

# BACHELOR THESIS



## CARLOS III UNIVERSITY OF MADRID

Bachelor's degree in Industrial Electronics and  
Automation Engineering

“A technical aid for San Rafael school disabled  
students: intelligent presence detector”

Author

Álvaro Machón Benítez

Director

Ricardo Vergaz Benito



**Title:** A technical aid for San Rafael school disabled students: intelligent presence detector.

**Author:** Álvaro Machón Benítez

**Director:** Ricardo Vergaz Benito

**THE TRIBUNAL**

**President:** Sánchez Montero, David Ricardo

**Vocal:** Sánchez Ruiz, Jorge

**Secretary:** Santayana Gómez, Raúl

Performing the defense and lecture of the Bachelor Thesis on the 6<sup>th</sup> of October of 2017 in Leganés at Universidad Carlos III de Madrid, agrees to give the GRADE of:

VOCAL

SECRETARY

PRESIDENT



*To mom and dad*

# Acknowledgements

The project has come to its end, and now it is time to be glad. It is time to admit that walking this tough and long path has been possible thanks to all the people who stayed and supported me.

Thanks to my friends from Badajoz. You always stayed there when things turned difficult. A great part of the success that I have today is undoubtedly because of you.

Thanks to my FAMily. From all the existing residences in Madrid, lucky me, I moved to the best one. You provided me with the extra energy and spirit needed in the daily situations, not only educational, but also personal ones. That is what makes us different from others. We are a FAMily.

Adrián, one day you will understand how lucky I feel of having such a brother. You used to defend me when I got in troubles, you gave me right advices when no one else did. Only you understood the frustration of failing Engineering exams and, thanks to that and the mutual support, today, we both have become Engineers. We did it.

To a very good partner, Antonio, for encouraging me to go over my psychological limits. I am more than happy to have a partner like you. Your help in my targets has been vital. I hope you a very good luck in the new stage of your life

To Ricardo Vergaz, not only my mentor in this Bachelor's degree thesis, but also a very special professor who taught me Electronics in the real world in a fascinating way. Ricardo, you taught me how to fight the Electronics problems in real scenarios, encouraged me to always go a step further. I also want to express my gratitude for allowing me to work with San Rafael school and you. Your trust in me means a lot. See you in the Photonics world, we have a lot to do there.

Mom, dad, thanks for all. Thanks for giving me the right education. Because of you, I am who I am today. You taught me how to behave responsibly and to treat people in a respectful way. You taught me how to be a successful person without trampling others. You may be proud of your kid, but the pride is mine for having the parents I have.

## **Abstract**

The bachelor thesis exposed here attempts to be part of the “Design for All” community, where age, gender, capabilities and culture do not matter. This project is guided by its design rules and methodology.

The aim of the project is to develop a functional system which will allow students from San Rafael School to easily recognize themselves, as well as the room they are entering.

The system is composed by an open source small motherboard which will be properly programmed to recognize every student going into a room inside the school. This task is carried out by scanning Low Energy Bluetooth beacons which are held by the students.

The system then will launch a custom voice message welcoming them by their own name and the name of the room they are entering. The system will also recognize the rest of Bluetooth Low Energy devices around, but only the students will be recognized and get their custom voice message. To do that, the motherboard will be adapted with a small speaker and a modest power supply.

## **Resumen**

El presente Trabajo Fin de Grado pretende formar parte de la comunidad “Diseño para Todos”, en la que edad, género, capacidades y cultura no importan. Este proyecto está guiado por sus doctrinas de diseño y metodología.

El propósito de este proyecto es desarrollar un sistema funcional que permitirá a los alumnos del colegio San Rafael reconocer la estancia en la que entran, además de reconocerse a sí mismos.

El Sistema está compuesto por una pequeña placa base de licencia de código abierto que será programada adecuadamente para reconocer a cada alumno que entre a una estancia dentro del colegio. Esta tarea será llevada a cabo escaneando balizas emisoras de Bluetooth de baja energía, portadas por los alumnos.

El sistema reproducirá un mensaje de voz personalizado dándoles la bienvenida con su propio nombre y la estancia a la que acceden. El sistema reconocerá de igual forma cualquier dispositivo Bluetooth de baja energía cercano, sin embargo, solo los estudiantes reconocidos obtendrán el mensaje de voz. Para ello, la placa base será adaptada con una fuente de alimentación y un pequeño altavoz.





---

## TABLE OF CONTENTS

<b>LIST OF FIGURES.....</b>	<b>12</b>
<b>LIST OF TABLES.....</b>	<b>15</b>
<b>INTRODUCTION AND OBJECTIVES .....</b>	<b>16</b>
1.1 INTRODUCTION.....	16
1.1.1 Design for all .....	16
Design for All principles .....	18
1.1.2 San Rafael School.....	20
1.2 OBJECTIVES .....	21
1.3 CASE OF STUDY .....	23
1.4 PHASES OF THE PROJECT .....	24
<b>SYSTEM DESIGN .....</b>	<b>25</b>
2.1 SYSTEM SPECIFICATIONS .....	25
2.1.1 General Overview .....	25
Central and peripheral devices .....	27
Bluetooth Low Energy (BLE).....	27
Internet of Things.....	33
2.1.2 List of Components.....	33
The Central device: Raspberry Pi 3.....	33
The peripheral devices: beacons.....	37
The programming language: Python .....	41
Speakers .....	43
Power source .....	44
SD card.....	47
2.1.3 Alternatives.....	48
RFID.....	48



---

2.2 BUDGET .....	54
<b>IMPLEMENTATION .....</b>	<b>55</b>
3.1 SOFTWARE .....	55
3.1.1 RASPBERRY START-UP: INSTALLATION OF OS .....	55
3.1.2 INSTALLATION OF REQUIRED MODULES .....	58
3.2 THE PYTHON CODE .....	58
3.2.1 iBeacon Scanner .....	58
3.2.2 Kids Scanner .....	61
Case I: detection of beacon in users list .....	63
Case II: detection of beacon not in users list .....	64
Case III: beacon in user list removed and added again to acknowledged list .....	66
3.2.3 Code structure .....	67
Scan method .....	68
Update Beacon List method .....	70
Play Welcome method .....	72
3.2.4 Other implemented methods .....	74
__out method .....	74
Exit_gracefully method .....	76
3.3 PROGRAM FILES .....	77
Config.json file .....	77
Blescan.log .....	80
Music files .....	81
3.4 PROGRAM START-UP .....	82
3.5 CODE SPECIFICATIONS .....	83
3.4.1 Timeout .....	83
Concept .....	83

---



---

Value .....	84
3.4.2 Threshold .....	84
Concept .....	84
Value .....	85
3.5. FINAL BUDGET .....	86
<b>CONCLUSIONS AND FUTURE DIRECTIONS.....</b>	<b>88</b>
4.1 CONCLUSIONS .....	89
4.2. FUTURE DIRECTIONS .....	90
<b>REFERENCES .....</b>	<b>94</b>
<b>ANNEXS.....</b>	<b>102</b>
ANNEX I. USER’S MANUAL .....	103
ANNEX II. BLESCAN.PY CODE .....	112
ANNEX III. TESTBLESCAN.PY.....	118
ANNEX IV. KIDSSCANNER.PY.....	120



---

## LIST OF FIGURES

<i>Figure 1. Design for All Foundation layout [4].</i> .....	18
<i>Figure 2. Wheelchair accessible vehicle [6].</i> .....	19
<i>Figure 3. San Rafael school entrance [7].</i> .....	20
<i>Figure 4. Traffic light with sound signal [8]</i> .....	22
<i>Figure 5. Gantt diagram</i> .....	24
<i>Figure 6. General Schematic of the complete system.</i> .....	26
<i>Figure 7. Examples of central and peripheral BLE devices [9].</i> .....	28
<i>Figure 8. Classic Bluetooth vs. BLE devices [15]</i> .....	30
<i>Figure 9. GATT Profile layout [17].</i> .....	31
<i>Figure 10. Raspberry Pi 3 top view [20].</i> .....	35
<i>Figure 11. Several beacons of different brands [25].</i> .....	37
<i>Figure 12. Button cell widely used in beacons [26].</i> .....	37
<i>Figure 13. Beacons distribution in a store [28].</i> .....	38
<i>Figure 14. BLE key finder [30].</i> .....	39
<i>Figure 15. Bluesky BBTS10BK portable speakers [40].</i> .....	44
<i>Figure 16. Raspberry Pi 3 GPIO Header [42]</i> .....	45
<i>Figure 17. 5V/2A generic charger [43].</i> .....	46
<i>Figure 18. Commuted 5V/2A power supply [44].</i> .....	46



---

<i>Figure 19. Samsung micro USB 2A charger [45].....</i>	<i>47</i>
<i>Figure 20. 16 GB SD card [46] .....</i>	<i>47</i>
<i>Figure 21. Generic Tag layout [47].....</i>	<i>48</i>
<i>Figure 22. RFID different tags [49]. .....</i>	<i>49</i>
<i>Figure 23. NOOBS pre-installer screenshot.....</i>	<i>57</i>
<i>Figure 24. Raspbian installation screenshot. ....</i>	<i>57</i>
<i>Figure 25. Terminal scan of BLE screenshot.....</i>	<i>59</i>
<i>Figure 26. Kids Scanner general flowchart.....</i>	<i>61</i>
<i>Figure 27. Kids Scanner detection screenshot with listed user. ....</i>	<i>63</i>
<i>Figure 28. Kids Scanner detection screenshot with user not listed. ....</i>	<i>65</i>
<i>Figure 29. Kids Scanner detection screenshot with listed user. ....</i>	<i>67</i>
<i>Figure 30. Scan method flowchart.....</i>	<i>68</i>
<i>Figure 31. Update Beacon List flowchart.....</i>	<i>71</i>
<i>Figure 32. Play Welcome method flowchart.....</i>	<i>73</i>
<i>Figure 33. __out method flowchart.....</i>	<i>75</i>
<i>Figure 34. JSON generic object structure [65]. ....</i>	<i>78</i>
<i>Figure 35. JSON generic array structure [65]. ....</i>	<i>78</i>
<i>Figure 36. Configuration file layout example screenshot.....</i>	<i>79</i>
<i>Figure 37. Log of case III scenario example. ....</i>	<i>80</i>



---

*Figure 38. Audio files folder ..... 81*

*Figure 39. rc.local file modified to start KidsScanner.py ..... 82*



---

## LIST OF TABLES

<i>Table 1. Class, power consumption and typical distance of Bluetooth communications. ...</i>	<i>29</i>
<i>Table 2. Comparison between Raspberry pi 3 and pi 2B+ models.....</i>	<i>36</i>
<i>Table 3. RFID typical frequencies. ....</i>	<i>51</i>
<i>Table 4. Price of RFID readers. ....</i>	<i>53</i>
<i>Table 5. Price of RFID tags.....</i>	<i>53</i>
<i>Table 6. Estimated budget of complete system.....</i>	<i>54</i>
<i>Table 7. Threshold values for different distances .....</i>	<i>85</i>
<i>Table 8. Final system budget. ....</i>	<i>86</i>
<i>Table 9. Engineering costs.....</i>	<i>87</i>
<i>Table 10. Total budget of the project.....</i>	<i>87</i>



# CHAPTER 1

## Introduction and objectives

### 1.1 Introduction

#### 1.1.1 Design for all

According to the WHO (World Health Organization), 15% of the world population suffers from any type of disability, pointing out that between 110 and 190 million adults have significant difficulties in functioning. Also, the WHO affirms that these numbers are increasing constantly due to the ageing populations and the increase in chronic health conditions [1].

Design for All is a designing philosophy whose objective is to build environments, products and services which can be used and enjoyed by anyone without caring about





---

people age, gender, capabilities and culture. This is, without any type of neither physical nor psychological barrier [2].

Design for All intends to create products and services for the highest possible number of people, with designs that can be used normally by any user. This not only has the benefit of allowing any person to live a normal life in terms of independency, but also has a really important benefit for their mental health by increasing their self-esteem, pushing them into a longer and happier life.

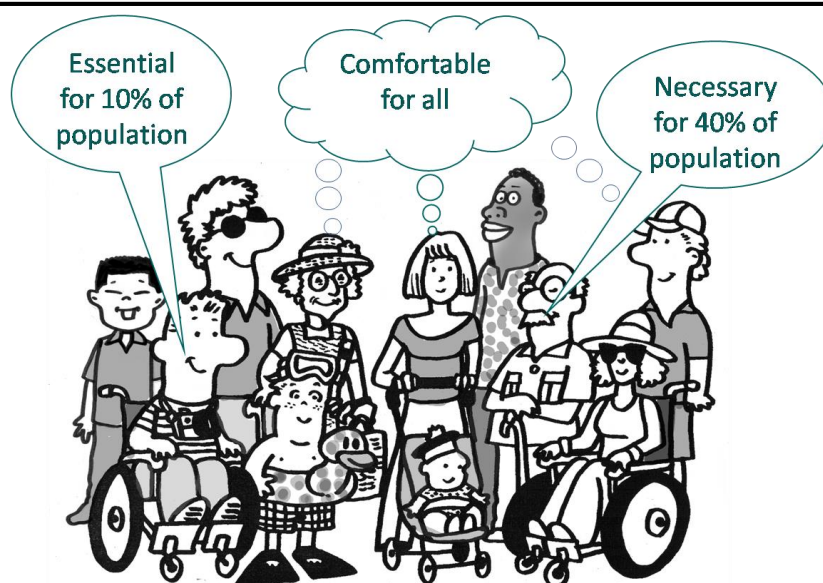
This model of designing was launched by the EIDD Design for All Europe platform together with the European policies about accessibility.

Design for All is defined in Spain by law 51/2003, 2<sup>nd</sup> of December [3], about equality of opportunity, no discrimination and universal accessibility of disabled people, as:

*“The activity which is conceived from the origin, and always that is possible, environments, processes, goods, products, services, objects, instruments, devices or tools, so they can be used for all the people, in the most possible range”*

According to the Design for All foundation (see figure 1), this type of design concept is: fundamental for 10% of the population, necessary for the 40% of the population, and comfortable for the 100% of it [4]. It is true that the importance of universal accessibility is increasing, but there is still a long way to go to satisfy disabled collective necessities.

The first step in this project was, therefore, a visit to the school. The aim of this first visit was to ask what they do need, what the requirements are. This was a fundamental step to comply with the Design for All philosophy: the end users must be present in the whole design process in order to avoid fatal –non reversible- errors in it.



© Design for All Foundation

Figure 1. Design for All Foundation layout [4].

### Design for All principles

The following principles are intended to be used as a guide in the design of products and services which fulfill the aforementioned users' conditions and allow them to use the devices, regardless the abilities they have. According to the Center for Universal Design in NCSU (North Carolina State University), the Principles "may be applied to evaluate existing designs, guide the design process and educate both designers and consumers about the characteristics of more usable products and environments" [5].

The 7 principles are:

1. Equitable use: the design is useful and marketable to people with diverse abilities.
2. Flexibility in use: the design accommodates a wide range of individual preferences and abilities.
3. Simple and intuitive use: use of the design is easy to understand, regardless of the user's experience, knowledge, language skills or current concentration level.
4. Perceptible information: the design communicates necessary information

---

effectively to the user, regardless of ambient conditions or the user's sensory abilities.

5. Tolerance for error: the design minimizes hazards and the adverse consequences of accidental or unintended actions.
6. Low physical effort: the design can be used efficiently and comfortably and with a minimum of fatigue.
7. Size and space for approach and use: appropriate size and space is provided for approach, reach, manipulation, and use regardless of user's body size, posture, or mobility.

Nowadays, there are many devices and tools “designed for all” in our environment, covering a wide range of situations, from simple automatic doors in markets, pharmacies, banks... which open under the presence of any moving object that has interceded with the laser beam installed on the top of it, to cars with pre-installed ramps for wheel chairs, audiobooks and more ideas that have been implemented along the time.

As it was compulsory to follow the aforesaid principles to design the requested solution due to the users involved, the first step in this project was to visit the school to have a conversation where professors could explain what the requirements were, as well as they introduced the school, their facilities and the children. More visits to the school were scheduled over the process of the design to assure the device was proper for the purpose.



Figure 2. Wheelchair accessible vehicle [6]

---

## 1.1.2 San Rafael School

Founded in 1976, San Rafael School belongs to the order of San Juan de Dios Brothers and it is dedicated to students with special necessities covering the age range from 3 to 21 years old.



Figure 3. San Rafael school entrance [7]

Located in Calle Serrano, 199, Madrid, the school has several types of rooms adapted for the kids' necessities, such as toilets, dining room, physical therapy room, psychologist's room, sensorial room and the classes. Every room in the school has a drawing at the entrance which represents the activity that is carried out inside so the kids associate them.

The educational offer is divided into three stages to cover all the possible ages:

- Preschool (3-5 years old)
- Compulsory Basic Education (6-15 years old)
- Transition to adult life program (16-21 years old)

Kids in San Rafael school enjoy several types of activities to stimulate their capabilities and senses, such as:

- Playing workshops.



- Adapted sports.
- Sensorial workshops.
- Communication and leisure workshops.

All the devices used by the children are adapted properly so they are able to use them with ease. It is not always easy to find such type of products since the market is full of goods for general purpose, but tends to forget about the minorities who need special products. Apart from that, this type of products have a really high price. To solve that obstacle, the school tends to adapt normal products to the specific necessities of the environment. An often used technique by the teachers is to adapt the devices with a jack cable. The device will have the female connector of the jack cable, and every child will be able to connect their push buttons, which have the male connector of the jack, to the device. This way, every kid can interact with the game or device normally.

Displays and Photonic Applications group from Universidad Carlos III de Madrid (GDAF-UC3M) has been collaborating with San Rafael school for several years through this type of projects by designing devices adapted to the necessities stated by the professors of the center, who help with meeting the requirements of the design by introducing the center, the children and the problem to be solved.

## **1.2 Objectives**

The main goal of any project under the Design for All basis, and also this one, is to develop devices which can be used by anyone, regardless their physical and/or psychological conditions, helps a considerable part of the population and it is necessary for a certain part of the users.

Specifically, this project plans to design a system to be installed in every door of the school that could be crossed by the children, which let them know where they are entering together with their own name in custom a voice message.

---

By this simple idea, the system is helping the children by orienting them inside the building, letting them know where they are. Launching the name of the room together with their names is a plus: this way, every kid knows the device is talking to them, not to others. As with other systems in this school, this feature is oriented to get a better stimulation and to increase environment relationship of the kids.

This type of orientation ideas are not something new: we can find many places where people is welcomed by a short voice message, such elevators, which tell the passengers the floor where they are, the button they pressed, as well as some indications about the doors (and even the weather forecast, in some new models).

Another example of sounds for orientation is the use of them in traffic lights for pedestrians, which emit a sequence of beeps when the green light is ON for them so they can safely cross the lane.



Figure 4. Traffic light with sound signal [8]

This example connects with the idea of use behind the device: using sounds to help users to get knowledge on their current position. Even if the idea might look a bit simple for most of the people, many children in this school suffer from any type of visual impairment, or the relation with environment problems due to their cerebral palsy, what



---

makes a suitable scenario to design a device such as the one developed in this project.

The device designed in this project aims to help the kids not only to know the room they are entering, but also to grow their ability to interact with the environment and to relate the name of the room to the activity that will be carried out inside it.

### **1.3 Case of study**

In the first visit to the school, teachers and caregivers exposed the existing problem: more than 10 rooms surrounded by around 30 children of diverse ages who get distracted by any visual or musical stimuli, capable of moving along the school, each one by his own method, i.e. by wheelchair, crawling and, in less cases, on foot.

This causes kids with diverse disabilities to be wandering around the school freely, willing to touch and play with anything that might be on their scope. As a result of this, there could be cases in which children are found confused and disoriented in a random room.

Professors proposed to obtain a device that identifies children nearby and emits a short voice message with the identification of the room to reinforce their own location, and also their name, to guarantee the kids to listen their own message.

To do that, it is proposed a system in which a central device is installed on the frame of every door at the school, scans the peripheral devices, called beacons, with a unique identifier for each one, being carried by every kid at the school.

The main requirements exposed by the professors where related only with the interaction with the kids since a discreet and modest device was needed. Professors explained that this requirement was essential due to the constant intentions of the children to touch, play, drag and pull out any piece that moves, lights up or simply reveals their position by any stimuli.





