EQUO**. It is based in the Microchip 8-bits PIC18F4550 chip. It is similar to Arduino Uno format, compatible with Arduino Shields and it has an USB-B connector.
- **EMPEROR 795**. It is based on the Microchip 32-bits PIC32MX795F512L microcontroller programmed by USB. It has an 8 MHz crystal and voltage regulator.
- **PINGUINO-OTG**. It has a Microchip 32-bits PIC32MX440F256H and USB OTG. This is the board used by this project and it will be described with detail in the next section.
- **PINGUINO-MICRO**. It uses a Microchip 32-bits PIC32MX440F256H with USB OTG. It has 256KB Flash and 32KB RAM. Besides, a microSD card for data logging and UEXT connector to allow modules to be connected.
- **PINGUINO-MX220**. It has a Microchip 32-bits PIC32MX220F032D microcontroller with 32KB Flash, 8KB RAM and USB OTG. Moreover, ultra low power design and Li-Po battery connector.

3.2.2. PINGUINO-OTG

Features:

- PIC32MX440F256H 80 Mhz microcontroller
- 256KB Flash and 32KB RAM

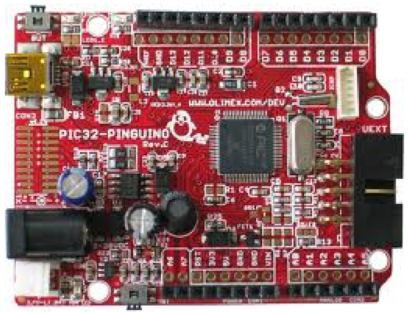


Figure 12: Pinguino-OTG. Source [14]

- MicroSD card for data logging
- Allows power supply voltage from 9 to 30V DC
- The board can be used in industrial applications as all components work reliable in industrial temperature range -25 +85C
- Low power consumption
- Uses mini USB connector
- Allows real time clock
- UEXT connector that allows many modules like RF, ZigBee, GSM or GPS to be connected.
- Noise immune design

3.2.3. ZigBee MRF24J40MA module

MRF24J40MA is a certified 2.4 GHz IEEE 802.15.4 radio transceiver module. It has an integrated antenna and supports ZigBee, MiWi and MiWi P2P protocols. The module has a 4-wire SPI interface and it can be connected to PIC32-PINGUINO and PIC32-PINGUINO-OTG thanks to the UEXT connector.

Features:

- Compatible with PIC32 microcontrollers
- 2.4 GHz IEEE 802.15.4 Transceiver Module

Figure 13: ZigBee module. Source [15]

- Low current consumption (Tx:23 mA, Rx:19 mA, Sleep:2 μ A)
- Frequency Range: 2.405-2.48 MHz
- 20 MHz clock
- It has sleep mode
- It uses an AES128 encryption
- It has RSSI

3.3. Pinguino vs Arduino

In this section, we are going to see the advantages of Pinguino compared to Arduino, that made us to use Pinguino-OTG instead of an Arduino board.

- Pinguino boards are available with 8-bit and 32-bit PIC microcontroller, while Arduino is only at 8 bits. This makes Pinguino to be faster. Moreover, Pinguino-OTG CPU works at 80MHz while Arduino UNO at 20MHz.

- Pinguino does not need an UART interface to communicate with the PC, because the PIC microcontrollers have a USB module integrated which reduces hardware costs and leaves the UART port free for applications.
- Pinguino is cheaper that Arduino.
- The MRF24J40MA module is cheaper than the Xbee shield for Arduino, what reduces the cost of the project.

The only Pinguino inconvenience compared with Arduino is the smaller community and the few number of projects available ready to use.

4. LOCATING AND POSITIONING

Several possibilities in Local Positioning Systems design exist. In this chapter, different magnitudes and ways to process the measurements are going to be explained. These systems are usually based on two kind of devices:

Slave
: One or several devices, generally static, that respond to the requests of the Master.

Master
: It is the device to locate. It asks for information from the rest of devices to extract the magnitude needed.

4.1. Magnitudes

Location is usually based in one of the next magnitudes. It is also possible to combine some of them to increase the precision or combine two procedures or methods.

4.1.1. Time of Arrival (ToA)

ToA (also called Time to Fly (ToF)) technique measure the time that the signal from the Slaves takes to arrive to the Master. There is a linear relation between the propagating time and the distance, therefore, it is possible to locate the Master using trilateration. This is the technique used by GPS. The disadvantage of this magnitude is that require a synchronization mechanism where all the devices must have one real time clock perfectly synchronized and one variation in the clock can produce uncertainty.
Using the Round-trip Time of Flight (RToF) the synchronization problem is avoided.

4.1.2. Angle of Arrival (AoA)

For this magnitude, the Slaves must have two antennas (at least) to measure the distance of reception time, what allows the microcontroller to calculate the angle

with the Master. After that, the location estimation is based in triangulation of these three angles.
These measurement is pretty precise, but not all devices have two antennas to find the angle, and the ones which have them are more expensive.

4.1.3. Received signal strength indication (RSSI)

This measurement is based on the power level of the signal received in the Master. As radio waves propagate following the inverse-square law, the distance can be determined from RSSI values. The main advantage is that it does not require that the Master has a direct sight with the Slaves.
It is a magnitude sensitive to reflections and absorption from walls. However, precision can be achieved using trilateration or *fingerprinting*. Both will be explained in next section.

4.2. Locating Methods

4.2.1. Triangulation

Triangulation is based in the next property: knowing the vertexes of a triangle and its internal angles, it is possible to determined any position inside the triangle. Each Slave will be on the vertex of the triangle. Knowing the AoA of each Slave and using geometry, the position can be found.