Modularity is a must in this project since, depending on the room and the kids entering it, personnel from school may need to remove the device eventually or even relocate it with the pass of the time. To fulfill this important condition, a small and detachable motherboard was considered as a first approach, fed by a modest power supply and complemented with an external speaker to launch the voice message.

This condition has to be fulfilled by the device carried by the kids, so a small device that can be carried in their own bags or attached to the pockets was considered.

The complete list of the material used to build the complete system, hardware and software, can be found in section 2.1.2 List of Components, starting from page 33.

## 1.4 Phases of the project

The phases the project has been through are collected in the Gantt diagram appended in figure 5.

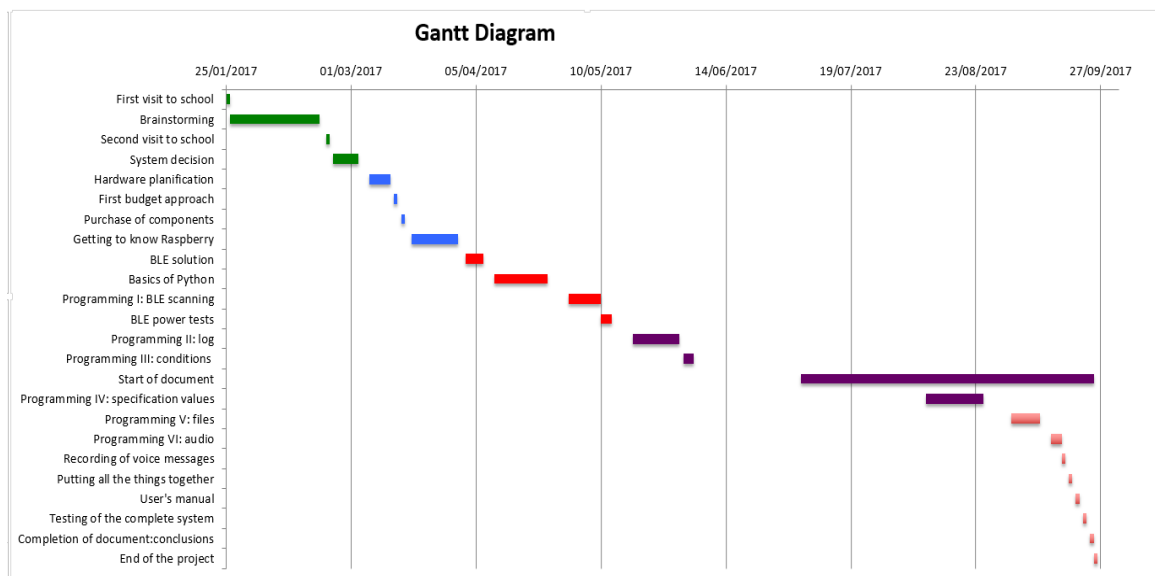


Figure 5. Gantt diagram





# CHAPTER 2

## System Design

### 2.1 System Specifications

#### 2.1.1 General Overview

As agreed with the school professors, the complete system must be as discreet as possible in order to avoid the attraction of the attention of the children. The system has to be a simple, small and modular device which scans the environment constantly, searching for the kids who penetrate the room in which the device is installed. This device is called “Central device” and is in charge of collecting data and deciding what to do with it. From the kids’ part, every kid carries a really discreet beacon called “peripheral”, which contains a unique identifier for every kid who possess one, together with information about the power of the beacon. The relative position of the kids with respect to the central device, located on the frame of the door, is calculated. If the power is strong enough, the

kids are considered to be entering the room.

The school wanted the complete system to be as robust as possible, but also low cost, so one the main conditions used to find the most suitable option to develop the required system was the price of every component, specially the scanner.

Figure 6 shows a generic schematic of the complete system proposed which can be installed in any room of the school.

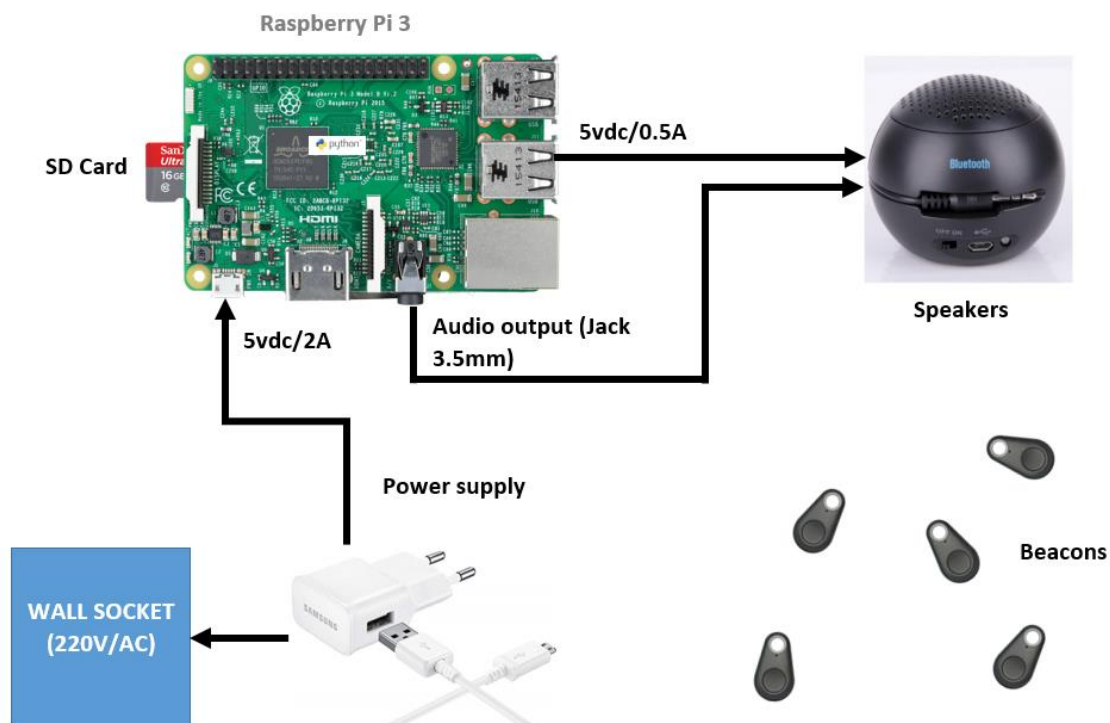


Figure 6. General Schematic of the complete system.

The system will be installed in the doors' frame of the rooms. To ensure that the kids cannot reach the device and cause damages to it (or to them, due to electrical currents), the chosen height is 1.70m.

This height is enough to make sure that the device is out of the scope of the kids, but also cables of the power supply can reach the motherboard.



---

## Central and peripheral devices

Central and peripheral devices interact to become a completely functional system. In systems, as computers, the central device is a unique part which receives, collects and interacts with data provided by its peripherals. These peripherals could be mice, keyboards, screens, USB sticks, hard drives... Even smart watches, anything that can be paired or connected to the CPU, and this is, the Central Processing Unit

This type of structures do not only work for computers, human bodies work in a similar way: our brain is the CPU, connected to our peripherals: eyes, ears, nose, mouth and skin. These peripherals interact with the environments and transmit the information to the central, which will react depending on the information received.

The system described in this chapter behaves in a similar way. As a central device, a motherboard Raspberry Pi 3 is chosen. Using the built in Bluetooth module the motherboard has and a Python based code, the central device will search for the peripherals nearby. These peripherals are called beacons: devices that emit a Bluetooth signal transmitting information.

### **Bluetooth Low Energy (BLE)**

Bluetooth Low Energy (also known as Smart Bluetooth, versions from v4.0) is the specific technology used to transmit the information the aforementioned way. This technology is widely used in communications between our cell phones or computers and smart watches, intelligent wristbands and many type of generic sensors which work together with cell phone apps to retrieve data about weather, proximity, as well as other type of apps that allow the user to control certain house devices (air conditioner, TV).

BLE technology is raised as a new generation of wireless communications, with some common features with the classic Bluetooth technology, but developed for smart applications and simultaneous communications. Bluetooth versions are grouped as Classic from v1.0 to v3.0, and version v4.0 and above are Low Energy Bluetooth (in fact, versions from v4.0 include both Classic and LE technologies).

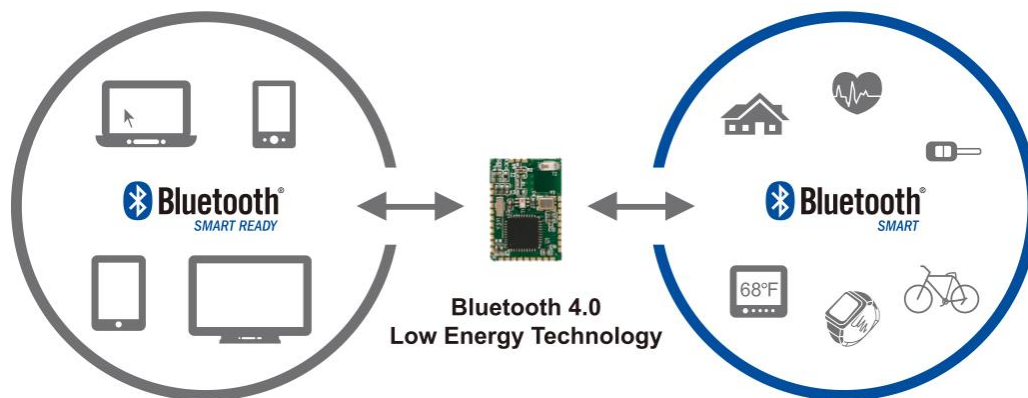


Figure 7.Examples of central and peripheral BLE devices [9].

The main differences between Low Energy Bluetooth and Classic Bluetooth are:

- The maximum distance of transmission<sup>1</sup> do not depend of the version, but on the power class<sup>2</sup>: in Classic Bluetooth, the distance range is from 1m to 100m, depending on the class (3 to 1) [10]. In BLE, distance range varies depending on the version: v4.0 uses a class which reaches up to 50m [11], while newest BLE versions can afford more than 100m (v4.1 – v5.0) [12].
- The power consumption of Classic Bluetooth is set as 1W (this value varies depending on the class of the device). BLE consumption is between 0.01-0.5W, up to 100 times better. In terms of current consumption, Classic Bluetooth has a

<sup>1</sup> Theoretical values of transmission distance; these values vary depending on external conditions, such as barriers (e.g. Walls) or devices orientation.

<sup>2</sup> Power class: power consumptions are categorized in different classes depending on the maximum permitted power.

---

peak current of less than 30mA, while BLE peak is below 15mA [12].

- BLE does not need to be paired, while this is a must in Classic Bluetooth. This is one of the main differences since BLE is intended to establish intermittent connections (only when required), unlike Classic, which is intended to establish a connection to transfer data, then finishing it.
- While Classic Bluetooth only allows connections one to one, BLE allows one peripheral to connect to only one central device, but central devices can be simultaneously connected to many peripherals [13].
- Classic Bluetooth data rate<sup>3</sup> goes from 1 to 3 Mbps, depending on the version. BLE data rate stands for 1Mbps in v4.x, and reaches up to 2Mbps [12] [14].

Class	Max. permitted power		Typical range (m)
	(mW)	(dBm)	
1	100	20	100
2	2.5	4	10
3	1	0	1
4	0.5	-3	0.5

Table 1. Class, power consumption and typical distance of Bluetooth communications.

In terms of uses, the main difference between Classic Bluetooth and Low Energy Bluetooth is that Classic Bluetooth is intended to be used for communications between devices which require a unique and constant connection, such as tablets, cell phones, computers... with cars (hands free devices), headphones, other cell phones (e.g. Files

---

<sup>3</sup> Data rate: speed of data transmission.

transfer). BLE is intended to establish an intermittent (only when it is necessary) connection with Smart Bluetooth devices, as the previously mentioned smart wrist band, heart rate sensors, key finders or weather stations.

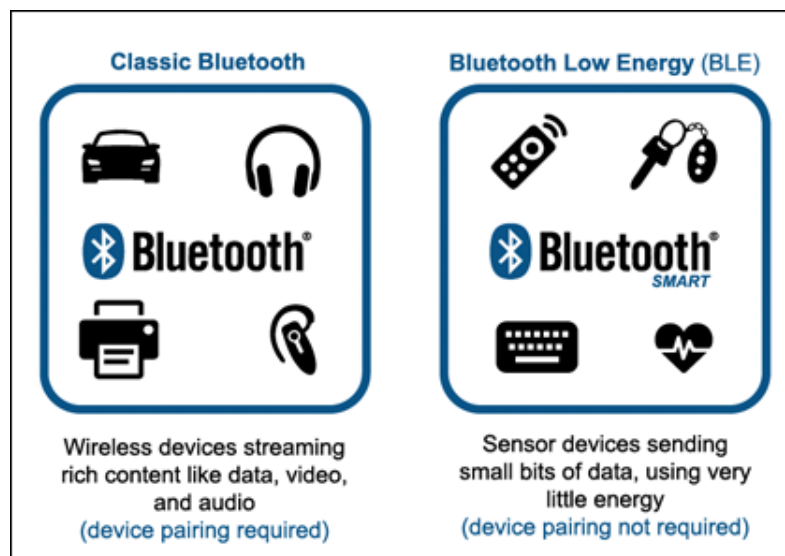


Figure 8. Classic Bluetooth vs. BLE devices [15]

BLE then is a communication solution for connections in which the distance is short (typically less than a few meters), and the power consumption is a matter: thanks to the reduction in distance, the lower data speed transmission and the intermittent connection, consumption decreases significantly [16]. Other alternatives were also considered and will be developed in next section *2.1.3 Alternatives*.

As a communication standard, Bluetooth Low Energy has its way of structuring data. **Generic Attributes Profile (GATT)** is the way of structuring data in BLE. GATT defines several elements called services and characteristics, which compose the GATT profile.

**GATT Profile** is structured in a hierarchical from. The profile stands on the top level of the hierarchy. The **profile** can be understood as the frame in which services and characteristics are allocated [17]. Actually, a profile is just a virtual frame used to get several services grouped.

Inside the profile, one or more services are contained. A **service** can be defined as a collection of data, information for a specific purpose. GATT services are defined in the specification section of the official Bluetooth webpage [18] as:

*“Collections of characteristics and relationships to other devices that encapsulate the behavior of part of a device”*

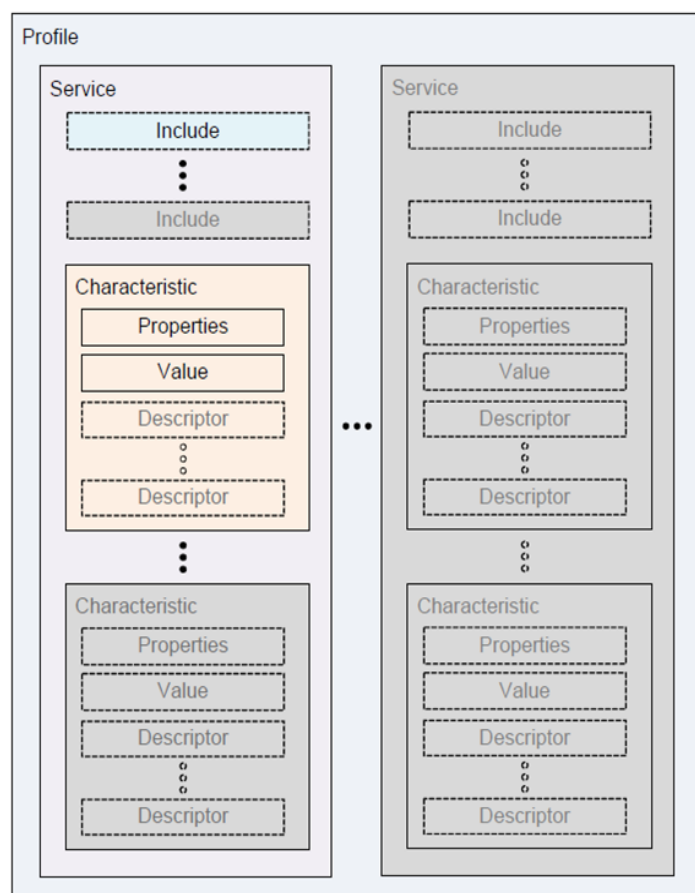


Figure 9. GATT Profile layout [17].

In the GATT specifications section, several profile and service specifications can also be found, as a Blood Pressure profile, Body Composition Service, Find Me profile, Heart Rate profile, and many more.

Inside profile files, an abstract of the document is shown, as well as the configuration



---

instructions, the used devices requirements (both collector of information and sensor), apart from connection procedures and security considerations.

The service document contains information about the measurement of the device and information about the characteristics which compose a service.

A **characteristic** is the way in which the information is described. For example, in a Heart Rate sensor, Body Sensor location is a characteristic that includes the information of the position where the heart rate sensor is located. Heart Rate Measurement would be another characteristic; in this case, this characteristic is used to send the values of the measured heart rate.

The complete example would look like this: a smart watch used to monitor heart rate. In this example, the watch would contain a heart rate *service*, and this service would contain at least the heart rate measurement *characteristic*, which is the characteristic that describes the actual measured value. Optionally, it could also contain other characteristics.

Since smart watches normally inform about their remaining battery, this example would typically contain another service as well: the battery service. For this purpose, the associated characteristic would be the battery level.

Characteristics are differentiated between them thanks to their UUID number. **Universally Unique Identifier or UUID** is a 16-bit or 128 bit number used to unambiguously identify information.

UUID's are separated in 5 different groups of characters, which can be numbers from 0 to 9 and letters from A to F. The combination is free, thus, there could be an UUID composed by either only numbers or letters. An example of an UUID could be: 54f34sd3-efgt-yy5t-5ru8-eeeeeeeeeeee.





---

## Internet of Things

The concept of Internet of Things (IoT) is closely related to the BLE technology and also to the small motherboards as Raspberry. IoT could be defined as the connection between physical devices, also known as Smart devices, buildings and vehicles, to items provided with electronics, software, hardware, sensors and actuators and also internet with the target of retrieving and exchanging data. In other words, Internet of Things is a new way of connection between our devices themselves and between them and the internet.

Thanks to the Internet of Things, it will be possible to connect homes to internet and automate the resupply of the fridge by identifying the lack of products in it. The fridge will automatically order the products to the store. Homes will be connected to online weather stations and will exchange information through IoT. This way, homes will adapt their comfort temperature, windows... Anything that can be sensed [19].

Internet of Things implementation in industry has a huge economic impact, affecting to the production (long-term money savings), but also affecting job positions, replaced by the new technology.

Anything which is used in this project, and also beacons uses described in following paragraphs, can be implemented or improved with the idea of Internet of Things and its internet use.

### 2.1.2 List of Components

#### The Central device: Raspberry Pi 3

As a central device to interact with the beacons carried by the kids, a Raspberry Pi 3 was chosen. As aforementioned, alternatives considered are explained in page 48, section 2.1.3 Alternatives. This small motherboard seemed suitable for the purpose due to its



---

really low cost, discreet appearance and wide modularity. A speaker is connected to the jack output of the Raspberry motherboard for the custom voice message to be played.

Raspberry is actually a small computer intended to ease the teaching of Computer Science at schools. As Raspberry founders mention on their official website: *“We provide low-cost, high-performance computers that people use to learn, solve problems and have fun. We provide outreach and education to help more people access computing and digital making”*.

Although there are several operating systems available on their own webpage which can be installed, their official OS is called Raspbian, and adapted version of Debian, an open source, non – commercial Operating System which works with linux.

Some of the key applications of the Raspberry motherboard are:

- Low cost PC
- Media center
- Industrial/home automation
- Print server
- Web camera
- Wireless access point
- Environmental sensing/monitoring
- IoT (Internet of Things Applications)
- Robotics
- Server/cloud server
- Security monitoring
- Gaming

For the purpose of this project, several of the above features are combined to develop an Internet of Things application. In fact, due to its huge versatility, there are many things

that can be implemented in the future to take the maximum advantage of the motherboard, for example, a camera implementation through its camera connector to use computer vision applications, the use of its GPIO pins to exploit the IoT, environmental sensing and monitoring features (e.g. ambient temperature, light and pressure conditions) and more gaming and normal computer applications by simply adding a mouse and a keyboard through its USB ports. Another possible option may be the connection of a screen to the motherboard for it to act as a media center, which could display cartoons for the kids.

Among the different models of Raspberry motherboard, Raspberry Pi 3 was finally chosen due to its clear advantage in features and the price difference was even better: the price of the model 3 was even cheaper than the model 2B<sup>4</sup>.