4.2.2. Trilateration

While triangulation uses angles, trilateration uses only distances to estimate the position of the object to locate in a bidimensional plane.
For that, the distance must be calculated through three antennas,at least. These antennas have a circumference reach where the location point will be. The intersection of the three circumferences will determine a small region where the object will be located. If we consider the centre of that region, we will have located the object, and the distance from the centre to one vertex will be the mathematical accuracy of our system.

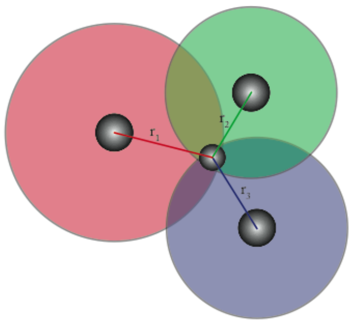


Figure 14: Trilateration

4.2.3. Fingerprinting

Fingerprinting or patterns recognition, tries to match the vector of the magnitude measured coming from different Slaves to a sample usually stored in a database using k-nearest neighbor algorithm.
The samples stored in the database were taken as a calibration value, so before using this method, an on-site magnitude mapping is needed. This compensates the errors produced by objects or walls, as the attenuations are the same at the moment of the calibration and the location measurement.
The main advantage of this method compared to trilateration is that the calculations are reduced, but it will be inaccurate with a change in the environment.

4.3. Accuracy in the locations

An ILS based on ZigBee will have too much incertainty if it only uses RSSI information from one Slave, due to reflections and diffractions. In this project we will improve the precision using trilateration with three Slaves. Moreover, accurating the RSSI measurements from each Slave will be done taking multiple samples in consecutive moments and using statistics models to processing them.
In addition, fingerprinting will be implemented with a database where some measurements will be added, so that errors made by attenuations or wall absorption can be compensated. As the Systems will be installed in houses, where the environment can change a lot (movements in the furniture, i.e), our system will count with a machine-learning mode where it will improve from its own mistakes.

5. PROJECT DEVELOPMENT

5.1. Project architecture

The project will have the next architecture:

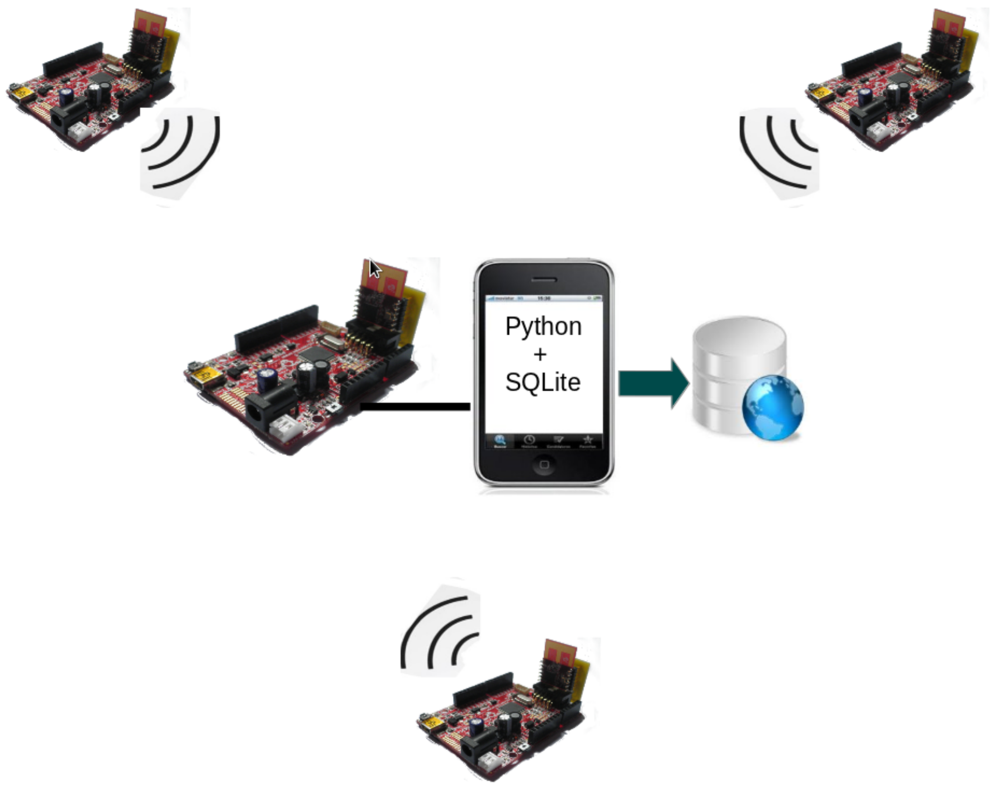


Figure 15: Project architecture

In the place to locate there will be three slaves that will send the information required by the Master. The Master will be connected to a PC, mobile phone or any device with operating system able to run the application and store the information in a database.

5.2. Project phases

- Indoor Location Basic System

  1. MRF24J40MA ZigBee module library study.
     Study of the ZigBee module library and registers to know how to get RSSI data.

2. Pinguino MRF24J40MA module library modification.

   Modification of the library to create one function that returns RSSI value.

3. Getting RSSI and verification.

   Verification and test of the function.

4. Master discovers the Slave nodes.

   Master sends a broadcast to know how many nodes there are in the network.

5. Request of consecutive RSSI samples.

   Master request a specific number of RSSI samples from each Slave.

6. Python study.

   Study of Python programming language.

7. Communication Pinguino-Python.

   Receiving data from Pinguino and sending data to it.

- Improving precision

1. Statistical study.

   Study of statistics methods to process the consecutive samples of RSSI.

2. Data processing with Python.

   Implementation of an algorithm to estimate the best RSSI value.

3. SQLite study.

   Study of SQLite databases and programming.

4. Database creation.

   Creation of a database of calibration.

5. Communication SQLite-Python.

   Connecting Python to the database, writing data on it and retrieving data.

6. On-site mapping.

   Measurement of calibration samples and implementation of a program which asks to enter measures.

7. K-nearest algorithm.

   Implementation of the algorithm which determines the room from the samples compared to the database.

8. Machine learning.

   Development of a method which will allow our system to be able of learn from its mistakes.

5.3. Project planning

In the section before, a description of the main tasks and subtasks was done. In any project, having a schedule is important so that it is easier to monitor and control the project.

5.4. Indoor Location Basic System

This main task of the project will give us the magnitude to allow location. Through a python program, we will receive consecutive RSSI values that will be processed. This task is divided in the next subtasks that makes achieving goals easier.

5.4.1. MRF24J40MA ZigBee module library study

This step in the project is one of the most important as it will allow us to get the magnitude to measure the distance: RSSI. For this purpose, we have to study the MRF24J40MA module (already explained in chapter 3) and how it gets the information, where is located and how we can access it.
This information can be found in MRF24J40 (family of our module) Datasheet [17]. From this datasheet, we can extract that RSSI values are between 0-255 and in the next figure, the relation between these values and the power that it represents can be seen.

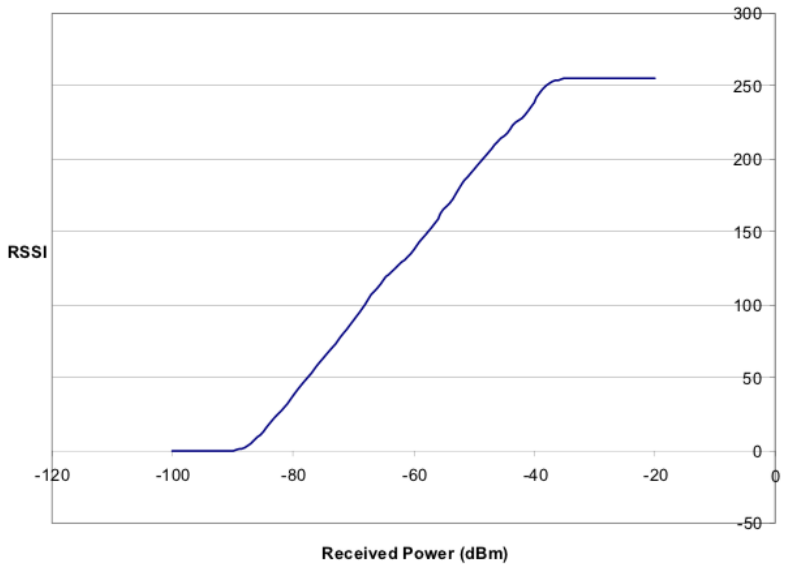
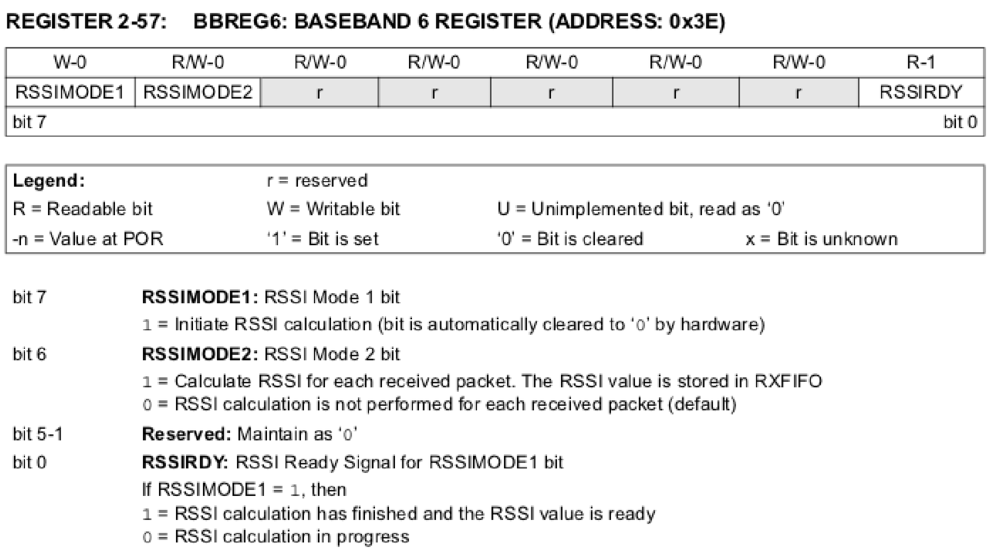


Figure 16: MRF24J40MA relation RSSI value-power. Source [17]

But, how we can get the RSSI from the microcontroller?
Going through the datasheet we find a register called BBREG6 componed by bits that allow the reception of RSSI.