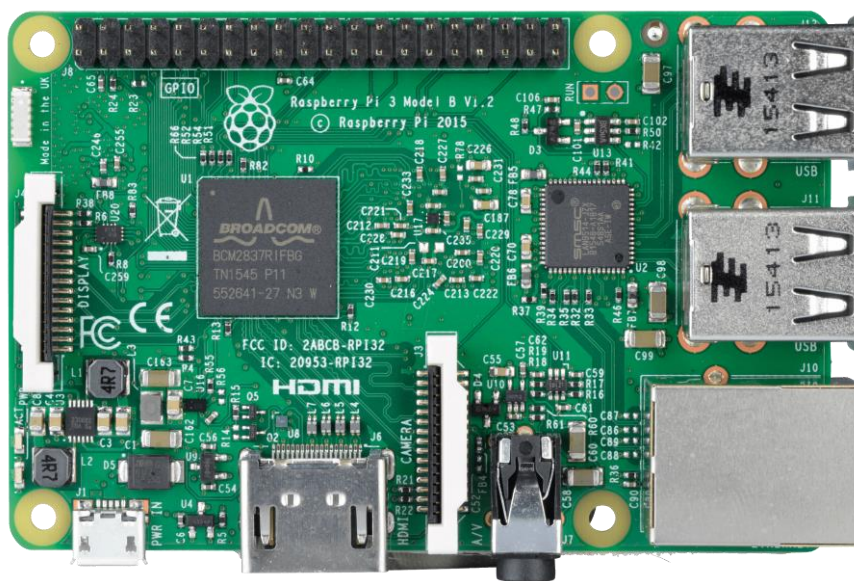


Figure 10. Raspberry Pi 3 top view [20].

<sup>4</sup> Note: at the time of the purchase of the Raspberry Pi 3 (February, 2017), the price of it was a bit higher than model 2B. Along the time, fame of the latest model has made it to lower its price to be positioned in a cheaper price than model 2B as reflected in *Table 2. Comparison between Raspberry pi 3 and pi 2B+ models*. In any case, the decision of purchasing the Raspberry Pi 3 model in February of 2017 was made due to the really low difference in price between the models (40€ vs. 31€) for model 3 and model 2B, respectively in comparison with the higher features and performance of model 3.



Main specifications of both of them are collected in table 2:

Model	Raspberry Pi 3	Raspberry Pi 2B
<b>Dimensions(mm)</b>	85 x 56 x 17	85 x 56 x 17
<b>Processor</b>	QuadCore 1.2Ghz Broadcom BCM2837 64 bits	QuadCore 900MHz Broadcom BCM 2836 32 bits
<b>Core</b>	ARM Cortex-A53	ARM Cortex-A7
<b>RAM memory</b>	1GB	1GB
<b>Bluetooth</b>	4.1 (classic and Low Energy)	No
<b>Wireless</b>	802.11 b/g/n, 2.4 GHz	No
<b>Ethernet</b>	1 port 10/100	1 port 10/100
<b>Usb</b>	4 ports (2.0)	4 ports (2.0)
<b>GPIO</b>	40 pins	40 pins
<b>HDMI</b>	Yes	Yes
<b>RCA Video /Jack 3.5mm</b>	1 port	1 port
<b>Camera connector</b>	Yes	Yes
<b>DSI screen connector</b>	Yes	Yes
<b>External storage</b>	microSD	microSD
<b>Power source</b>	5V-2.5A (max. Current)	5V-1.8A (max. Current)
<b>Price</b>	30.19€ [21]	31.90€ [22]

Table 2.Comparison between Raspberry pi 3 and pi 2B+ models

All the specifications for both models of Raspberry can be found in their website [23] [24].

### The peripheral devices: beacons

A beacon is a low consumption device which emits signals to other devices without the necessity of pairing them. Due to its small size, beacons can be easily transported by users inside their pockets. As an example, Gimbal beacons dimensions are 40 x 28 x 5.5 mm.



Figure 11. Several beacons of different brands [25].

One of the key features of the beacons is their really low power consumption, a characteristic which allows the beacon to not need a power source larger than a button cell. Beacons commonly use CR2032 button cells, which are sufficient for them to provide enough power to the beacons to last from 3 months to even several years, depending on the model.



Figure 12. Button cell widely used in beacons [26].

Beacons are widely used for many market applications in the present. Main applications are related to the position of the user with respect to the central device. Beacons and their BLE communication are more used indoors than GPS location, whose precision is worse due to the loss of signal from the satellites. This technology is mostly used thanks to the possibility of using our cell phones and PCs as a central device. BLE is available in the following Operating System versions [27]:

- Android 4.3
- iOS5+
- Apple OS X 10.6+
- Windows 8
- GNU/Linux Vanilla BlueZ 4.93+

Some of the most famous applications using beacons when the user (and its beacon) enters the scanning zone of the central device solve many daily situations [28] [29]:

- Launching the description of the item the user is facing in a museum (e.g. a painting), as the date when it was painted, relevant information about the author and its own style. This can be extrapolated to other scenarios, as zoos, architecture in old cities, and even product descriptions in stores.

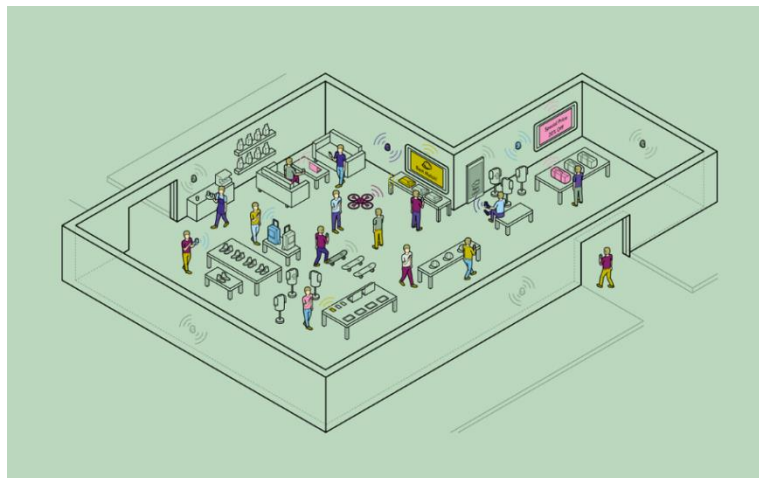


Figure 13. Beacons distribution in a store [28].

- 
- Identification of passengers in airports: some airlines are testing the location of beacons along the airports and their security lines to inform the company about possible passengers stuck in it, therefore they will calculate the probability of missing the flight and hold the plane for the remaining passengers, or even send airport cars to pick them up and take them to their corresponding gates.
  - Monitoring of pets and humans inside houses and surroundings: this application is really important for animals in a huge countryside where they can get hidden in holes and more tricky places. Humans can be monitored the same way as the industrial tracking while, for example, entering a house: kids have a curfew and parents are not home to check they got home at the right time. Then, kids have to carry a beacon with them which will be identified by a central device (for example, a Raspberry) that is connected to internet so, whenever the kids cross the entrance door, an email or SMS will be sent to their parents with the information of the arrival.
  - Key finders: beacons with the shape of a key chain attached to the keys and connected to the cell phones with the BLE technology. Every time that the keys and the phone get separated a certain distance, a sound will be played for the keys to be found.



Figure 14. BLE key finder [30]





- 
- Industry tracking: attaching of beacons to big industry containers and pallets to know and locate the stock in a warehouse.
  - Monitoring of customers path in malls as a commercial strategy to know and analyze the most frequented zones inside the building (e.g. beacons located inside hypermarket carts).

The previous list is only a bit sample of the many applications that can be developed with the use of beacons, but the list continues as far as our imagination can reach: unlocking of cars, building doors or even computers by a unique beacon, “where am I?” applications within large buildings to find specific rooms with a map, and many more.

Beacons, as any other type of Bluetooth Low Energy device, emit their UUID and, additionally, other characteristic values which identify them unambiguously. These values are:

- Their MAC (Media Access Control) address, a 48 bits identifier composed by 6 blocks of 2 hexadecimal characters [31]. MAC address is a unique identifier for any network device. Cell phones, computers, tablets, routers, printers... any networkable device have its own MAC address. In other words, a MAC address is a unique digital footprint intended for communications [32]. MAC address is a really used identifier that people use to restrict the use of a network for only certain users (i.e. routers can be configured to only allow the internet connection to the introduced MAC address of the home devices).
- Major and minor values: these values are an extra identification for beacons used to identify every beacon in a more precise way [33]. Although *UUID* values are unique for every beacon, major and minor values present a more visual relation of identification between sets of beacons. For example, a set of bought beacons can contain the same major and a different minor. Major and minor values can be





---

changed by the user. An example of application use could be a company of 7 floors in which every employee has to carry a beacon. A major is established for every floor, and personnel from the same floor would have it. Minor value would distinguish between every employee of the same floor. Major and minor values are integer numbers ranged from 0 to 65535 [33].

- RSSI (Received Signal Strength Indicator) value: value that measures the power quality of the received signal. RSSI is a relative measurement defined for IEEE 802.11 standard devices [34], i.e. RSSI has not units itself. In RSSI measurements, the higher the number, the better the signal is. This means that if the measured value is negative, values close to zero mean high quality, and negative values far from 0 mean a poor power strength [35]. In this project, The RSSI value is measured in dBm, so, as described before, values closer to zero will mean that the beacon is approaching the Raspberry.

The purchased beacons for this project is the key finder model shown in figure 14 due to its simplicity and low price, 2,9€ per unit.

### **The programming language: Python**

Created and released by Guido Van Rossum in 1991, Python is a multi-paradigm<sup>5</sup>, open source and interpreted<sup>6</sup> programming language that allows programmers to extend their way of coding to object orientation, imperative programming and functional

---

<sup>5</sup> Paradigm (Programming): programming language classification based on their features. These features may refer to the execution model, the code organization or even the style of syntax.

<sup>6</sup> Interpreted programming language: instructions are executed directly, without a previous compilation of the program. In other words, the code is executed step-by-step from the source code without being translate to the machine code. On the other hand, compiled languages have a pre-run time and are translated into the machine language prior to the execution of the program.



---

programming, and not just one style. [36]

Python was mainly designed to be read in a simple, easy way. These are a few examples of how Python eases the human reading of the code:

- Logical operators are not symbols (!, ||, &&), but words (not, or, and).
- Blocks are delimited by spaces and tabs, also known as indentation, not the same way as other programming languages as C, whose blocks are delimited by brackets ( { } ).
- Variables are dynamic, there is no need to specify its type prior to the use, and can have other values and types in a different moment of the first use. Other programming languages require to declare the variable before the use, otherwise, an execution error would show up.

Python is widely used in many different applications. Actually, according to TIOBE programming community index, a measurement of the popularity of programming languages, Python stands for the 5<sup>th</sup> position in popularity by August 17 [37]. Only Java, C, C++, and C# are over Python in this popularity list. The popularity list uses the number of skilled programmers over the world, lessons and third party vendors as indicators of popularity.

Some of the uses of Python include scripting, artificial intelligence, information security industry, apart from the already mentioned use of it by Raspberry Foundation as the main programming language, not only for deep programming, but also to teach kids how to program in an easy and user friendly interface [38].

Due to all of the previously mentioned Python characteristics, Raspberry Pi Foundation decided to adopt Python as the standard programming language for the Raspberry Pi applications. In fact, IDLE, one of the most famous Python interpreters, is pre-loaded by default in Debian versions to be installed in the Raspberry Pi [39].



## Speakers

Speakers are a must in this project: without them, the custom voice message would only be text lines in the log.

The requirements for the speakers are simple since they do not have to be really close to the entrance of the room, as long as the jack cable reaches the motherboard.

Bluetooth portable speakers would be a really nice option since they work really well in short/medium distances from the emitter, they are cheap, portable (a plus for modularity), and can be easily charged. Nevertheless, this project is already using the Bluetooth adapter for the scanner, thus, there is no chance to connect the audio via Bluetooth to wireless speakers.

As an alternative, low cost, jack connected and portable speakers can be used. The model must be as less invasive as possible to fulfill the discreet requirement but, at the same time, powered enough for the kids to hear the voice message. Like this, the speakers can be placed in any place of the room without obstructing the room more than the necessary (free space is needed in the rooms due to the fact that most of the kids move by either wheelchairs or adapted chairs), so extra space is needed.

PC speakers were also discarded. Even though the cheapest ones had a similar price as the portable ones, its power consumption was up to 10 times higher (10W), which is not needed. Moreover, they would not be as portable as the jack portables, since they need an external socket power source.

After a first search on the internet, many low cost speakers were found. The sound power of all of them was relatively similar between them and enough of the purpose (range between 1-5W).

The final purchased speakers are the Bluesky BBT10BK shown in figure 15 due to

---

its low price (9,90€) and its audio output power (1,5W RMS), which, after testing, it was checked as more than sufficient for the purpose.



Figure 15. Bluesky BBTS10BK portable speakers [40].

This model, apart from its small size, brings a micro USB connector to charge its own battery. Although the battery life (playing sounds) is only 3.5 hours, the connector will be connected to the Raspberry permanently, so it will be charged constantly through one of the USB ports of the motherboard. It will stop consuming when the battery reaches the 100%, and will only consume when sounds are emitted (plus the stand-by status, which is negligible).

This model is suitable for the purpose also considering electrical specifications since its power consumption is of 0.5A. The Raspberry will be powered by a 2A power supply so there is a great margin for the motherboard to work properly.

### **Power source**

The specifications for the power source are limited by 4 main characteristics. The suitable power source must:

- Be discreet as possible.
- Modular.
- Supply sufficient power to allow the Raspberry to work, but at the same time not

supply a higher current than the permitted (this would cause non-reversible damages on the motherboard).

- As cheap as possible.

Thus, the electrical specifications found on the Raspberry pi 3 datasheet [41] are:

- Voltage: 5V1.
- Maximum current: 2.5A. Recommended current: 2.1A.

Raspberry has two ways to be powered. One of them is the microUSB socket, powered by a 5V/2.5A (maximum current) source. The other way is through its power pins. As it can be seen in figure 16, pins 02, 04 and 06 are intended to be powered by a 5V power supply (with its corresponding ground).

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I <sup>2</sup> C)		DC Power 5v	04
05	GPIO03 (SCL1 , I <sup>2</sup> C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I <sup>2</sup> C ID EEPROM)		(I <sup>2</sup> C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Rev. 2  
29/02/2016  
www.element14.com/RaspberryPi

Figure 16.Raspberry Pi 3GPIO Header [42]

The most suitable option which covers all the 4 mentioned specifications is a simple charger. This type of power supplies are normally sold as “fast chargers” (due to the high current they supply, often sold to charge cell phones, commonly fed by only 1A, not 2A). This type of chargers are available in any store so, apart from covering the specifications,

---

they are really easy to find.



Figure 17. 5V/2A generic charger [43].

The alternative of a commuted power supply was completely discarded for the following reasons:

- Not discreet.
- Not as modular as a portable charger. Chargers can be easily unplugged.
- Not easy to find a commuted power of 5V/2A. Actually, most of the times, searches on the internet for commuted power supplies of 5V/2A directly led to conventional chargers as the ones chosen as best option. The only power supplies with these electrical characteristics found as commuted power supplies were found in low cost Chinese importation web pages [44]. The minimum current found for a commuted power supply of 5V was 2.5A, which is the actual maximum current rating. The current provided by the 5V/2A is more than sufficient taking into account that the purpose of the Raspberry is only to execute one program, so no higher currents are needed.



Figure 18. Commuted 5V/2A power supply [44].

---

The model chosen is a Samsung ultra-fast charger as shown in figure 19. The decision of purchasing this model and not others is based on its price (8.99€).



Figure 19. Samsung micro USB 2A charger [45]

### **SD card**

MicroSD card is also a must in this project since Raspberrys do not have neither internal memory nor hard disk. Before purchasing the card, Raspberry official website was consulted to find out the minimum storage capacity needed to load the Raspbian Operating System.

Although the size of the Operating System installer is around 1.5GB, the size of the complete program plus all the audio files is not known prior to the purchase of the card. Since prices of microSD cards between 2 and 8GB are really similar, 8GB card is chosen to ensure that the space is not an issue.

After looking for the 8GB micro SD card, it was found out that this type of size are not available on market anymore. The lowest size available currently is 16GB. Thus, this is the size that it was bought.

The purchased SD card can be observed in figure 20 for a price of 8,90€.



Figure 20. 16 GB SD card [46]

---

### 2.1.3 Alternatives

The first idea of technology to be implemented in the project was not using the concept of Bluetooth Low Energy together with beacons and a Raspberry Pi. In fact, this was the last alternative that could be found, but several options were considered first due to their reputation. This point describes the possible alternatives that were discarded and expects to describe the reasons why they were rejected.

#### RFID

Radio Frequency Identification, or RFID, is one of the most famous technologies of contactless ways of identification. RFID allows the identification of a device called “tag” by using radio frequency.

The system is mainly composed by tags and readers.

The tag is composed by an encoding/decoding circuit, a memory which contains the identification information, an antenna, a power supply and the communications control. There are several types of tags: active tags, which periodically emit their ID signal, powered by a battery. This type of tags are used for long distance transmissions. Passive tags do not contain a power source, so they simply wait for a signal emitted by the reader, and then, the resonant circuit of the tag absorbs the power of the reader’s antenna.

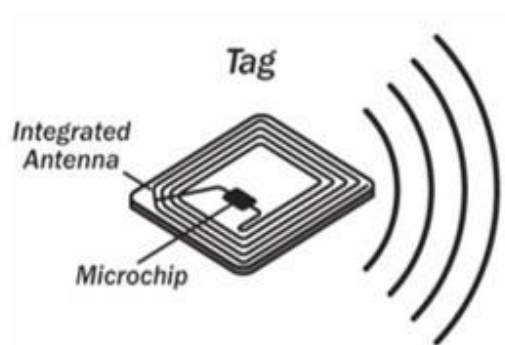


Figure 21. Generic Tag layout [47].



The electromagnetic property that allows this absorption is called Near Field. Near Field is a phenomenon which occurs in radio transmissions, where the magnetic portion of the electromagnetic field is strong enough to induce an electrical field in a coil. Due to the lack of batteries in passive tags, the distance of visibility is shorter than active tags. There is a last type, which is the battery-assisted passive, which repowers the returning signal, but it is still activated only by the presence of a reader. The actual difference between active and semi-passive tags or battery-assisted passive tags is the fact that active tags do not rely on the Near Field, while semi-passive tags do wait for the readers signal to induce the coil, although the response is also powered by the battery [48].

Tags can also be classified into read-only, whose identification number is assigned at the moment of manufacturing, or read/write: in this type, identification number can be modified by the reader.

RFID tags exist in many different presentations, as key-rings (like the BLE beacons described previously), stickers, cards, wristbands, buttons and many more.



Figure 22.RFID different tags [49].



---

The reader is composed by an antenna, a transducer<sup>7</sup> and a decoder. Readers can be classified by the type of tags and readers themselves, which are [50]:

- PRAT (Passive Reader Active Tag)
- ARPT (Active Reader Passive Tag)
- ARAT (Active Reader Active Tag)

The reader sends periodic radiofrequency signals to detect if there are any tag close by. When there is a tag in the surroundings, they generate a RF signal which sends the unique identifier contained by the integrated chip. The reader then reads the signal coming from the tag, converts the information through the transducer and processes it thanks to the decoder.

The scope of the RFID system depends on the frequency range of the devices used. The frequency ranges are classified by law, since RFID is based in the emission of electromagnetic signals and, if used incorrectly, they may interfere with other radio spectrum. For this reason, RFID has its own reserved frequency range. These frequencies are worldwide classified as ISM frequency ranges (Industrial-Scientific-Medical) [51].

Low frequencies work in short distances and commonly use passive tags, which are cheaper than the active ones and sufficient to generate the signal coming back to the reader with a proper power. High Frequencies work in longer distances, but also require a power up of the signal, this is why active tags own a battery. Next table summarizes the frequency bands and distances used in RFID technology [52].

Although ISM involves more ranges, only a few are commonly used in RFID. Table 3 shows the main four ranges implemented in RFID applications [53] [54] [55]:

---

<sup>7</sup> Transducer: device capable of transforming one type of Energy to another. In this context, radio signals are converted to electric current.