REGISTER 2-57: BBREG6: BASEBAND 6 REGISTER (ADDRESS: 0x3E)

W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1
RSSIMODE1	RSSIMODE2	r	r	r	r	r	RSSIRDY
bit 7							bit 0

Legend:	r = reserved		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7	**RSSIMODE1:** RSSI Mode 1 bit 1 = Initiate RSSI calculation (bit is automatically cleared to '0' by hardware)
bit 6	**RSSIMODE2:** RSSI Mode 2 bit 1 = Calculate RSSI for each received packet. The RSSI value is stored in RXFIFO 0 = RSSI calculation is not performed for each received packet (default)
bit 5-1	**Reserved:** Maintain as '0'
bit 0	**RSSIRDY:** RSSI Ready Signal for RSSIMODE1 bit If RSSIMODE1 = 1, then 1 = RSSI calculation has finished and the RSSI value is ready 0 = RSSI calculation in progress

Figure 17: BBREG6 register. Source [17]

As it can see, RSSIMODE1 must be high to start calculating the RSSI value. The RSSI value can be appended at the end of each successfully received packet if RSSIMODE2=1 and the value will be at the end of the RXFIFO packet. Finally, when RSSIRDY=1, the module has finished calculating the value and it can be read.

The RXFIFO can only hold one packet, therefore, it is recommended that the microcontroller read the entire RXFIFO without interruption so that received packets are not missed.

For reading the RXFIFO, where the RSSI value is appended we have to follow the next steps:

1. Receive RXIF interrupt.
2. Disable microcontroller interrupts to avoid losses.
3. Set RXDECINV = 1 that disable receiving packets.
4. Read address, 0x300 that is the RXFIFO frame length value.
5. Read RXFIFO addresses, from 0x301 through (0x300 + Frame Length + 2) so we read packet data plus LQI and RSSI.
6. Clear RXDECINV = 0 that enables receiving packets.

7. Enable microcontroller interrupts.

In the last byte of RXFIFO we will find RSSI value.

5.4.2. Pinguino MRF24J40MA module library modification

In the next figure, we can see the diagram of how the library is divided and the modifications that were necessary to do. The programmer sees an IDE (integrated development environment) where some functions can be used as ZIG.send for sending information to a node. The purpose of this step is to get a function ZIG.RSSI() usable in the IDE that returns as an integer the rssi value.

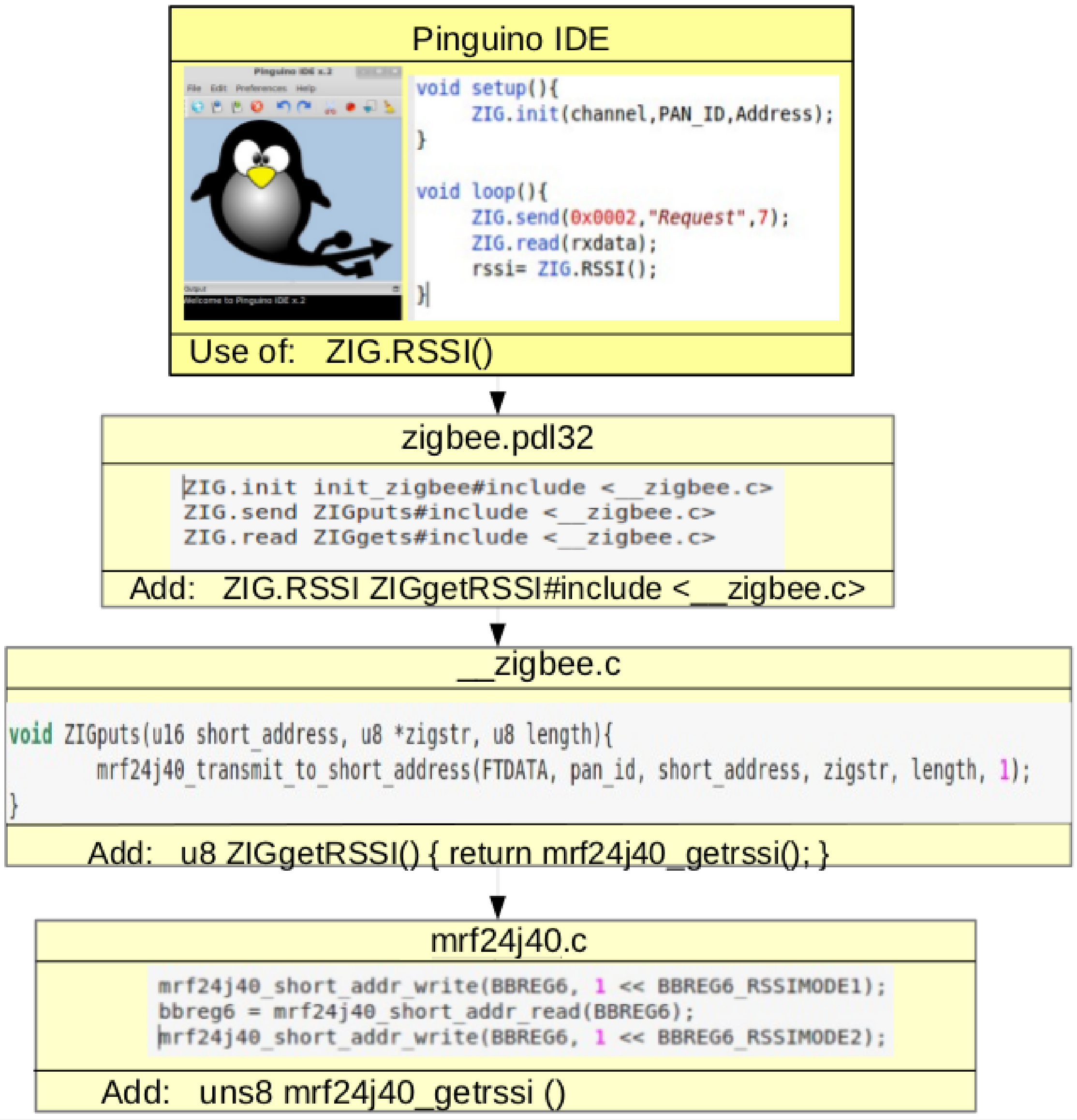
The relation between this function and its code is in the document *zigbee.pdl32* where appears declarations of the functions that can be used for zigbee and where the code of these functions is allocated, in our case, in *__zigbee.c* it will be a function called ZIGgetRSSI().
So, *zigbee.pdl32* after the modification will be as follows:

```
ZIG.init init_zigbee#include <__zigbee.c>
ZIG.send ZIGputs#include <__zigbee.c>
ZIG.read ZIGgets#include <__zigbee.c>
ZIG.RSSI ZIGgetRSSI#include <__zigbee.c>
```