Description	Frequency Band	Max. Distance	Typical model	Key applications
<b>LF (Low Frequency)</b>	125kHz	0.5m	Passive	Control tags (animals, boxes)
<b>HF (High Frequency)</b>	13.56MHz	1m	Passive	Door entry systems
<b>UHF( Ultra High Frequency)</b>	915MHz	10m	Active/Passive	Military assets
<b>Microwaves</b>	2.45Ghz	+10m	Active	People tracking in buildings, tolls

Table 3. RFID typical frequencies.

It is important to remark that previous listed distances are theoretical values, and the real values do depend on the boundary conditions and on the seller product as well. This is the main reason why RFID was discarded. Even though RFID seemed the best option due to the fact that it was a technology created specifically for applications as the proposed in this project, the only affordable frequency range in price was LF (125 kHz), whose maximum distance is up to theoretical 50cm. There could be several configurations of positions between the central system and the beacon that could lead to higher distances.

In practice, available on market LF RFID readers only reached up to 20cm and only in a few cases; readers up to 10cm were more common. 10-20cm were completely insufficient for the project specification and no alternative of RFID reader position could



---

be contemplated to be closer to the tags: tags are intended to be inside kids' bags (or, in the case of RFID tags, sticks attached to their wheel chairs); readers would be positioned on the doors' frame. Even though the reader was aligned parallel to the tags height, the probability of missing the signal was really high since, in most of the cases, kids would enter the room further than 20cm (considering they are entering in a straight position and aligned to the reader, something that could not always happen due to their physical conditions),

Apart from the problem of the insufficient distance range in parallel heights, this position for the reader was forbidden by the school teachers: one of the main requirements of them was to avoid any possibility so the kids can touch the device. Thus, the height of the reader has to be more than the kids' hands scope. This made RFID in Low Frequency to be discarded.

Then, Higher Frequencies readers were searched, such as HF (13.56MHz), whose scope reaches up to 1m. In principle, 1m was a really good distance to work with. The problem of HF and above frequencies was the price of the reader. The price of only the reader was under no condition reachable, so RFID had to be definitely discarded, as table 4 shows.

It is necessary to remark that some 13.56MHz readers of cheap and affordable prices were found, but the maximum distance was really short for these models.

Table 4 summarizes a small portion of available tag readers and their prices, classified by their frequency range as well.



Brand	Model	Frequency Range	Max. Distance	Price
ID-innovations	ID-20LA	125kHz	20cm	39.33€
Mifare	SM130	13.56MHz	8cm	30.61€
Mifare	SL060	13.56MHz	5cm	19.30€
Mifare	SL040	13.56MHz	6cm	20.11€
StrongLink	UHF RT400A	840~960MHz	10m	235.62€
StrongLink	UHF SL130	920~925MHz	>8m	452.65€

Table 4. Price of RFID readers.

On the other hand, RFID seemed to be a really good option due to the low prices of the tags. Some of them are listed in table 5 as sample.

Brand	Model	Frequency Range	Price
Parallax	Key ring	125kHz	1.15€
StrongLink TK4100	Card	125kHz	0.29€
BricoGeek	Card	13.56MHz	2.30€
StrongLink SLK01	Key ring	13.56MHz	0.67€
StrongLink UHF	Card	860~960MHz	0.37€
StrongLink	Sticker	840~960MHz	0.55€

Table 5. Price of RFID tags.



---

## 2.2 Budget

As one of the main requirements and limitations of the system is the price, a study of the prices and a tight budget is one of the keys to success with the work. A first approach of the final budget is shown in table 6.

Element	Unitary price	Units	Total price
Raspberry Pi	40€	1	40€
Beacons	5€	10	50€
Speakers	10€	1	10€
Power source	8€	1	8€
Micro SD	10€	1	10€
-	-	-	<b>118€</b>

Table 6. Estimated budget of complete system.

This first price approach will be compared in the final budget at the end of the document to check if the price objective was fulfilled.



# CHAPTER 3

## Implementation

### 3.1 Software

#### 3.1.1 Raspberry start-up: installation of OS

As the first step of the actual project implementation, it was needed to initialize the Raspberry for the first time since it loads its Operating System from the micro SD that had to be bought independently.

To do that, the micro SD card was pre-loaded with the Operating System obtained from the Raspberry section of downloads of the manufacturer official webpage [56]. This



---

website provides users with both the Operating System installer and a proper guide to install it or also an Operating System installer called NOOBS (New Out Of Box Software) which contains Raspbian and also offers a selection of compatible Operating Systems which can be downloaded from internet and then installed. NOOBS is a suitable option for new users of Raspberry and GNU/Linux in general.

The only required material needed to start is:

- Raspberry
- Monitor
- HDMI cable
- Keyboard
- Mouse
- Power supply
- 8gb card

To launch NOOBS on the Raspberry, it has to be pre-loaded in the SD card from a computer with a SD card reader. The card must be formatted first, either with windows generic tool or with other formatting tool, as the one they recommend, called “SDformatter”.

Once the SD card is correctly formatted and NOOBS.zip file has been downloaded, it is needed to unzip these files. When the files have been unzipped, they only have to be transferred to the SD card. NOOBS version used in this project is v2.2.

The SD card then is introduced in the SD card reader of the Raspberry and it can be plugged in with the power supply.

Once the Raspberry has been powered and connected to a TV/monitor, NOOBS will automatically start and will display the following installation screen, which contains the default pre-loaded software and more options to be downloaded via internet.



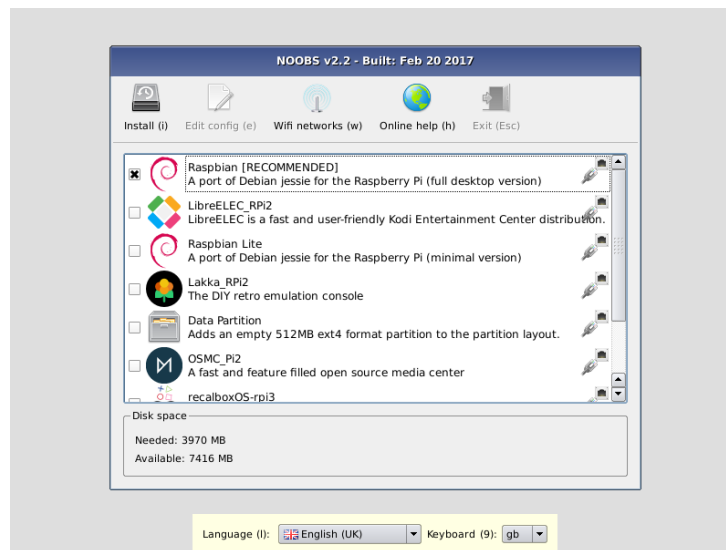


Figure 23.NOOBS pre-installer screenshot.

After selecting Raspbian as the Operating System to be installed and clicking “install” on the top of the screen, a message shows up asking for permission to overwrite the SD card.

Once confirmed, Raspbian will start installing and next screen will be launched.



Figure 24.Raspbian installation screenshot.

The process takes a while depending on the Raspberry Pi model. After the installation, a simple click on “OK” button will restart the Raspberry and Raspbian will be launched.



---

### 3.1.2 Installation of required modules

Although Raspberry Pi 3 has its own Bluetooth module included, the Raspbian Operating System is not pre-configured to scan BLE devices, so the first step in order to start programming and testing the actual program, it was needed to install some packages to make this possible.

The Bluetooth package needed to allow the Raspberry to find BLE is called Bluez. It supports more than the basic protocols that the Raspberry can handle by default, including GATT profiles, which are the ones that we need to work with. To install the corresponding libraries and the actual bluez, some commands had to be introduced in the Raspbian terminal. The commands used to install Bluez were found at Adafruit website [57].

## 3.2 The Python Code

In this section it is described the needed code to be able to scan the beacons and, once the beacons are found, how the code analyses them, this is: their UUID, major, minor and RSSI to complete the required task.

### 3.2.1 iBeacon Scanner

There is an easy way to obtain a quick list of all of the beacons nearby by simply activating the scanner through the terminal by activating the integrated Bluetooth adapter typing the following code in the Raspberry terminal:

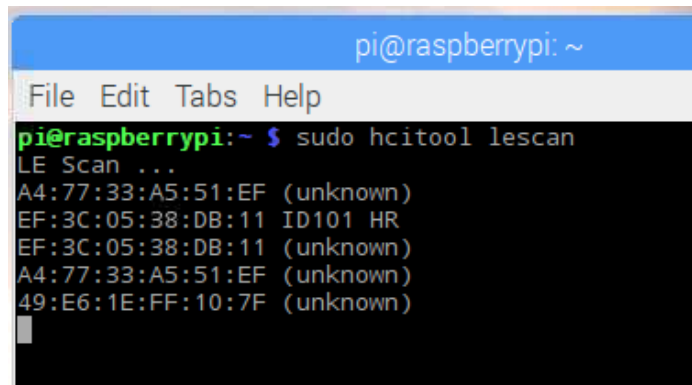
```
sudo hciconfig hci0 up
```

By using this command, the Bluetooth adapter is set up and ready to start searching Bluetooth devices.

To start scanning Bluetooth Low Energy devices, the following command can be executed:

```
sudo hcitool lescan
```

By using the above command, a scan starts showing the information displayed in figure 25.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo hcitool lescan  
LE Scan ...  
A4:77:33:A5:51:EF (unknown)  
EF:3C:05:38:DB:11 ID101 HR  
EF:3C:05:38:DB:11 (unknown)  
A4:77:33:A5:51:EF (unknown)  
49:E6:1E:FF:10:7F (unknown)
```

Figure 25. Terminal scan of BLE screenshot.

As it can be seen in the previous picture, the scanning contains only the MAC address of the devices, but UUID, major, minor and RSSI are not displayed, so this option is not sufficient for this purpose. Moreover, terminal does not allow to interact automatically with the information scanned. The purpose is to automate the analysis of each beacon power strength provided by the scanner, and perform actions depending on the values, something that cannot be coded and automated with only the terminal screen.

In order to obtain the reading of the beacons in a rough layout (only their UUID, major, minor and RSSI), a research on the internet was made to learn how to get and display the beacon information.

*iBeacon scanner* is a program composed by two different Python files, *blescan.py* and *testblescan.py*. Developed by SwitchDoc Labs, *iBeacon scanner* is an application which scans for BLE devices nearby and prints them in a more user friendly way, displaying the MAC address, UUID, major, minor and RSSI in terminal.



---

*Blescan.py* file contains all the necessary coding to print the beacons' information with the needed structure. The purpose of this file is to use the Bluetooth adapter of the Raspberry to scan for the beacons.

Imported by *testblescan.py*, it can be considered as a big method called by it whose input parameter is the Bluetooth module start command, and the output (returned value) is a list of the scanned beacons.

The code is actually in charge of scanning the environment and transform the obtained data in a more structured and user-friendly way, instead of printing the MAC address only. This helped a lot to understand and code the program. The code can be found in ANNEX II.

*Testblescan.py* is the file that has to be executed in order to scan and print a list of all the detected beacons. As previously mentioned, *testblescan.py* imports *blescan.py* to print the beacons. Actually, this file is short in lines since it only has 25 lines, where it starts the thread of the BLE scanning and the print sentence to show the scanned beacons. This is inside an infinite while loop in order to print the beacons constantly until the program is closed.

Additionally, the code includes an exception if, for any reason, the Bluetooth adapter cannot be accessed. This exception prints an error message and exits the program. The code for *testblescan.py* can also be found in ANNEX III.

*iBeacon scanner* is the part of the code which allows to print beacons in a readable way, and it is used to start coding the actual program required to fulfill the requirements, thus, *testblescan.py* file will be modified with all the conditions needed to scan, identify and play the sound for the kids.

### 3.2.2 Kids Scanner

Starting from *testblescan.py* code, *KidsScanner.py* is the actual code which scans the environment searching for the kids' beacons, analyses them and plays a sound if the beacon is included in a list of users and is close enough to the scanner. Figure 26 shows the generic flowchart of the program.

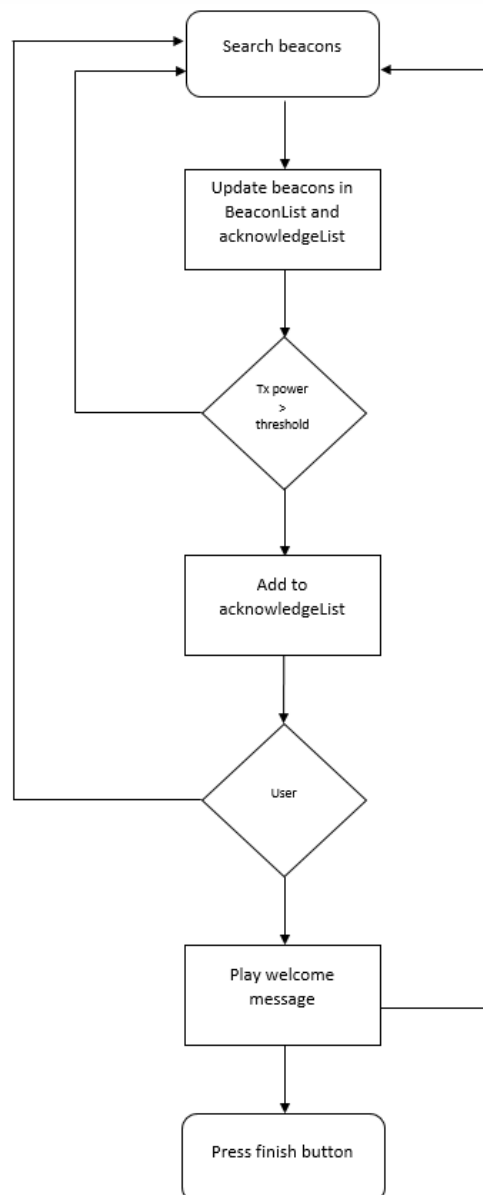


Figure 26. Kids Scanner general flowchart



---

Once the Python file is open, the Raspberry starts scanning. Once it finds any beacon nearby, it is added to a file where there is a list called “*beaconList*”. On this list of beacons, every BLE device found by the scanner is inserted (appended), it does not matter if it is a kid’s beacon or any other BLE emitted (e.g. any professor with a smart wristband). To do that, the program checks if the beacon found is already on the list.

If the beacon is not on the list of beacons yet, it is added. But if the beacon is already on the list, the program removes it and appends it again. The purpose of this is to update the last scanning date of the beacon, which will be used later in the code to decide whether the beacon is still around or it left the room. Actually, every loop of the scanner updates the list of beacons, updating (removing and adding newly) the beacons nearby, and definitely removing the ones that were not scanned in a certain time.

In order to not launch the voice message more than once, a timeout has been set. If the beacon is out of range after been added to the list of beacons, the beacon will not update its date. The code compares the last beacon time when it was added to the list with the timeout and, if the timeout has expired, the beacon is removed from the list of beacons.

Next step is to compare if the detected beacon transmission power is greater than a previously established threshold. If the power of the beacon is not over the threshold, the program continues scanning and performs no action on that beacon. If the power is greater than the threshold, then the program adds that beacon to a list of “already acknowledged beacons”.

The purpose of the “already acknowledged beacons” list is to collect all of the beacons UUID whose power transmission has reached the power threshold, meaning that they are close enough to the Raspberry to consider that the beacon has entered the room.

After adding the beacon to the acknowledged beacons list, the program checks if the beacon added is one of the users contained in a list of users. This list contains all the

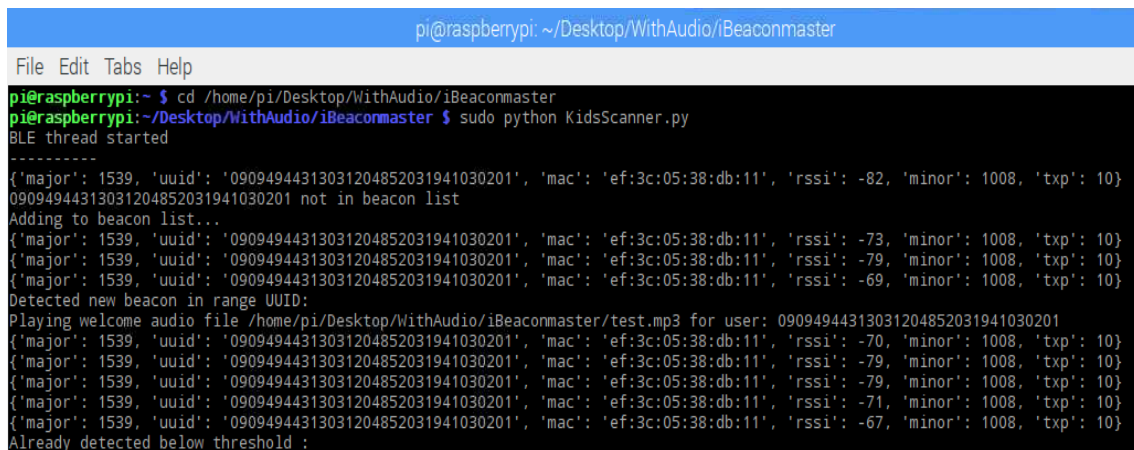
UUIDs of the kids' beacons, which are the ones who have to listen to the voice message.

If the beacon UUID is inside the users' list, the voice message for the kid will be played. Otherwise, the program will continue searching beacons. Once the beacon is removed from that list, it is also removed from acknowledged beacons list.

*KidsScanner.py* code can be found in ANNEX IV.

### Case I: detection of beacon in users list

The following screenshot is an example of use with a beacon which is included in the users list. The threshold power introduced for this example was -70dBm. For any power above -70dBm, the beacon will be added to the acknowledge list.



```
pi@raspberrypi: ~/Desktop/WithAudio/iBeaconmaster
File Edit Tabs Help
pi@raspberrypi:~$ cd /home/pi/Desktop/WithAudio/iBeaconmaster
pi@raspberrypi:~/Desktop/WithAudio/iBeaconmaster$ sudo python KidsScanner.py
BLE thread started
-----
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -82, 'minor': 1008, 'txp': 10}
09094944313031204852031941030201 not in beacon list
Adding to beacon list...
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -73, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -79, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -69, 'minor': 1008, 'txp': 10}
Detected new beacon in range UUID:
Playing welcome audio file /home/pi/Desktop/WithAudio/iBeaconmaster/test.mp3 for user: 09094944313031204852031941030201
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -70, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -79, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -79, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -71, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -67, 'minor': 1008, 'txp': 10}
Already detected below threshold :
```

Figure 27. Kids Scanner detection screenshot with listed user.

As the figure shows, after launching the application the scanning thread starts searching for the BLE devices close by.

After the first loop, the scanner detects a device in the surroundings and its information is displayed. As it is the first loop, the list of beacons is empty so the beacon is added to it ("*09094944313031204852031941030201 not in beacon list, adding to beacon List...*").



---

Since the transmission power is not enough yet to reach the threshold of -70dBm, the beacon is not added to the acknowledged list and the voice message is not played.

After getting closed to the Raspberry with the beacon, power transmission increases until it reaches the value of -69dBm, which is above the threshold. Thus, the beacon enters the set range to play the voice message. As seen in the picture, a message is printed “*Detected new beacon in range*”. Then, the beacon is included in the acknowledged beacons list.

As the beacon is included in the users list, the voice message is played for that specific user. Every user has its own customized message.

After having acknowledged the device, it goes out from the threshold range and then comes back inside it. Since the device had been already acknowledged, a message is printed (“*already detected below threshold*”), thus, no voice message is played again.

### **Case II: detection of beacon not in users list**

In this case study, the same beacon UUID has been removed from the users list. This time, the application works in the same way excepting from the launching of the voice message. This occurs due to the fact that, although the beacon is scanned by the Raspberry, added to the list of beacons and also to the already acknowledged beacons list, as it can be seen in figure 28, the beacon is not on the users’ list, thus, the voice message will not be played this time.

Figure 28 shows the program running with the test beacon getting closer to interact with the lists.