The next step is to make the function ZIGgetRSSI() that is allocated in *__zigbee.c* and interconnects the manipulation of registers with the use of the function in the IDE. This function will be really simple as it will only return the value of the function that is able to manipulate registers.

```
u8 ZIGgetRSSI(){
return mrf24j40_getrssi(); }
```

The functions that are capable of using and manipulating the module registers must appear in the *mrf24j40.c* document. Here, we make a function called mrf24j40_getrrsi(), where we can decide the values for each register, therefore, we get the rssi value.


Pinguino IDE
Use of: ZIG.RSSI()
zigbee.pdl32
Add: ZIG.RSSI ZIGgetRSSI#include <__zigbee.c>
__zigbee.c
Add: u8 ZIGgetRSSI() { return mrf24j40_getrssi(); }
mrf24j40.c
Add: uns8 mrf24j40_getrssi ()

```
void setup(){
      ZIG.init(channel,PAN_ID,Address);
}

void loop(){
      ZIG.send(0x0002,"Request",7);
      ZIG.read(rxdata);
      rssi= ZIG.RSSI();
}
```

```
ZIG.init init_zigbee#include <__zigbee.c>
ZIG.send ZIGputs#include <__zigbee.c>
ZIG.read ZIGgets#include <__zigbee.c>
```

```
void ZIGputs(u16 short_address, u8 *zigstr, u8 length){
        mrf24j40_transmit_to_short_address(FTDATA, pan_id, short_address, zigstr, length, 1);
}
```

```
mrf24j40_short_addr_write(BBREG6, 1 << BBREG6_RSSIMODE1);
bbreg6 = mrf24j40_short_addr_read(BBREG6);
mrf24j40_short_addr_write(BBREG6, 1 << BBREG6_RSSIMODE2);
```

Figure 18: Library diagram

5.4.3. Getting RSSI and verification

With the modification in the library done, we can use our function in the IDE to get RSSI values. The next step is to verificate that the function is working properly and for that, we can create a program where the transmitter sends a packet to one receiver that is really close and the last one, sends the RSSI by usb, using the function CDC.print(). We can see the RSSI using Cutecom and if it is working well, the value will be 255. In this step, the main problem to be solved was that a lot of packets were lost and the RSSI values received were zero. For solving that, before sending the RSSI by usb, we can check if the value is zero and in that case, we can proceed as a ARQ protocol and request another packet where the RSSI information will be appended.

5.4.4. Master discovers the Slave nodes

At this stage, the Master is able to ask for RSSI to a specific number of slaves which destination address is known. However, the right practice will be that the Master finds how many slaves and their addresses by itself.
For this purpose, we have created a function called *find_id()* where the Master sends a broadcast to the network. Each slave that belongs to our system will send a reply to the Master and from these packets, the Master gets the number of slaves and their addresses.

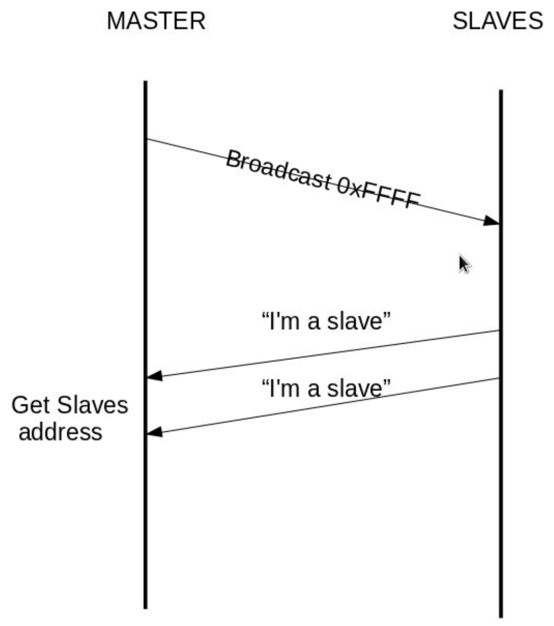


Figure 19: Discovering Slaves protocol

After that, our program is able to ask separately each slave for 10 consecutive RSSI samples that we will process getting an appropriate and accurate RSSI value for each node. This function is called *request_data()* and it uses our function ZIG.RSSI().

5.4.5. Communication Pinguino-Python

In order to process all the information that we are going to receive from the slaves, we are going to use the programming language Python, which makes development faster and easier. We need to establish a connection between our Python program and our microcontroller so that we can have all the information in variables and manipulate them.

```
def connect():
    connection= serial.Serial("/dev/ttyACM0")
    connection.setTimeout(0.8)
    variable = connection.readall()
```

The first line is the definition of the function *connect()*. The second line opens the serial port where the Pinguino is connected and the last line stores the information that Pinguino sends in *variable*. Now we have in variable ten samples of RSSI values waiting to be processed.

5.5. Improving precision

Having ten samples of RSSI from each slave, it is the moment to process all these samples to get an accurate result to be stored in a database. As it was explained Python is going to be the programming language and some statistical methods are going to be implemented.

5.5.1. Statistical study

RSSI values are quite susceptible of small changes. This means that ten consecutive samples are correlative but they are not exactly the same. In addition, there sometimes can be noise or interference that affect our measurement that should be avoided.

A statistical study was made in order to get an accurate RSSI value. Some statistics were considered as mean, mode and deviation. Mean and mode are measures of centralization, mean gives us the average value of the ten samples and mode

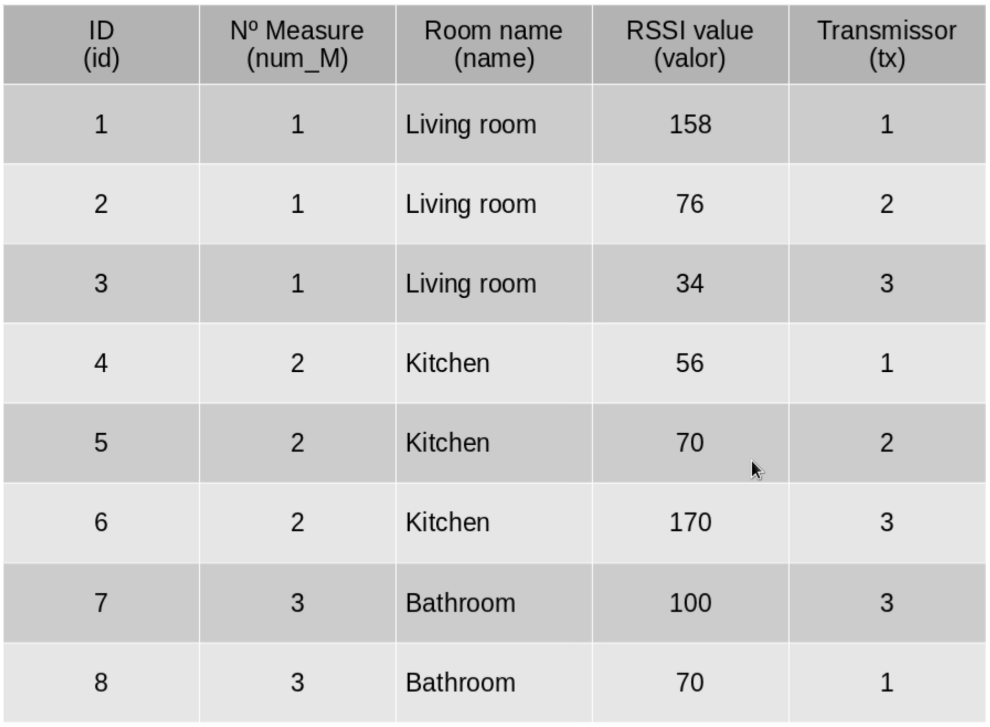
is able to give us which one is the most frequent. As first approach, the mean could be a good way to find the RSSI value, getting the average as the best RSSI. However, the main disadvantage using only the mean is that is strongly affected by data very far from the rest. In other words, if an interference affect the data having only one peak data from the rest, the final RSSI can be produce a wrong location.
In case of avoiding this, we also need to measure how far the data is spread apart from the average. Deviation is the right statistic to be used.
The result is a function called *valueRSSI()* that first, check if the deviation is higher than a constant (DEV=30). In that case, it looks at the mode which give the value most repeated. It can also happens that there is no a value repeated, so the mode will be zero and our function will eliminate from the ten samples the highest and the lowest value to eliminate the peak values and recalculate the mean. If the deviation is less than 30, the value will be the mean.
At this stage we have a Python class with the next methods:

```
class ProcessingRSSI:
    def __init__(self):  #initialize the variables
    def connect(self):  #stablish a connection with Pinguino
    def eliminateZeros(self):#filtering zeros
    def getLists(self):  #create a list for each ID with its RSSI
    def average(self):  #get the average of RSSI for each ID
    def deviation(self):  #get the deviation of each ID
    def mode(self):  #get the mode of each ID
    def valueRSSI(self):  #get the best RSSI value from all the samples
```

5.5.2. Python-SQlite communication

The next step in the project is to develop a database able to store all the measurements to use the fingerprint method. The application (in Python) will get RSSI

samples, calculate the best one and store it, so that when we try to locate, the application will look for the most similar row in the database.
The database will contain only one table with the next scheme:



ID (id)	Nº Measure (num_M)	Room name (name)	RSSI value (valor)	Transmissor (tx)
1	1	Living room	158	1
2	1	Living room	76	2
3	1	Living room	34	3
4	2	Kitchen	56	1
5	2	Kitchen	70	2
6	2	Kitchen	170	3
7	3	Bathroom	100	3
8	3	Bathroom	70	1