```
pi@raspberrypi: ~/Desktop/WithAudio/iBeaconmaster
File Edit Tabs Help
pi@raspberrypi:~$ cd /home/pi/Desktop/WithAudio/iBeaconmaster
pi@raspberrypi:~/Desktop/WithAudio/iBeaconmaster$ sudo python KidsScanner.py
BLE thread started
-----
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -70, 'minor': 1008, 'txp': 10}
09094944313031204852031941030201 not in beacon list
Adding to beacon list...
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -73, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -72, 'minor': 1008, 'txp': 10}
{'major': 16, 'uuid': '0100019fca507b167a0b02011a07ff4c', 'mac': '7a:16:7b:50:ca:9f', 'rssi': -102, 'minor': 523, 'txp': 0}
0100019fca507b167a0b02011a07ff4c not in beacon list
Adding to beacon list...
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -70, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -71, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -75, 'minor': 1008, 'txp': 10}
{'major': 16, 'uuid': '0100019fca507b167a0b02011a07ff4c', 'mac': '7a:16:7b:50:ca:9f', 'rssi': -98, 'minor': 523, 'txp': 0}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -71, 'minor': 1008, 'txp': 10}
-----
{'major': 16, 'uuid': '0100019fca507b167a0b02011a07ff4c', 'mac': '7a:16:7b:50:ca:9f', 'rssi': -89, 'minor': 519, 'txp': 0}
{'major': 16, 'uuid': '0100019fca507b167a0b02011a07ff4c', 'mac': '7a:16:7b:50:ca:9f', 'rssi': -90, 'minor': 519, 'txp': 0}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -72, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -77, 'minor': 1008, 'txp': 10}
{'major': 16, 'uuid': '0100019fca507b167a0b02011a07ff4c', 'mac': '7a:16:7b:50:ca:9f', 'rssi': -97, 'minor': 519, 'txp': 0}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -63, 'minor': 1008, 'txp': 10}
Detected new beacon in range
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -82, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -77, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -50, 'minor': 1008, 'txp': 10}
Already detected below threshold
{'major': 0, 'uuid': '012e244b03308306264b033083050100', 'mac': '24:4b:03:30:83:06', 'rssi': -89, 'minor': 0, 'txp': 0}
012e244b03308306264b033083050100 not in beacon list
Adding to beacon list...
```

Figure 28. Kids Scanner detection screenshot with user not listed.

As it can be seen, the UUID of the device is the same as in the previous case, and after running the code, it detects the device and adds it to the list of beacons. In fact, other unknown devices were detected by the thread (“0100019fca507b167a0b02011a07ff4c” and “012e244b03308306264b033083050100”), but even if these intruders reach the threshold range and get acknowledged (appended to the “already acknowledged beacons list”), thanks to the users list, no voice message will be displayed for them because, although they are listed in both the list of beacons and in the already acknowledged beacons list, only the beacons’ UIDD included in the users’ list will get a voice message.

Again the beacon gets closer to the Raspberry and penetrates the range but, in this case, only the “Detected beacon in range” message is displayed, not the voice message. Since the power transmission of the beacon is greater than -70dBm as it can be seen in the thread (between -50dBm and -60dBm), the message “Already detected below threshold” is displayed. This demonstrates that the beacon is already listed in the “already acknowledged beacons list”.



---

### Case III: beacon in user list removed and added again to acknowledged list.

As the complete scenario, the system must be able to work continuously with the kids' beacons, also with the intruder beacons. For that, the system must add the beacons to the list of beacons, if they penetrate the range of the threshold, must be added to the acknowledged list as well, and only the users will receive the message. In addition, beacons can leave the range for a long time, meaning that the kids abandoned the room, then, the system must forget the acknowledgment for the next time that beacon comes back to the room: the voice message must be launched again.

Next screenshot shows a complete example of the aforementioned scenario: again, the beacon used in the two previous scenarios is added to the users list so the voice message can be played when threshold range is reached.

As it can be seen in figure 29, the recognized beacon is not on the list of beacons, so the application adds it and continues scanning normally. Several lines down show how the RSSI value is not sufficient to reach the threshold until value of -59dBm is reached; then, it enters the range and the welcoming audio is played. Since it is inside the threshold, it starts to print the “*Already detected below threshold*” message. From that point, the beacon leaves for a long time. One minute after leaving the room, the beacon is removed from the acknowledged list of beacons and the message “*Removing element in already ack list UUID: [u'09094944313031204852031941030201', -59, '2017-09-19 17:24:49']*” is displayed. It is also displayed how another beacon has reached the threshold but, as an intruder beacon, no voice message is played for him.

After a certain time, the beacon listed comes back to the range, and, as shown in the figure, it is newly added to the list of beacons. Moreover, it enters the threshold directly, so the voice message is played again, only once. Several messages of “already detected below threshold” are shown to prove that. Additionally, the intruder beacon is removed from the acknowledged list.

```
pi@raspberrypi: ~/Desktop/WithAudio/iBeaconmaster
File Edit Tabs Help
09094944313031204852031941030201 not in beacon list
Adding to beacon list...
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -82, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -82, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -82, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -71, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -78, 'minor': 1008, 'txp': 10}
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -59, 'minor': 1008, 'txp': 10}
Detected new beacon in range
Playing welcome audio file audio1.mp3 for user: 09094944313031204852031941030201
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -67, 'minor': 1008, 'txp': 10}
Already detected below threshold
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -69, 'minor': 1008, 'txp': 10}
Already detected below threshold
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -43, 'minor': 1008, 'txp': 10}
Already detected below threshold
-----
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -67, 'minor': 523, 'txp': 0}
0100016323db9620550b02011a07ff4c not in beacon list
Adding to beacon list...
Removing element in already ack list UUID: [u'09094944313031204852031941030201', -59, '2017-09-19 17:24:49']
Detected new beacon in range
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -89, 'minor': 523, 'txp': 0}
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -74, 'minor': 523, 'txp': 0}
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -87, 'minor': 523, 'txp': 0}
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -71, 'minor': 523, 'txp': 0}
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -70, 'minor': 523, 'txp': 0}
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -84, 'minor': 523, 'txp': 0}
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -74, 'minor': 523, 'txp': 0}
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -74, 'minor': 523, 'txp': 0}
{'major': 16, 'uuid': '0100016323db9620550b02011a07ff4c', 'mac': '55:20:96:db:23:63', 'rssi': -94, 'minor': 523, 'txp': 0}
-----
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -56, 'minor': 1008, 'txp': 10}
09094944313031204852031941030201 not in beacon list
Adding to beacon list...
Detected new beacon in range
Playing welcome audio file audio1.mp3 for user: 09094944313031204852031941030201
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -40, 'minor': 1008, 'txp': 10}
Removing element in already ack list UUID: [u'0100016323db9620550b02011a07ff4c', -67, '2017-09-19 17:28:22']
Already detected below threshold
{'major': 1539, 'uuid': '09094944313031204852031941030201', 'mac': 'ef:3c:05:38:db:11', 'rssi': -36, 'minor': 1008, 'txp': 10}
Already detected below threshold
```

Figure 29. Kids Scanner detection screenshot with listed user.

### 3.2.3 Code structure

The code is composed only by one class called “*class testBLEScan()*”, which contains all the needed methods of the code. Out of the class, only an object is created and the *scan()* method is invoked by that object. There are several imports at the beginning of the code, needed to work with *blescan.py* file, exit signal, import of the Bluetooth module, libraries to get the current time, import of JSON to be able to work with these type of files, and finally import of *pygame*, used to play the sound.

Apart from the scan method, *class testBLEScan* contains a method called *\_\_init\_\_(self)*, which is actually the constructor of the class [58]. It initializes all the objects used in the program.

## Scan method

The scan method is composed by all the methods that performs the complete analysis and actions in the program. It is the frame in which the program is involved. Figure 30 shows the flowchart of the scan method.

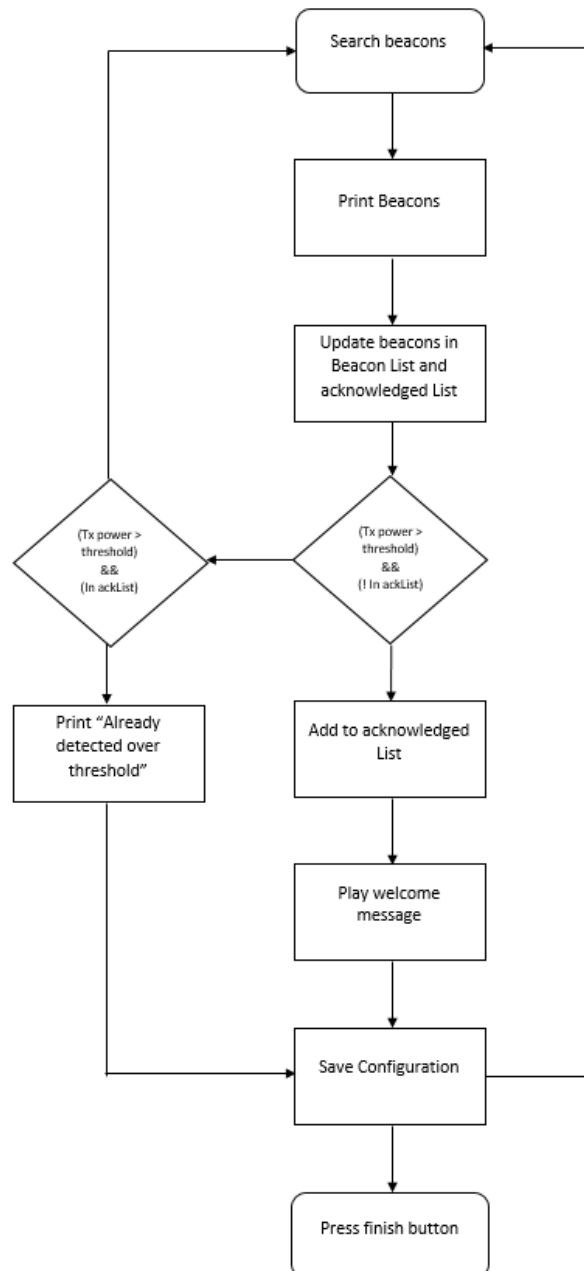


Figure 30. Scan method flowchart



---

The scan method starts by enabling the Bluetooth socket to allow the code start scanning. If this process is correct, a message “*BLE thread started*” will be printed and it will be printed in the log. If, for some reason, the Bluetooth port cannot be accessed (for example, it is being already used for any other purpose), an exception will launch an error message and the *sys.exit(1)* call will exit the program after printing the error message in the log.

Once the Bluetooth has been enabled, the scanning starts. A while loop will be constantly executed to start the actual retrieving of beacons data. To do that, beacons are retrieved into a variable called “returnedList”.

The configuration file will be open and loaded into the “config” variable in order to be able to work with the configuration parameters (users, threshold, timeout), the list of beacons and the acknowledged beacons list.

From that point, a *for* loop will go over every beacon object retrieved in “returnedList”. Then, they are printed.

After that, *updateBeaconList* is called to work with the retrieved data. When the *updateBeaconList* method is finished, *scan* method continues evaluating the power transmission of the beacon handled. In case the beacon is not listed in the acknowledged beacons list and its power is greater than the threshold, the beacon is added to its corresponding object (“alreadyAck”), with its current power and the current time. All of this process is both printed in terminal and written in the log. Finally, *playWelcome* method is called with the beacon UUID as a parameter. After that the loop will start again scanning.

In case the beacon power is also above the power threshold, but it is already in the acknowledged list, a message “*already detected above threshold*” is printed, meaning that this device has been already acknowledged, and the scanning will start again.



---

If none of these cases happen, meaning that the detected beacon power is not over the threshold level, the scanning will start again normally.

Before the start of the scanning in the next loop, all the updates of the configuration objects are dumped into the configuration file so they can be used in the next loop again.

### **Update Beacon List method**

The *UpdateBeaconList* method evaluates the beacons nearby and adds /update /remove them from both list of beacons and already acknowledged beacons list based on the time of the last known scanning compared with a set timeout.

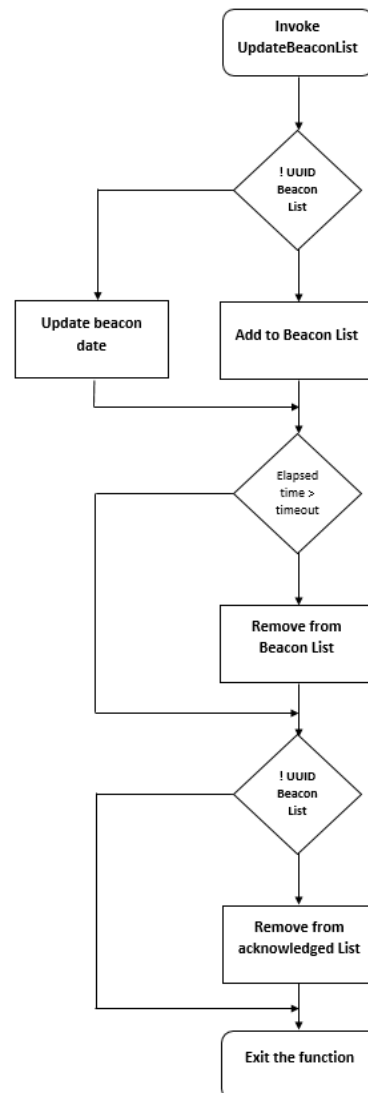


Figure 31. Update Beacon List flowchart

The method first evaluates if the scanned beacon is already on the beacons' list. In case of a new beacon, it is added to the list, and in the case it is an already scanned beacon, its recognition time is updated. To do that, the code first removes the beacon with the old date and then, appends it newly with the new date of scanning.

After that, the code makes the difference between the last listed time of the beacon on the list with the current time. This result is called "elapsed time" (i.e. time elapsed



---

since the last time the beacon was scanned). Then, the elapsed time is compared with a pre-established and configurable (in configuration file, described in “Config.json file (section 3.3 Program files).

In case the beacon has exceeded the timeout it is removed from the list of beacons. If the timeout is not exceeded, the beacon is not removed, meaning that the beacon is still around.

To end the method, the code checks if the beacon is in the list of beacons: if the beacon is listed, the program exits the method. In case the beacon has been removed from the list of beacons, it is also removed from the already acknowledged list. Then, the method is exited.

### **Play Welcome method**

*PlayWelcome* method is the specific method to launch the voice message to the users. Even though this method is always called when a beacon has reached the threshold, not always it plays the voice message. In fact, *PlayWelcome* method is in charge of evaluating if the custom voice message should be played or not, depending on the beacon UUID passed as parameter to it.

Figure 32 shows the flowchart of the complete method sequence.



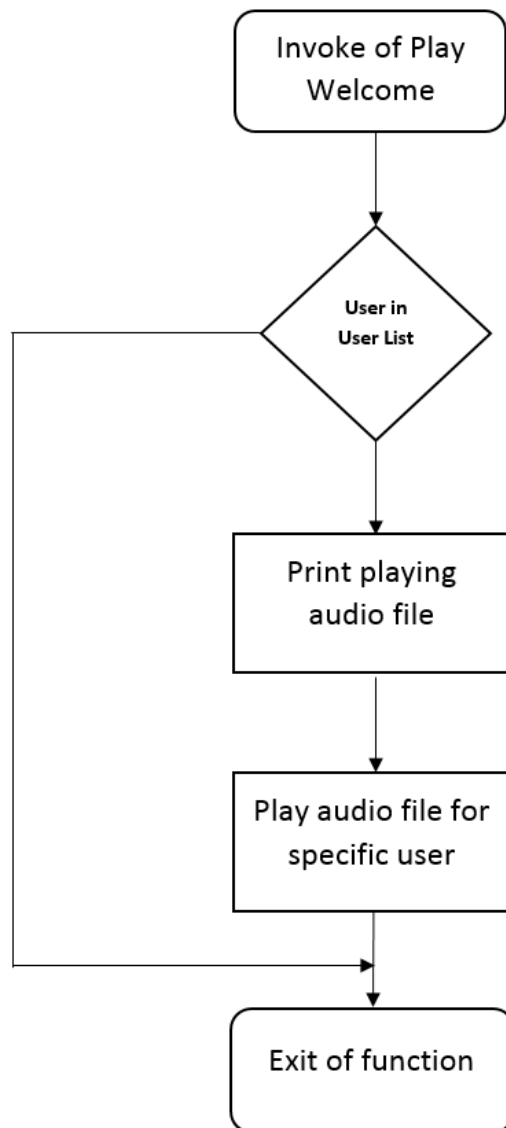


Figure 32. Play Welcome method flowchart.

When this method is called, the UUID of the beacon is passed as parameter user. The method checks if the passed user is on the list of the users introduced in the configuration file.

If so, the method will play a custom audio file for each user passed. Every user has



---

its own customized audio file stored in a path. The code executes a library called “pygame” which plays the audio file. To do that, first, pygame is invoked, then, the path of the audio file for the specific user is passed as parameter to the pygame load command. Once the audio is loaded, it is played. To correctly play the entire file, a while loop is executed always that the pygame is busy. Then, the program continues.

In fact, pygame is an open source programming language library based in Python. This library is made for developing games and multimedia applications [59]. Since this library is pre-installed in Raspbian Operating Systems, it was not needed to download and install any type of package. The actual pygame library used in this code is the music mixer module. Pygame music mixer is composed by various commands for loading and playing sounds. All of these commands, including those used in this code can be easily found in their website [60].

This method ends after writing “Playing welcome audio file:” the path and its user, to the log.

### 3.2.4 Other implemented methods

#### **`__out` method**

`__out` method was specifically designed to write every event that happens during the program in a log that will be explained in section 3.3 *Program files*, page 80.

As it can be seen in figure 33, `__out` method is quite simple. It only opens the file, writes it, flushes the buffer and exits the method after closing the file.

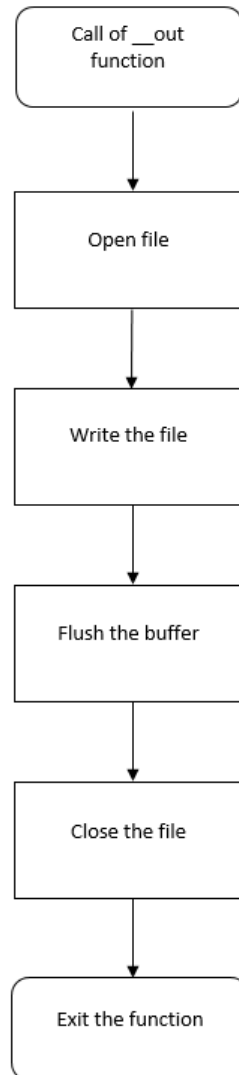


Figure 33. `__out` method flowchart.

Method `__out` uses the “*with open*” expression to open and close the file. This expression is widely used to interact with object files due to its simplicity and ease of use. This is the reason why this statement has been used for both write the “Blescan.log” file and load/write the “config.json” file.

Moreover, “*with open*” statement adds a plus: there is no need to explicitly close the file. It closes it automatically when the code is done with the file.



---

The information of how to use it was found in “Python for Beginners” website [61].

The open mode chosen for the “with open” statement was “a”. “a” opening mode opens the file for writing, the same as “w”. The difference is that “w” would truncate the file every time it is open and would add the text at the beginning of the file, while append opens the file and “appends” the text at the end of the file, conserving the previously saved. This is the best configuration mode for the log since its objective is to know all the old events and also the new ones. Information about the opening modes was found at a Stack Overflow web page thread [62]

After opening the file with the “with open” statement, *write()* method is called to save whatever is passed as a parameter to it. The codification chosen to write the file is “latin-1” after a search on the best option to type any character that could be saved in the log file. Use of latin-1 encoding was found in a StackOverflow webpage thread [63].

### **Exit\_gracefully method**

To allow the users to safely quit the program, it was needed to add a method that allows this functionality by pressing a letter on the keyboard.

*Exit\_gracefully* method allows the user to exit the program by typing “CTRL+C” in terminal by using a signal handler.

After that, the user will be asked “Really quit? (y/n)”. If the input is a “y”, the program will be finished.

Before completely closing the program, the method will clean both lists of beacons and already acknowledged devices. After that, “with open” statement will save that removal in the “config.json” file.

Once everything is saved, the file is closed and the program is exited by means of



---

“*sys.exit(1)*” method.

The idea of *exit\_gracefully* method is to exit the program with any type of errors and without crashing any of the external files. A search was made to find a proper solution and a thread provided it. The original method can be found at CodeReview website [64]. After testing it, it was modified to obtain the objective it has, which is the removal of the list of beacons in the “config.json” file.

### 3.3 Program files

All of these specifications are collected in the configuration file “config.json”. The mission of this configuration file is multiple: on the one hand, the file interacts with the code to load and dump the information regarding the lists of beacons and the acknowledged beacons. All the beacons added, updated and removed by the program are constantly interacting with this file.

On the other hand, the file is also used to save the configuration settings of the program, which are: the threshold power, the timeout and the list of users whose audio file is launched. All the instructions to configure these settings can be found in ANNEX I.

#### Config.json file

The previously mentioned configuration file is written in the JSON (JavaScript Object Notation). JSON is a standard for the interchange of data. The JSON is an easy and human-friendly notation to interact with objects. It is a text format widely used in programming languages as C, C++, Java, Python, and many more [65].

The reason why this type of format was chosen for the data to interact with the code is due to the way the file treats the information, which is really simple and can be used and modified by any user.

The structure of any JSON file is composed by lists and objects. Since Python works with objects, the use of JSON is a good option for the configuration file for the aforementioned specifications. It was only needed to create every object that interacts with the code and list their name and value. Next figure shows the structure of any generic object, composed by its name (the string) and the value between curly brackets. An example of the use of this structure is the users list, composed by the name “users”, the string, which is the UUID of the beacon, followed by the value, which, in this case, is the audio path.

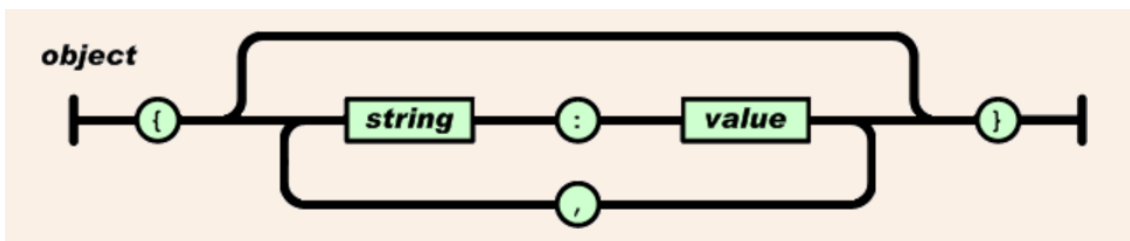


Figure 34. JSON generic object structure [65].

Arrays were also used in the JSON file for the beacons’ list and the already acknowledged list. In this case, the structure is the name of the array (“BeaconList”), followed by the values of the array and separated by commas and between brackets. Figure 35 shows the generic array structure layout for JSON files.

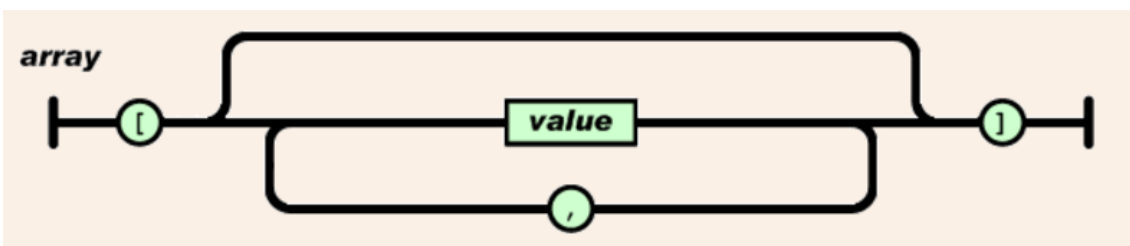
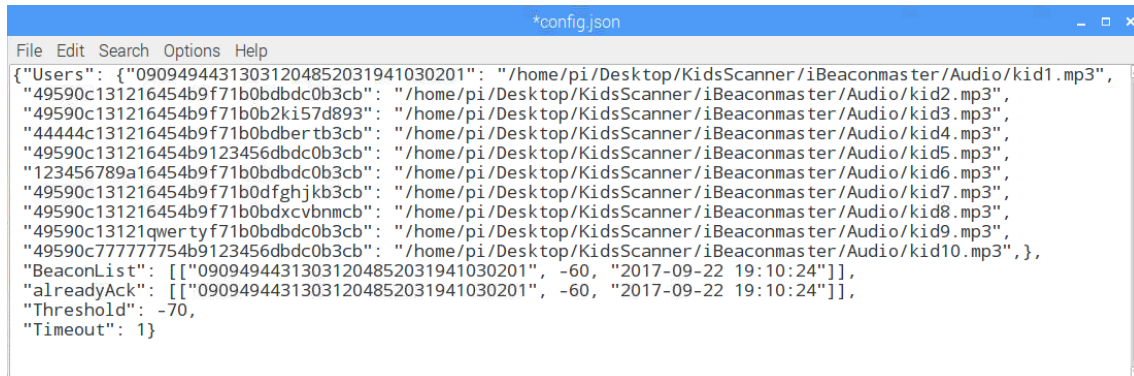


Figure 35. JSON generic array structure [65].

The “config.json” file is several times used in the code to load its information in the code (as in the constructor), and also to dump the beacons information into the file (e.g. the update of the list of beacons).

Figure 36 is an example of the configuration file layout.