Figure 20: Database structure

As it can be seen, the database has 5 columns. The first one is the identification of row that it is always increasing, then the number of measure, the name of the room, the measure value and the transmitter which sent that information. According to this, every measure that is done has its own number and there will be as many rows (with the same number of measure) as transmitters where involucrated in the measurement. In this way, we can join all the RSSI values from each transmitter in one measurement with this *num_M*.
The Python application should be able of creating the database if it was not created before and connect to it. This can be done with the next code:

```
pathdb=os.path.abspath(location)
if not os.path.isfile(pathdb):
    connection2=sqlite.connect(location)
    cursor=connection2.cursor()
    cursor.execute(CREATE TABLE home (id INTEGER PRIMARY KEY,
        num_M INTEGER, name VARCHAR(50),valor INTEGER, tx INTEGER)
    connection2.commit()
else:  #just connect to the database
    connection2=sqlite.connect(location)
    cursor=connection2.cursor()
```

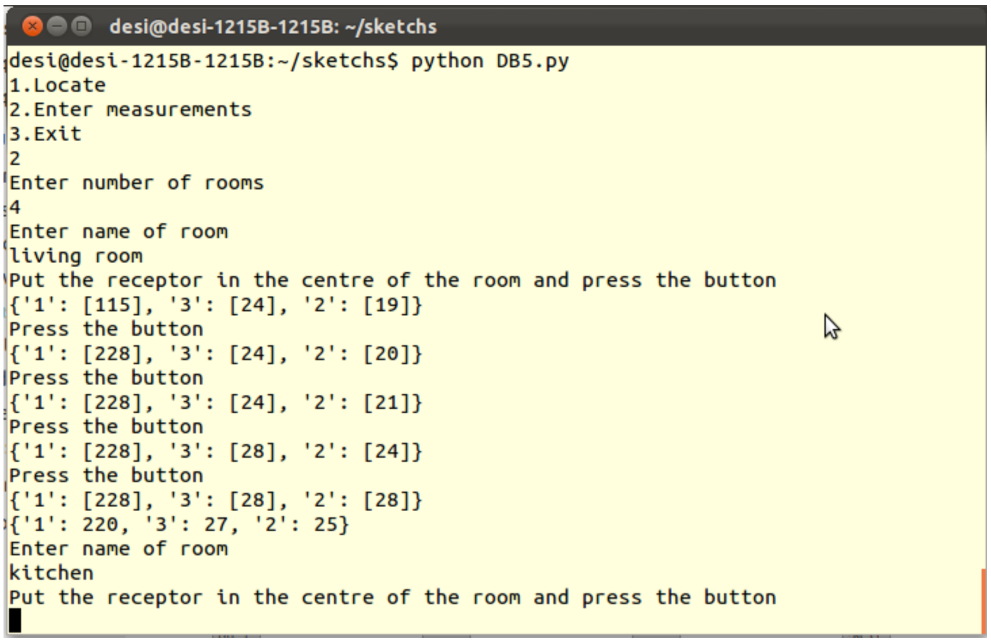
In addtition, the application will be able of enter data in the database and to retrieve it.

5.5.3. On-site mapping

Once the database is created, we need to have some measurements stored on it so that the application is able to compare using the K-nearest algorithm (implemented as it will be explained in the next point) to decide the location. The application will print an user menu and it will wait for the user to enter an option.

1. Locate
2. Enter a measure
3. Exit

In case the user types 1 and the on-site mapping was not done, it will print an error message: *Sorry but first you have to enter measurements in the database.* This will make the user to type 2. If it is the first time that the user types this option, the application will proceed to create the database and the on-site mapping. The on-site mapping is just several measurements in each room, so the application will ask for the number of rooms and it will guide the user to make the measures. It will suggest the user to be in the middle of each room and to press one button in the microcontroller, this will be repeated five times to ensure that the measure is stable (it will store the mean of all the measures for each transmitter).



```
desi@desi-1215B-1215B: ~/sketchs
desi@desi-1215B-1215B:~/sketchs$ python DB5.py
1.Locate
2.Enter measurements
3.Exit
2
Enter number of rooms
4
Enter name of room
living room
Put the receptor in the centre of the room and press the button
{'1': [115], '3': [24], '2': [19]}
Press the button
{'1': [228], '3': [24], '2': [20]}
Press the button
{'1': [228], '3': [24], '2': [21]}
Press the button
{'1': [228], '3': [28], '2': [24]}
Press the button
{'1': [228], '3': [28], '2': [28]}
{'1': 220, '3': 27, '2': 25}
Enter name of room
kitchen
Put the receptor in the centre of the room and press the button
```

Figure 21: On-site mapping

If this mapping was already done, the application will do the same but just with the actual position of the Master.

With the on-site mapping created, if the user types 1, the application will ask to press the button five times to get a stable measure and it will process all the received data in the K-nearest algorithm.

5.5.4. K-nearest algorithm

The K-nearest algorithm belongs to the function *locate()* that prints on the screen the position of the Master. This algorithm has to compare the last measure with all the measures in the database to stablish the nearest one, and therefore the room. For this purpose, it will go through all the database and it will select the measures with the same number of measurement, joining the measurements with its transmitters, so the result will be a group of tuples.

After that, this algorithm will calculate the euclidean distance (is the distance between two points in straight line and is given by the Pythagorean formula) between each tuple and the actual measure.

$$d(measure, database) = \sqrt{((m_{TX1} - d_{TX1})^2 + (m_{TX2} - d_{TX2})^2 + ... + (m_{TXN} - d_{TXN})^2)}$$

According to these distances, an error dictionary where the key will be the name of the room will be created. It will store the minimum euclidean distance of all the measures for that room, so that, getting the minimum of all the error dictionary and looking to the key, will give the name of the room.

5.5.5. Machine learning

Machine learning is an improvement of the project that allows our system to learn from mistakes in order to improve the precision and to increment the data stored in the database. It is just a modification in the *locate()* function where we add the function *machine_learning()*. This function asks the user if the location that it proposed was right or not. If the user types that it is not right, the application asks for the right room and it stores all the information in the database, so the next time that those values are measured it will this information to check. This is also a good method to improve the database and make it bigger.

6. CONCLUSIONS

This thesis has made me improve a lot in several aspects of my knowledge and I am really glad with the result.

Not only having improving a lot my IT English vocabulary, I have been able to do an important research labour and to learn by myself new technologies and programming languages. I am proud that I was able to make a modification in a library of a cutting the edge wireless technology that I think it will progress and become pretty important in sensors mesh networks.

Besides, Pinguino community is not too spread and doing a project with its board is not only a challenge because of the lack of information but also a big contribution for this platform. I have also used Python which is a powerful programming language that I did not know before and it allowed me to connect to the microcontroller and to the database. So as it can be seen, the field of the project is wide, integrating Telecommunications aspects (with ZigBee and wireless networks) and also, Computer Sciences topics (databases and programming).

Finally, the most difficult for me in this project was making and deciding the project architecture and how every component was going to be integrated.

6.1. License

This whole project is under GPL (GNU Gneral Public License) license. It is a license created by Free Software Foundation in 1989 and it is oriented mainly to protect the free distribution, modification and software use.

Its purpose is declare that the software under this license is open source and it protects it from appropriation attempts. This means that anyone can download it, use it and make modifications on it, but these modifications must be also open source.

6.2. Implementation Budget

In the next table, an implementation budget appears. As it can be seen, for a location system inside an normal size appartment is a feasible and not expensive solution.

Number	Concept	Price	Subtotal
4	Pinguino-OTG microcontroller	21.95 €	87.8€
4	MRF24J40MA ZigBee module	14.35 €	57.4€
3	3.3V-5V Power adapter	4€	12€
		Total	157.2€

Table 2: Project budget

6.3. Future improvements

The system is currently based on fingerprinting method and it is improving with time as the database is getting bigger. A first improvement could be basing the system in another measurement as time-of-flight that could give a distance in metres and matching these results with the actual system, having two parallel ways to stabilize the locations. This would be a big challenge as the time-of-flight is a magnitude where small differences (order of ns) can be translated in several meters. It is easy to get an idea if we compare to GPS that uses three satellites with atomic clocks to get the synchronization and an accurate precision.
As the mobile phone market is growing, creating an Android application would be a good idea, so the user could have it in the mobile phone and carry it while is trying to be located better than a laptop. This would not be too difficult as the application was made with Python, so only a few changes will be needed.
Finally, the application could be improved adding a GUI interface with the possibility of painting the home floor view and to point where the user is. Besides, a statistic graph will be interesting, so the user could see which room is the most visited, how many hours were spent there and also, how accurate the system is having a percentage of mistakes and right guessings.

6.3.1. Future applications

An improved version of the system could be used to track people inside buildings and joining this location system with home automation, we could get a smart house able to turn on and off the ligths while we are walking in our appartment,

as an example.
Besides, it could also be used, as a GPS, to navigate inside big buildings as shopping centres, showing where one shop is located.
Making some modifications and adding an alarm, the application will be able to warn the user when one child or a handicapped person try to leave the home and preventing some accidents.

7. REFERENCES

References

[1] J. Werb and C. Lanzl, “Designing a position system finding things and people indoors”, IEEE Spectrum, vol. 35, no. 9, Sep.1998.

[2] GNSS Indoors fading problems
`http://www.insidegnss.com/node/590`

[3] IEE 802.15.4 Standard
`http://en.wikipedia.org/wiki/IEEE_802.15.4`

[4] J. N. Ash and L. C. Potter, “Sensor network localization via received signal strength measurements with directional antennas,” in Proc. Allerton Conference on Commununication, Sep. 2004

[5] F. Seco Granja, “Indoor Location Systems based in RF“, Instituto de Automática Industrial - CSIC
`www.car.upm-csic.es/lopsi/static/.../Apuntes%20RF-LPS.pdf`

[6] ZigBee Alliance webpage
`http://www.zigbee.org/`

[7] CSMA/CA protocol diagram
`http://en.wikipedia.org/wiki/File:Csma_ca.svg`

[8] M. Galeev, Home networking with Zigbee
`http://www.eetimes.com/design/embedded/4006430/`
`Home-networking-with-Zigbee`

[9] Comparison of Wi-Fi, Bluetooth and ZigBee
`http://www.sena.com/blog/?p=359`

[10] J. Darke, D. Najewicz, W. Watts, ”Energy Efficiency Comparisons of Wireless Communication
`www.brymercreative.com/geal_2010/images/120910_zigbee.pdf`

[11] Libelium webpage: 802.15.4 vs ZigBee `http://www.libelium.com/802.15.4_vs_zigbee/`

[12] Arduino official webpage
`http://www.arduino.cc/en`

[13] Pinguino official webpage
`http://www.pinguino.cc`

[14] Pinguino Olimex distributor webpage
`http://www.olimex.com/dev/pic32-pinguino-otg.html`

[15] Microchip MRF24J40MA module
`http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en535967`

[16] Pinguino official wiki
`http://wiki.pinguino.cc/index.php/With-Zigbee-MRF24J40MA%2B`

[17] MRF24J40MA Datasheet
`ww1.microchip.com/downloads/en/devicedoc/39776c.pdf`

[18] Using SQLite in Python Tutorial
`http://www.devshed.com/c/a/Python/Using-SQLite-in-Python/1/`