```
File Edit Search Options Help
*config.json
{"Users": {"09094944313031204852031941030201": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid1.mp3",
"49590c131216454b9f71b0bdbdc0b3cb": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid2.mp3",
"49590c131216454b9f71b0b2ki57d893": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid3.mp3",
"44444c131216454b9f71b0bdbber tb3cb": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid4.mp3",
"49590c131216454b9123456dbdc0b3cb": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid5.mp3",
"123456789a16454b9f71b0bdbdc0b3cb": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid6.mp3",
"49590c131216454b9f71b0dfghjkb3cb": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid7.mp3",
"49590c131216454b9f71b0bdcvbnmcb": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid8.mp3",
"49590c13121qwer tyf71b0bdbdc0b3cb": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid9.mp3",
"49590c77777754b9123456dbdc0b3cb": "/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/kid10.mp3"},
"BeaconList": [{"09094944313031204852031941030201", -60, "2017-09-22 19:10:24"}],
"alreadyAck": [{"09094944313031204852031941030201", -60, "2017-09-22 19:10:24"}],
"Threshold": -70,
"Timeout": 1}
```

Figure 36. Configuration file layout example screenshot

It contains all the configurable elements and both the list of beacons and the list of already acknowledged beacons.

First, the list of users. As previously explained, the list of users is composed by the name of the object (users), followed by every element on the list: its string (the UUID) and its value (in this case, the value is the path of each custom voice message.) It can be seen that each different beacon has its own path for the .mp3 file.

Next element found on the image is the list of beacons. Every scanned beacon by the Raspberry is appended to that list (and removed or updated by the *UpdateBeaconList* method). The structure of the list is: ["UUID", RSSI, "date"].

The list of already acknowledged beacons follows the list of beacons. It works with the same structure as the list of beacons. As figure 36 shows, at the time of the screenshot, a beacon was detected and appended to the list of beacons. Moreover, the RSSI value of the beacon was over the threshold (-60dBm, threshold was set to -70dBm), so it was also appended to the list of already acknowledged beacons.

Threshold value is displayed in a simple way: name of the object and its value. As mentioned in the previous paragraph, for this specific example, the threshold value was set to -70dBm.



The last element in the configuration file is the timeout to be compared with the elapsed time. With the same structure as the threshold value, the timeout of this example was set to 1 minute. Of course, this lists and their objects have not predefined units, but the code is in charge of handling the data to match in units i.e. the timeout is considered to be in minutes, so the code is in charge to change the elapsed time to minutes as well to be properly compared.

These elements can be changed in the configuration file “config.json” with the instructions attached in ANNEX I.

### **Blescan.log**

To be able to check the correct functioning of the code, all the events are written in a log. This is done by calling the method `__out` in critical parts of the code (i.e. updating of the list of beacons, add and removal of beacons on the lists, `playWelcome` method).

This log does not really contribute any useful feature to end-users, instead, it helped a lot the developer to solve all the coding issues and also the calibration of timing.

Figure 37 shows a screenshot of the log of Case III.

```
BLEScan.log
File Edit Search Options Help
2017-09-19 17:16:53: BLE thread started
2017-09-19 17:17:03: Adding to beacon list UUID: 09094944313031204852031941030201 with signal strength: -65
2017-09-19 17:17:03: Detected new beacon in range UUID: 09094944313031204852031941030201 with signal strength: -65
2017-09-19 17:17:03: Playing welcome audio file: audio1.mp3for user: 09094944313031204852031941030201
2017-09-19 17:17:33: Updating element in beacon list UUID: 09094944313031204852031941030201
2017-09-19 17:18:03: Updating element in beacon list UUID: 09094944313031204852031941030201
2017-09-19 17:18:33: Updating element in beacon list UUID: 09094944313031204852031941030201
2017-09-19 17:19:03: Updating element in beacon list UUID: 09094944313031204852031941030201
2017-09-19 17:19:33: Updating element in beacon list UUID: 09094944313031204852031941030201
2017-09-19 17:20:03: Updating element in beacon list UUID: 09094944313031204852031941030201
2017-09-19 17:20:33: Adding to beacon list UUID: 8020244b03308306264b033083050100 with signal strength: -88
2017-09-19 17:21:03: Updating element in beacon list UUID: 8020244b03308306264b033083050100
2017-09-19 17:21:03: Removing element in beacon list UUID: 8020244b03308306264b033083050100
2017-09-19 17:21:03: Removing element in already ack list UUID: [u'09094944313031204852031941030201', -65, '2017-09-19 17:17:03']
```

Figure 37. Log of case III scenario example.

The beacon is acknowledged and, after the timeout and having left the scanning





region, it is removed. Otherwise, it would have shown a constant updating element in the list of beacons, meaning that the beacon is still in the range and its time is constantly being updated. In fact, this functionality can be also checked in the screenshot, since it continued updating its time until it left the scanning region.

This log can be found in the folder called “log”, inside the program root folder, “KidsScanner”.

### Music files

As it has been mentioned several times along the document, every kid carries a beacon with a unique UUID. In the configuration file, each UUID is paired to a custom audio file for each kid.

All the music files are saved in the folder “audios”, with a specific path. This path is copied to the list of users in the configuration file.

Every time the end user wants to change the audio file or path, must copy the new audio file path. ANNEX I, contains the information of how to do it.

Figure 38 shows the distribution of voice messages for each kid in the audio folder.

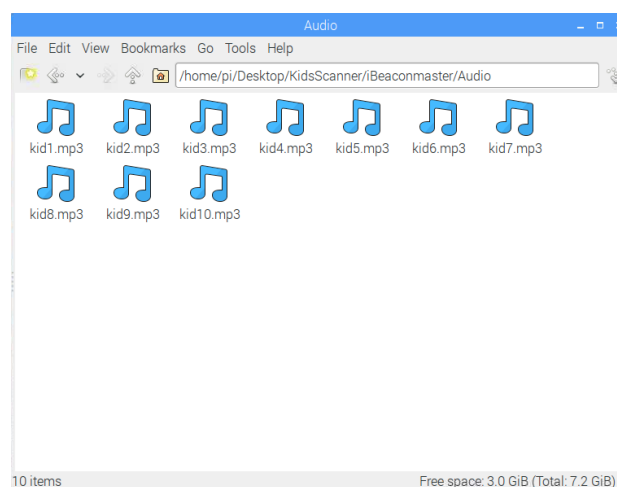


Figure 38. Audio files folder

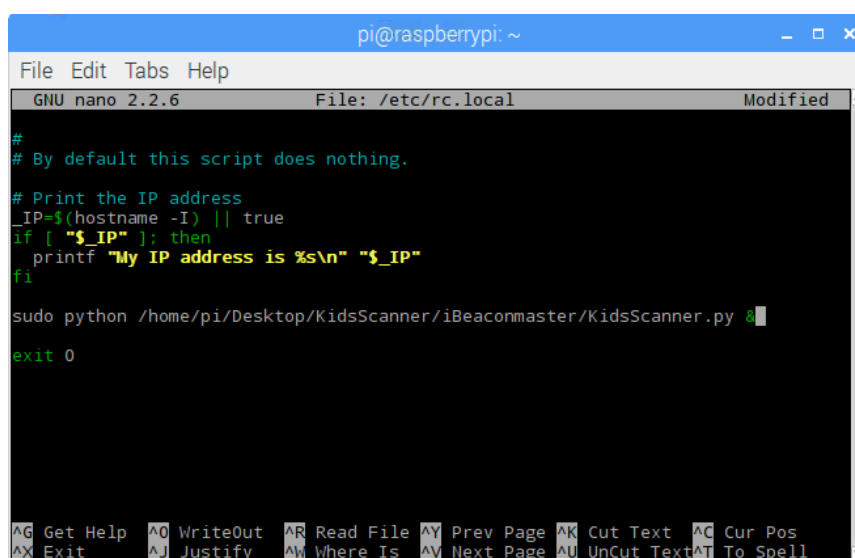
---

### 3.4 Program start-up

To start the program, the school will only need to connect the power supply to the Raspberry. By doing that, the Raspberry will start and the code will automatically start running once the Operating System is loaded and the Desktop is shown.

To do that, the Raspberry Website [66] offers a really simple solution.

It consist in adding the normal command needed to execute the program via terminal in a file called “rc.local”. By adding the running command to it, the program will be executed when starting the Raspberry.



```
pi@raspberrypi: ~  
File Edit Tabs Help  
GNU nano 2.2.6 File: /etc/rc.local Modified  
#  
# By default this script does nothing.  
# Print the IP address  
_IP=$(hostname -I) || true  
if [ "$_IP" ]; then  
  printf "My IP address is %s\n" "$_IP"  
fi  
  
sudo python /home/pi/Desktop/KidsScanner/iBeaconmaster/KidsScanner.py &  
  
exit 0  
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos  
^X Exit ^J Justify ^W Where Is ^N Next Page ^U UnCut Text ^T To Spell
```

Figure 39. rc.local file modified to start KidsScanner.py

As it can be seen in figure 39, the command line to execute “KidsScanner.py” normally is added to the file. This file is able to execute any terminal command by only asking two conditions to be fulfilled.

One of them is to add an “&” at the end of the line of the command in case the program is an infinite loop that could be running continuously. This will allow the process to start in parallel with the rest of the processes. Otherwise, the program would be locking



---

the start-up of the rest of the Raspberry indefinitely.

The other condition is to type everything above the “exit 0” line, which must be always at the end.

## 3.5 Code specifications

This code may be used for several different applications and scenarios, thus, there are some specifications that must be assigned for every case. As a part of the final device implementation, this chapter is intended to provide the information related to them and some guidance of recommended values.

### 3.4.1 Timeout

#### Concept

The timeout characteristic is a value in time units which is used in this project to evaluate if a beacon has definitely left a room. As the code works, every time the scanner detects a beacon, it updates the time of the beacon detection and it is registered in the configuration file.

In every loop of the code, the method *Update Beacon list* makes the difference between the current time and the last updated time of the beacon. As a result, the elapsed time is obtained, which is the time the beacon has not refreshed its scanning.

This elapsed time is compared with the timeout. The timeout is a configurable object of the code which helps to decide whether the kid left the room or not.

As mentioned in previous chapters, the elapsed time is used to remove the UUID beacon of the already acknowledged beacons list, in case that it is greater than the timeout. As a consequence, next time the beacons power transmission goes over the threshold



---

value, the voice message will be played again.

## Value

The selection of the timeout value may vary depending on the size of the room and also on the position of the central device (the Raspberry) inside it. Some daily scenarios must be also considered, such as the case in which the kids leave the room but suddenly come back for any reason (e.g. going to the toilet, but the activity in the room did not finish yet).

Since the project establishes a general code, implementable in any room, 15 minutes is a recommended value to cover these types of situations, and also unexpected events related with a possible eventual collapse of the Raspberry for a certain time, in which the scanner would not count the time and the loop would interpret that the beacon has left the room.

## 3.4.2 Threshold

### Concept

As previously described in the beacons description (*2.1.2 List of Components*), the RSSI value is the measure of the received signal power. In this project, RSSI is a vital measurement since the code actions are mainly based on its value i.e. the value of the RSSI of the kids' beacons determines if the custom voice message must be played or not.

In order to determine if the value is strong enough to consider that the kid is entering the room, a reference must exist. This reference is the threshold value. The threshold is one of the configurable characteristics of the system saved in the `config.json` file.



---

## Value

Since this system can be used and installed in every room the school teachers and caregivers would like, the threshold value may change depending on the distribution of the room. It is completely difficult and risky to establish a unique value for it. Although the system was thought and designed to be installed on the doors frame, some rooms may need the Raspberry to be installed in another different place. This can be done thanks to the modularity of all of the components by only changing the threshold value of the Raspberry depending on the room it is installed in.

As a reference value for the doors' frame at a height of 1.70m, the recommended threshold value is -60dBm.

As a solution approach for the possibility of setting the system in any room, a study of the power vs. distance was made to obtain the best threshold value with respect to the distance.

Table 7 summarizes the main values that can be used as threshold.

Distance(cm)	Threshold(dBm)
<50	25-40
50-60	40-55
60-80	50-70
80-120	60-80
120-200	70-90

Table 7. Threshold values for different distances

Although the values of table 7 were obtained from testing the scanner, it can be seen



that they are not unique values. For that reason, every scanner that should be tested in the specific room where it is going to be installed since the distribution of it may cause the variation of the values. Values from 200cm and longer are not stable and the detection of the beacon is not guaranteed.

Moreover, the values also depend on the quality of the beacon. In the case of the proposed system, beacons are low cost, so robustness is limited. Tests with cell phones as a peripheral device (beacon) concluded that with a better quality of BLE, the results are more stable and reliable.

### 3.5. Final budget

After having purchased all the necessary components, the final budget seems to fulfill the low cost objective. In fact, it has been improved with respect to the first approach made, which was 118€.

Device	Unitary price	Units	Total price
Raspberry Pi	39,95€	1	39,95€
Beacons	2,90€	10	29,00€
Speakers	9,90€	1	9,90€
Power source	8,99€	1	8,99€
Micro SD	8,99€	1	8,99€
Button cells	1,00€	10	10,00€
<b>Total price</b>	-	-	<b>106,83€</b>

Table 8.Final system budget.

Although this project is part of an educational project and process, salary is not



---

actually included as a part of the project.

Nevertheless, also as a part of any project, the engineering salary per hour must be included as a part of any budget.

Taking the salary per hour of an Engineer of Universidad Carlos III of Madrid as a reference, 26€ (included taxes, social security and unemployment), the total revenues for the engineer would be as described in table 9.

Personnel	Unitary price	Units	Total price
Engineer	26€/hour	350	9.100,00€

Table 9. Engineering costs

By considering the total budget as the material costs and the engineer salary, table 10 is shown:

<b>Engineer cost</b>	<b>9.100,00€</b>
<b>Material cost</b>	<b>106,83€</b>
<b>Total cost</b>	<b>9.206,83€</b>

Table 10. Total budget of the project



# CHAPTER 4

## Conclusions and future directions

This chapter summarizes the conclusions obtained at the end of the project and intends to make some possible proposals for future improvements of the system.





---

## 4.1 Conclusions

The objectives of the project have been fulfilled not only in terms of designing and developing a complete functional system as required by the school, but also by means of building the system according with the “Design for All” methodology and principles.

As a “Designed for All” system, it can be perfectly used by any person carrying a beacon. No matter their abilities or physical conditions. As mentioned at the beginning of the document: “fundamental for 10% of the population, necessary for the 40% of the population, and comfortable for the 100% of it”.

The system is easy to use and understand. The interaction needed to make it work is minimum: only a supervisor should modify the configuration file in case new beacons are added and, for that, a simple and short user’s manual is provided. Under normal conditions, the system does not need any interaction to work.

The system has been designed always taking into account that neither hazardous nor distracting elements would be part of it. As a result, the components of the developed system are as discreet as possible: raspberry attached to the wall at a sufficient height to not be reachable by the kids and beacons inside their bags which emit BLE silently.

Linked with the previous principle, the size of the school and the elements have been always considered. No big elements appear in this project, what is a must due to the necessity of space in the school (especially at the entrance of the rooms) for wheelchairs and the rest of assistive elements the kids possess and need to move.

The requirement of a low cost system has also been fulfilled satisfactorily, considering that the total price is not more than 120€, a really low price for a complete (software and hardware) system of presence detection with respect to the available in market solutions. Part of the success in this objective is thanks to the use of a non-commercial, open source software, combined with a complete, really low cost



---

motherboard, which made the price of the system much cheaper in terms of licenses.

From the educational part, another objective has been reached for this project as a Bachelor's degree final thesis. A necessity was proposed by real people, in a real environment with several conditions and requirements. As a response, a real and functional system has been developed in terms not only of the specific Industrial Electronics and Automation Engineering, by developing a system with hardware and software, but also applying the tools, general and specific competencies, abilities and knowledge that have been acquired along the bachelor as an Engineering degree in terms of receiving a problem, and providing a solution with the available resources.

Linking with the previous conclusion to finish, the fact of developing a complete device which is usable and adapted for the kids from San Rafael School is more than a satisfaction. It has been a pleasure to be part of the help they need. Because they do deserve it.

## 4.2. Future directions

One of the greatest features of the project is the central device used for it: the Raspberry Pi has countless uses that can be applied as future directions of this project. If wanted, this project could never end thanks to the wide range of possibilities the motherboard offers.

As already mentioned in the Raspberry section (*The Central device: Raspberry Pi 3*, page 33), the Raspberry can be used to add many new features and functionalities.

As a few examples of new uses of the Raspberry, it could be used as a multimedia center for the kids by simply connecting a monitor through the HDMI connector. Some educational lessons and games could be displayed.

By using the camera connector of the Raspberry (or even connecting an USB camera



---

through one of the 4 available ports), the Raspberry could be used as a game station, similar to computer vision camera based games (such as Kinect). Kids could develop their ability to move and to interact with the surroundings with the proper software (e.g. an adapted game in which kids are detected by the camera). What is more, as the system is already adapted with speakers, they could also contribute to the interactive game.

Another application that could be exploited is the Internet of Things feature. By improving the already existing software the Raspberry is equipped with (the Kids Scanner), the Raspberry could be connected to internet (without the need of cables, Raspberry Pi 3 already has a built-in wireless adapter). By connecting it to the internet, the beacons' detector could somehow load the information on the internet (in the cloud) and parents could see how their children interact and move inside the school.

In terms of improving this project, there are several ways of developing a more complex and robust system. From the hardware part, the Raspberry could be adapted with the speakers by using the GPIO pins of its header and a typical analog speakers commonly used in Electronics projects.

Also, some Electronics could be added to switch on/off the Raspberry properly, instead of disconnecting it directly from the power supply.

From the software part, the code could be complemented with some new features to make it more precise. As an idea, a second upper threshold could be implemented in the code to work as a comparator with Hysteresis. In other words, a Schmitt trigger of power transmission. The actual code has a threshold that, once exceeded, the beacons is added to the list of acknowledged beacons and will be only forgotten when it disappears from the list of beacons (i.e. it completely disappears from the scanning region).

By setting the proposed Schmitt trigger, the beacon could be removed from the list of already acknowledged beacons even if they still are in the scanning region. This would



---

cover some situations in which the kids may leave the room but to go to an adjoining room. In this case, the beacon could still be close enough to the scanner and continue detecting the kid, but he definitely left. Then, if he came back, the detector would not play the voice message.

With the proper calibration, a second upper threshold would consider the scanning of the beacons far enough to have left the room, but still on range.

It needs to be mentioned that this option is not possible with the current systems due to the fact that the robustness of the low cost beacons is limited, making this feature to be really difficult to calibrate and unstable.





---

## References

- [1] "World Health Organisation," [Online]. Available: <http://www.who.int/mediacentre/factsheets/fs352/en/>. [Accessed 18 07 2017].
- [2] «DesingForAll,» [En línea]. Available: <http://designforall.org/index.php>. [Último acceso: 22 09 2017].
- [3] «Boletín Oficial del Estado, N.289, 03/12/2003 (Chapter I, article 2, point d),» [En línea]. Available: <https://www.boe.es/boe/dias/2003/12/03/pdfs/A43187-43195.pdf>. [Último acceso: 27 07 2017].
- [4] «Design For All,» [En línea]. Available: <http://designforall.org/design.php>. [Último acceso: 19 07 2017].
- [5] «Universal Design,» [En línea]. Available: <http://universaldesign.ie/What-is-Universal-Design/The-7-Principles/>. [Último acceso: 19 07 2017].
- [6] «NMEDA, Automobile Mobility Solutions,» [En línea]. Available: <http://www.nmeda.com/>. [Último acceso: 22 09 2017].
- [7] C. Vázquez Ramos, *Transición a la vida adulta - Diana Adaptada*, 2016, p. 13.
- [8] «Cadenaser,» [En línea]. Available: [http://cadenaser.com/emisora/2017/04/20/radio\\_cordoba/1492705958\\_601663.html](http://cadenaser.com/emisora/2017/04/20/radio_cordoba/1492705958_601663.html). [Último acceso: 08 07 2017].
- [9] «Krify,» [En línea]. Available: <https://krify.co/tag/bluetooth-low-energy->



---

devices/. [Último acceso: 22 09 2017].

- [10] «Wikipedia,» [En línea]. Available: <https://es.wikipedia.org/wiki/Bluetooth>. [Último acceso: 03 09 2017].
- [11] V. Goyal, "Bluetooth Smart - A Revolution for Low Power Connectivity," *RTC Magazine*, 2014.
- [12] R. Kapur, "What is the difference between Bluetooth 5, Bluetooth 4.2 and Bluetooth 2.0?," *EverythingRF*, 2017.
- [13] «Adafruit,» [En línea]. Available: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt>. [Último acceso: 18 09 2017].
- [14] «Slux,» [En línea]. Available: <https://www.slux.guru/li-bluetooth>. [Último acceso: 03 09 2017].
- [15] «ONSET,» [En línea]. Available: <http://www.onsetcomp.com/content/bluetooth-low-energy-closer-look>. [Último acceso: 22 09 2017].
- [16] J. Velasco, «¿En qué consiste Bluetooth LE?,» *Hipertextual*, 13 12 2013.
- [17] «Bluetooth,» [En línea]. Available: <https://www.bluetooth.com/specifications/gatt/generic-attributes-overview>. [Último acceso: 09 09 2017].
- [18] «Bluetooth,» [En línea]. Available: <https://www.bluetooth.com/specifications/gatt>. [Último acceso: 02 09 2017].
- [19] N. Rivera, "Qué es el Internet of Things y cómo cambiará nuestra vida en el



futuro," *Hipertextual*, 2015.

- [20] «Pimoroni,» [En línea]. Available: <https://shop.pimoroni.com/products/raspberry-pi-3>. [Último acceso: 22 09 2017].
- [21] «RS Components,» [En línea]. Available: <http://es.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/8968660/>. [Último acceso: 22 09 2017].
- [22] «RS Components,» [En línea]. Available: <http://es.rs-online.com/web/p/kits-de-desarrollo-de-procesador-y-microcontrolador/8326274/>. [Último acceso: 22 09 2017].
- [23] «Raspberry Pi,» [En línea]. Available: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. [Último acceso: 22 09 2017].
- [24] «Raspberry Pi,» [En línea]. Available: <https://www.raspberrypi.org/products/raspberry-pi-2-model-b/>. [Último acceso: 22 09 2017].
- [25] «Gimbal,» [En línea]. Available: <https://store.gimbal.com/collections/beacons/products/s10>. [Último acceso: 01 08 2017].
- [26] «DALTEC,» [En línea]. Available: <http://dal-tec.com/batteries/205-cr2032-lithium-battery-3v.html>. [Último acceso: 22 09 2017].
- [27] K. Townsend, «Introduction to Bluetooth Low Energy,» *Adafruit*, 2015.





- 
- [28] C. Cabello, «NOBBOT, 9 Usos Reales para Comprender Qué Son los Beacons,» 2016. [En línea]. Available: <http://www.nobbot.com/redes/9-usos-reales-comprender-los-beacons/>. [Último acceso: 11 08 2017].
- [29] G. Gruman, "Beyond iBeacons: 7 cool uses of beacons you may not expect," *InfoWorld*, 2014.
- [30] «QueCalidad,» [En línea]. Available: <http://site.quecalidad.com/shop/accesorios-a-tu-estilo/234-llavero-anti-perdida-con-gps.html>. [Último acceso: 24 08 2017].
- [31] «Wikipedia,» [En línea]. Available: [https://es.wikipedia.org/wiki/Direcci%C3%B3n\\_MAC](https://es.wikipedia.org/wiki/Direcci%C3%B3n_MAC). [Último acceso: 22 09 2017].
- [32] «Hipertextual,» [En línea]. Available: <https://hipertextual.com/archivo/2014/09/que-es-mac-address/>. [Último acceso: 22 09 2017].
- [33] «Kontakt,» [En línea]. Available: <https://support.kontakt.io/hc/en-gb/articles/201620741-iBeacon-Parameters-UUID-Major-and-Minor>. [Último acceso: 22 09 2017].
- [34] «Metageek,» [En línea]. Available: <http://www.metageek.com/training/resources/understanding-rssi.html>. [Último acceso: 22 09 2017].
- [35] «Wikipedia,» [En línea]. Available: [https://en.wikipedia.org/wiki/Received\\_signal\\_strength\\_indication](https://en.wikipedia.org/wiki/Received_signal_strength_indication). [Último acceso: 22 09 2017].



22 09 2017].

- [36] «Wikipedia,» [En línea]. Available: <https://es.wikipedia.org/wiki/Python>. [Último acceso: 03 08 2017].
- [37] «Tiobe,» [En línea]. Available: <https://www.tiobe.com/tiobe-index/>. [Último acceso: 03 08 2017].
- [38] «Wikipedia,» [En línea]. Available: [https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)). [Último acceso: 04 08 2017].
- [39] «¡Hola Pi! Empezando a programar en Raspberry Pi con Python,» *Fábrica Digital*, 2016.
- [40] «Carrefour,» [En línea]. Available: <https://www.carrefour.es/mini-altavoz-bluesky-bbts10bk-cvon-bluetooth-negro/2012972184/p>. [Último acceso: 26 09 2017].
- [41] «Electrocomponents,» [En línea]. Available: <http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf>. [Último acceso: 22 09 2017].
- [42] «Element14,» [En línea]. Available: The alternative to these type of portable chargers might be a commuted power supply. . [Último acceso: 23 09 2017].
- [43] «AcerOnline,» [En línea]. Available: <http://www.aceronline.es/ac-adapter-cargador-10w-compatible-5v-2a-micro-usb-eu-aca0075.html>. [Último acceso: 23 09 2017].



- 
- [44] «Aliexpress,» [En línea]. Available: <https://www.aliexpress.com/popular/5v-2a-110-adapter.html>. [Último acceso: 23 09 2017].
- [45] «Mediamarkt,» [En línea]. Available: <https://tiendas.mediamarkt.es/p/cargador-universal-para-movil-samsung-1288733#prodinfotabspec>. [Último acceso: 26 09 2017].
- [46] «SanDisk,» [En línea]. Available: <https://www.sandisk.es/home/memory-cards/microsd-cards/ultra-microsd-for-cameras>. [Último acceso: 23 09 2017].
- [47] «BarcodesInc,» [En línea]. Available: <https://www.barcodesinc.com/info/buying-guides/rfid.htm>. [Último acceso: 27 08 2017].
- [48] F. Thronton, B. Haines, A. M.Das, H. Bhargava, A. Campbell and Kleinschmidt, RFID Security, Syngress Publishing, Inc, 2006, pp. 14-15.
- [49] «EmmeShop Electronics,» [En línea]. Available: <http://www.blog.emmeshop.eu/node/19>. [Último acceso: 22 09 2017].
- [50] «Wikipedia,» [En línea]. Available: [https://en.wikipedia.org/wiki/Radio-frequency\\_identification#Readers](https://en.wikipedia.org/wiki/Radio-frequency_identification#Readers). [Último acceso: 25 07 2017].
- [51] D. Paret, RFID and Contactless Smart Card Applications, 1st edition ed., Wiley, 2005, pp. 161-166.
- [52] «Wikipedia,» [En línea]. Available: [https://en.wikipedia.org/wiki/Radio-frequency\\_identification#Frequencies](https://en.wikipedia.org/wiki/Radio-frequency_identification#Frequencies). [Último acceso: 25 07 2017].
- [53] M. Brown, E. Zeisel y R. Sabella, RFID+ Exam cram, Pearson, 2006.



- 
- [54] «E-scan,» [En línea]. Available: <http://www.e-scan.com/smart-card/mifare-rfid.htm>. [Último acceso: 22 07 2017].
- [55] A. Bridden, ZERO to RFID in 60 minutes challenge?, 2016.
- [56] «Raspberry,» [En línea]. Available: <https://www.raspberrypi.org/downloads/raspbian/>. [Último acceso: 18 09 2017].
- [57] «Adafruit,» [En línea]. Available: <https://learn.adafruit.com/install-bluez-on-the-raspberry-pi/installation>. [Último acceso: 23 09 2017].
- [58] «StackOverflow,» [En línea]. Available: <https://stackoverflow.com/questions/625083/python-init-and-self-what-do-they-do>. [Último acceso: 14 08 2017].
- [59] «Pygame,» [En línea]. Available: <http://www.pygame.org/wiki/about>. [Último acceso: 21 09 2017].
- [60] «Pygame,» [En línea]. Available: <https://www.pygame.org/docs/ref/mixer.html>. [Último acceso: 27 08 2017].
- [61] «Python for Beginners - Reading and writing files in Python,» [En línea]. Available: <http://www.pythonforbeginners.com/files/reading-and-writing-files-in-python>. [Último acceso: 09 09 2017].
- [62] «Stack Overflow,» [En línea]. Available: <https://stackoverflow.com/questions/1466000/python-open-built-in-function-difference-between-modes-a-a-w-w-and-r>. [Último acceso: 10 09 2017].
- [63] «StackOverflow,» [En línea]. Available:



---

<https://stackoverflow.com/questions/4299802/python-convert-string-from-utf-8-to-latin-1>. [Último acceso: 10 09 2017].

[64] «Code Review,» [En línea]. Available: <https://codereview.stackexchange.com/questions/54348/handling-signals-in-python-inside-a-function>. [Último acceso: 12 09 2017].

[65] «JSON,» [En línea]. Available: <http://www.json.org/json-es.html>. [Último acceso: 04 09 2017].

[66] «Raspberry Pi Documentation.,» [En línea]. Available: <https://www.raspberrypi.org/documentation/linux/usage/rc-local.md>. [Último acceso: 25 09 2017].

[67] «MarketingLand,» [En línea]. Available: <https://marketingland.com/yext-launches-xone-beacon-network-for-offline-to-online-retargeting-144502>. [Último acceso: 05 09 2017].



## ANNEXS



## **ANNEX I. User's manual**



---

## User's manual (English version)

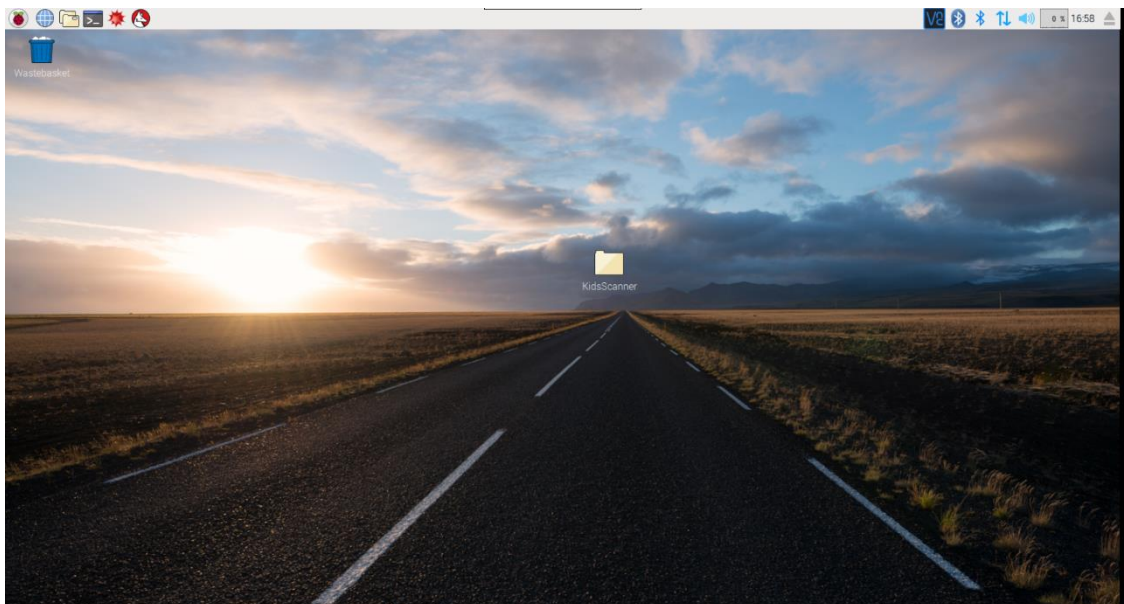
To modify the configuration file and/or the audio files, you will need:

- A monitor
- HDMI cable
- A mouse
- A keyboard

Connect the Raspberry Pi to the monitor through the HDMI connector and connect the power supply to the power socket.

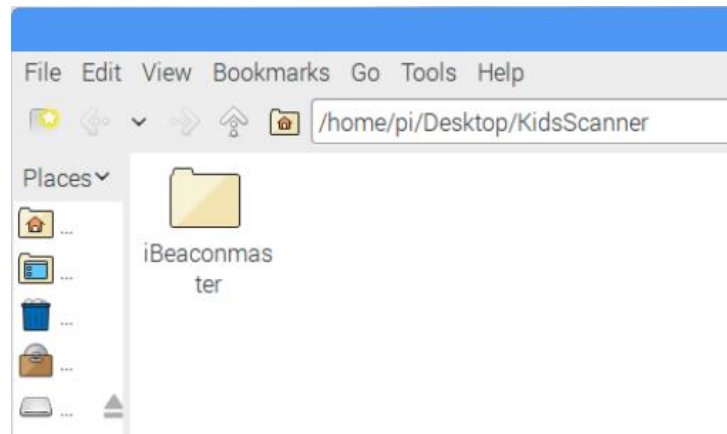
### a. How to modify the configuration file.

1. After powering the Raspberry and connecting the HDMI cable to the monitor, the Desktop will show up.

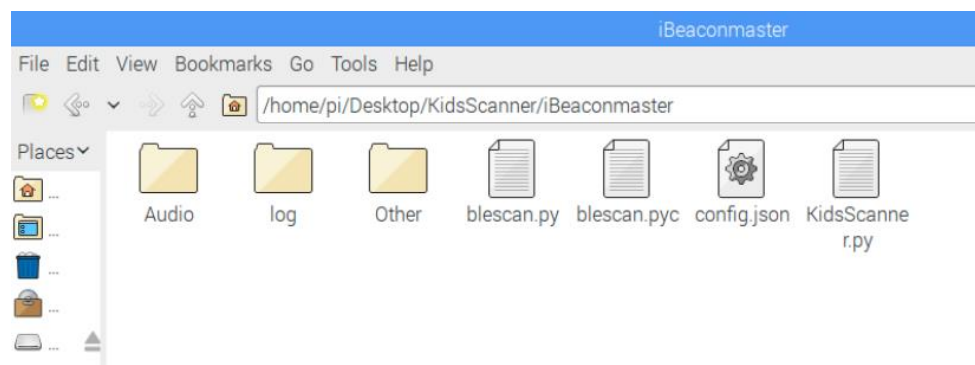




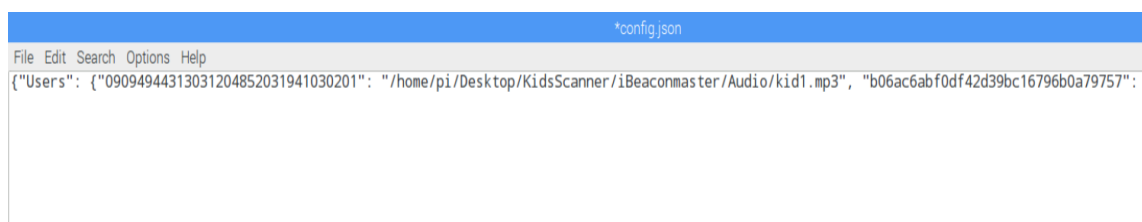
2. Click on KidsScanner folder to open it.



3. Click on “iBeaconmaster” folder to open it.



4. Click on “*config.json*” file to open it.



To **add new beacons**, just append new ones at the front or the back of any of the already existing ones. It is important to follow the following structure for the beacons to be recognized:

“**UUID**”: “/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/**new\_file.mp3**”,



---

Bold characters are the ones that have to be added.

A good way to append new beacons is to copy the structure of an existing one and paste it. Then, modify the UUID and the name of the audio file.

When adding a new beacon, an audio file is also needed for it. See how in point b – How to add audio files to the folder.

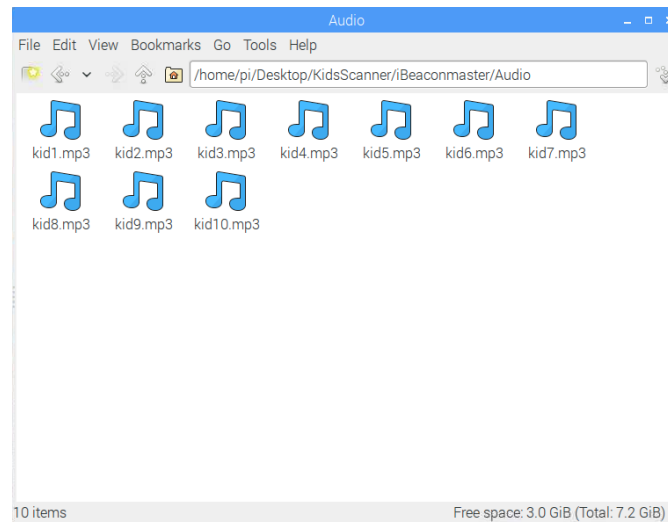
To **modify the threshold value and/or the timeout**, scroll the “*config.json*” file to the right until “timeout: X”, “Threshold: -Y” are reached.

You will only need to remove the numerical values and type the new ones. Please bear in mind that the timeout is in minutes, and the threshold value is a negative number, meaning that the closer to zero, the more restrictive the scanner will be (the beacon will have to be closer to the scanner to be acknowledged).

It is really important to **leave the structure** of the “*config.json*” file as it was before opening it, otherwise, **the program** will not be able to recognize it and **will collapse**.

#### **b. How to add audio files to the folder**

1. Go to */home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/*  
(Same procedure as in how to change configuration file, but opening the audio folder).
2. Drag the audio file to the folder from its location.



3. Go to the configuration file and add/modify the path of the new audio to the beacon (see how in point a - How to modify the configuration file).



---

## Manual de Usuario (Versión en castellano)

Para modificar el archivo de configuración y/o los archivos de audio, necesitarás:

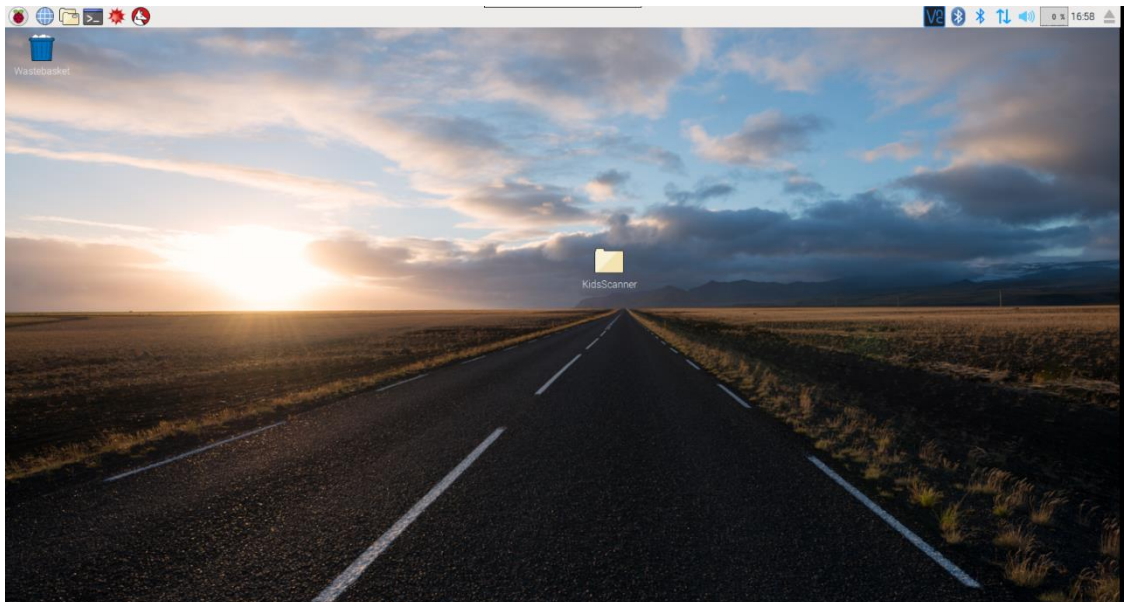
- Una pantalla
- Cable HDMI
- Un ratón
- Un teclado

Conecta el ratón, el teclado y el HDMI a la Raspberry.

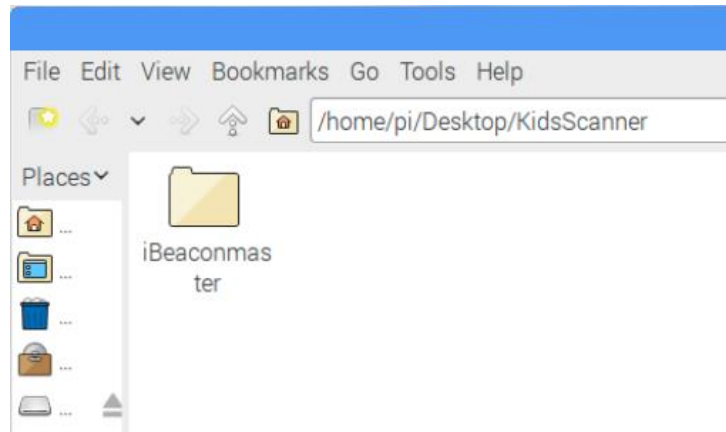
Conecta el HDMI al monitor y el cargador de la Raspberry a un enchufe.

### a. Cómo modificar el archivo de configuración.

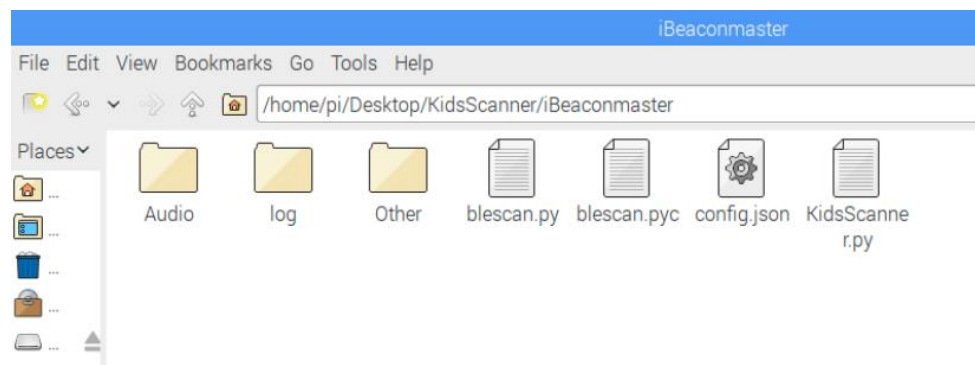
1. Después de encender la Raspberry (se enciende automáticamente al conectarla al enchufe) y conectar el HDMI al monitor, aparecerá el escritorio.



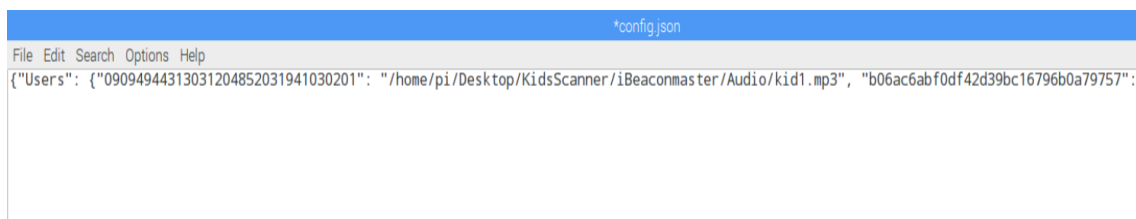
2. Haz “click” en la carpeta “KidsScanner” para abrirla.



3. Haz click en “iBeaconmaster” para abrirla.



4. Haz click en el archivo “*config.json*” para abrirlo.



Para añadir nuevos beacons, simplemente añade el nuevo delante o detrás de cualquier beacon existente. Es importante calcar la siguiente estructura para que los beacons sean reconocidos:



---

“**UUID**”: “/home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/new\_file.mp3”,

Los caracteres en negrita son los que deben ser modificados en la estructura.

Una buena forma de añadir beacons es copiar la estructura de algún beacon existente en la lista y pegarlo. Después, simplemente modifica los campos de UUID y el nombre del archive de audio.

Cada vez que se añada un nuevo beacon, debe ser asociado con un archivo de audio para ese beacon (Ve cómo en el punto b – Cómo añadir archivos de audio a su carpeta).

Para modificar el valor umbral (threshold) y/o el tiempo de expiración (timeout), desplaza la barra del archivo hacia la derecha hasta alcanzar “timeout: X”, “Threshold: -Y”.



Tan solo es necesario borrar los valores numéricos y escribir los nuevos. Por favor, ten en cuenta que el tiempo de expiración está en minutos, y que el valor umbral es un número negativo, lo que implica que cuanto más se acerque a cero, hará más restrictivo al escáner (el beacon tendrá que estar más cerca para ser reconocido en la lista de beacons reconocidos por encima del valor umbral).

Es muy importante **dejar la estructura** del archivo “*config.json*” tal y como estaba antes de abrirlo. De no ser así, **el programa** no reconocerá el archivo y **se corromperá**.

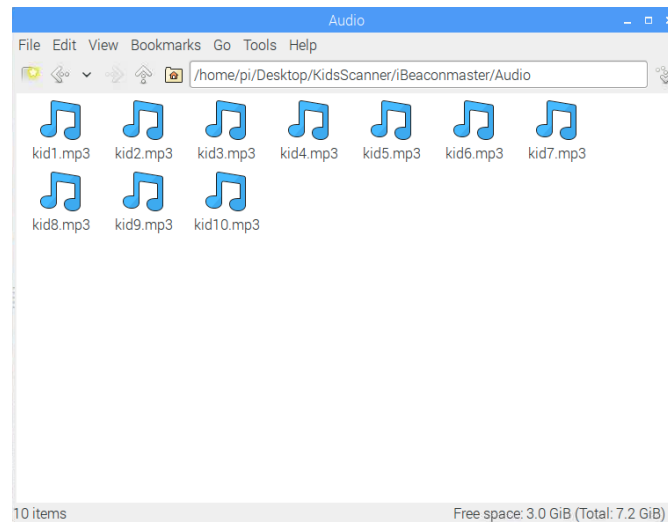
## **b. Cómo añadir archivos de audio a la carpeta.**

1. Ve a /home/pi/Desktop/KidsScanner/iBeaconmaster/Audio/

---

(Mismo procedimiento que al modificar el archivo de configuración, pero abriendo la carpeta de audios).

2. Arrastra el archivo de audio a la carpeta desde su ubicación.



3. Ve al archivo de configuración y añade/modifica la ruta del nuevo audio al beacon (ve cómo en el punto a – Cómo modificar el archivo de configuración).



## **ANNEX II. Blescan.py code**





---

```
# BLE iBeaconScanner based on https://github.com/adamf/BLE/blob/master/ble-scanner.py
# JCS 06/07/14
DEBUG = False
# BLE scanner based on https://github.com/adamf/BLE/blob/master/ble-scanner.py
# BLE scanner, based on https://code.google.com/p/pybluez/source/browse/trunk/examples/advanced/inquiry-with-rssi.py
# https://github.com/pauloborges/bluez/blob/master/tools/hcitool.c for lescan
# https://kernel.googlesource.com/pub/scm/bluetooth/bluez/+5.6/lib/hci.h for opcodes
# https://github.com/pauloborges/bluez/blob/master/lib/hci.c#L2782 for methods used by lescan
# performs a simple device inquiry, and returns a list of ble advertizements
# discovered device
# NOTE: Python's struct.pack() will add padding bytes unless you make the endianness explicit. Little endian
# should be used for BLE. Always start a struct.pack() format string with "<"

import os
import sys
import struct
import bluetooth._bluetooth as bluez

LE_META_EVENT = 0x3e
LE_PUBLIC_ADDRESS=0x00
LE_RANDOM_ADDRESS=0x01
LE_SET_SCAN_PARAMETERS_CP_SIZE=7
OGF_LE_CTL=0x08
OCF_LE_SET_SCAN_PARAMETERS=0x000B
OCF_LE_SET_SCAN_ENABLE=0x000C
OCF_LE_CREATE_CONN=0x000D
LE_ROLE_MASTER = 0x00
LE_ROLE_SLAVE = 0x01
# these are actually subevents of LE_META_EVENT
EVT_LE_CONN_COMPLETE=0x01
EVT_LE_ADVERTISING_REPORT=0x02
EVT_LE_CONN_UPDATE_COMPLETE=0x03
EVT_LE_READ_REMOTE_USED_FEATURES_COMPLETE=0x04
# Advertisement event types
ADV_IND=0x00
ADV_DIRECT_IND=0x01
ADV_SCAN_IND=0x02
```



---

```
ADV_NONCONN_IND=0x03
ADV_SCAN_RSP=0x04
def returnnumberpacket(pkt):
    myInteger = 0
    multiple = 256
    for c in pkt:
        myInteger += struct.unpack("B",c)[0] * multiple
        multiple = 1
    return myInteger
def returnstringpacket(pkt):
    myString = "";
    for c in pkt:
        myString += "%02x" %struct.unpack("B",c)[0]
    return myString
def printpacket(pkt):
    for c in pkt:
        sys.stdout.write("%02x " % struct.unpack("B",c)[0])
def get_packed_bdaddr.bdaddr_string):
    packable_addr = []
    addr = bdaddr_string.split(':')
    addr.reverse()
    for b in addr:
        packable_addr.append(int(b, 16))
    return struct.pack("<BBBBBB", *packable_addr)
def packed_bdaddr_to_string.bdaddr_packed):
    return ':'.join("%02x"%i for i in struct.unpack("<BBBBBB", bdaddr_packed[:-1]))
def hci_enable_le_scan(sock):
    hci_toggle_le_scan(sock, 0x01)
def hci_disable_le_scan(sock):
    hci_toggle_le_scan(sock, 0x00)
def hci_toggle_le_scan(sock, enable):
# hci_le_set_scan_enable(dd, 0x01, filter_dup, 1000);
# memset(&scan_cp, 0, sizeof(scan_cp));
#uint8_t    enable;
#    uint8_t    filter_dup;
```



---

```
# scan_cp.enable = enable;
# scan_cp.filter_dup = filter_dup;
# memset(&rq, 0, sizeof(rq));
# rq.ogf = OGF_LE_CTL;
# rq.ocf = OCF_LE_SET_SCAN_ENABLE;
# rq.cparam = &scan_cp;
# rq.clen = LE_SET_SCAN_ENABLE_CP_SIZE;
# rq.rparam = &status;
# rq.rlen = 1;
# if (hci_send_req(dd, &rq, to) < 0)
#     return -1;

cmd_pkt = struct.pack("<BB", enable, 0x00)

bluez.hci_send_cmd(sock, OGF_LE_CTL, OCF_LE_SET_SCAN_ENABLE, cmd_pkt)
def hci_le_set_scan_parameters(sock):
    old_filter = sock.getsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, 14)
    SCAN_RANDOM = 0x01
    OWN_TYPE = SCAN_RANDOM
    SCAN_TYPE = 0x01
def parse_events(sock, loop_count=100):
    old_filter = sock.getsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, 14)
    # perform a device inquiry on bluetooth device #0
    # The inquiry should last 8 * 1.28 = 10.24 seconds
    # before the inquiry is performed, bluez should flush its cache of
    # previously discovered devices
    flt = bluez.hci_filter_new()
    bluez.hci_filter_all_events(flt)
    bluez.hci_filter_set_ptype(flt, bluez.HCI_EVENT_PKT)
    sock.setsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, flt )
    done = False
    results = []
    myFullList = []
    for i in range(0, loop_count):
        pkt = sock.recv(255)
        ptype, event, plen = struct.unpack("BBB", pkt[:3])
        #print "-----"
```

---



---

```
if event == bluez.EVT_INQUIRY_RESULT_WITH_RSSI:
    i =0

elif event == bluez.EVT_NUM_COMP_PKTS:
    i =0

elif event == bluez.EVT_DISCONN_COMPLETE:
    i =0

elif event == LE_META_EVENT:
    subevent, = struct.unpack("B", pkt[3])
    pkt = pkt[4:]
    if subevent == EVT_LE_CONN_COMPLETE:
        le_handle_connection_complete(pkt)
    elif subevent == EVT_LE_ADVERTISING_REPORT:
        #print "advertising report"
        num_reports = struct.unpack("B", pkt[0])[0]
        report_pkt_offset = 0
        for i in range(0, num_reports):
            if (DEBUG == True):
                print "-----"
            #print "\tfullpacket: ", printpacket(pkt)
            print "\tUDID: ", printpacket(pkt[report_pkt_offset -22: report_pkt_offset - 6])
            print "\tMAJOR: ", printpacket(pkt[report_pkt_offset -6: report_pkt_offset - 4])
            print "\tMINOR: ", printpacket(pkt[report_pkt_offset -4: report_pkt_offset - 2])
            print "\tMAC address: ", packed_bdaddr_to_string(pkt[report_pkt_offset + 3:report_pkt_offset + 9])
            # commented out - don't know what this byte is. It's NOT TXPower
            txpower, = struct.unpack("b", pkt[report_pkt_offset -2])
            print "\t(Unknown):", txpower
            rssi, = struct.unpack("b", pkt[report_pkt_offset -1])
            print "\tRSSI:", rssi
            # build the return string
            Adstring = packed_bdaddr_to_string(pkt[report_pkt_offset + 3:report_pkt_offset + 9])
            Adstring += ","
            Adstring += returnstringpacket(pkt[report_pkt_offset -22: report_pkt_offset - 6])
            Adstring += ","
            Adstring += "%i" % returnnumberpacket(pkt[report_pkt_offset -6: report_pkt_offset - 4])
            Adstring += ","
```

---



---

```
Adstring += "%i" % returnnumberpacket(pkt[report_pkt_offset -4: report_pkt_offset - 2])
Adstring += ","
Adstring += "%i" % struct.unpack("b", pkt[report_pkt_offset -2])
Adstring += ","
Adstring += "%i" % struct.unpack("b", pkt[report_pkt_offset -1])
#print "\tAdstring=", Adstring
myFullList.append(Adstring)

done = True

sock.setsockopt( bluez.SOL_HCI, bluez.HCI_FILTER, old_filter )

return myFullList
```



## **ANNEX III. Testblescan.py**



```
# test BLE Scanning software
# jcs 6/8/2014
import bleSCAN
import sys
import bluetooth._bluetooth as bluez
dev_id = 0
try:
    sock = bluez.hci_open_dev(dev_id)
    print "ble thread started"
except:
    print "error accessing bluetooth device..."
    sys.exit(1)
bleSCAN.hci_le_set_scan_parameters(sock)
bleSCAN.hci_enable_le_scan(sock)
while True:
    returnedList = bleSCAN.parse_events(sock, 10)
    print "-----"
    for beacon in returnedList:
        print beacon
```



## **ANNEX IV. KidsScanner.py**





---

```
# test BLE Scanning software
# jcs 6/8/2014
import bleSCAN
import sys
import signal
import bluetooth._bluetooth as bluez
import os
import datetime
import time
import json
import pygame

class testBLEScan():
    def __init__(self):
        self.dev_id = 0
        self.original_sigint = signal.getsignal(signal.SIGINT)
        signal.signal(signal.SIGINT, self.exit_gracefully)
        with open("/home/pi/Desktop/KidsScanner/iBeaconmaster/config.json") as data_file:
            self.config = json.load(data_file)
            self.alreadyAck = self.config["alreadyAck"]
            self.threshold = self.config["Threshold"]
            self.beaconList = self.config["BeaconList"]
            self.timeout = self.config["Timeout"]
            self.users = self.config["Users"]
    def scan(self):
        try:
            sock = bluez.hci_open_dev(self.dev_id)
            self.beaconlist = []
            self.alreadyAck = []
            print "BLE thread started"
            now = datetime.datetime.now()
            self.__out(str(now)[:19] + ": BLE thread started",
                "BLEScan.log")
        except:
            print "Error accessing bluetooth device..."
            now = datetime.datetime.now()
            self.__out(str(now)[:19] + ": Error accessing bluetooth device...",
                "BLEScan.log")
            sys.exit(1)
        bleSCAN.hci_le_set_scan_parameters(sock)
        bleSCAN.hci_enable_le_scan(sock)
        while True:
            returnedList = bleSCAN.parse_events(sock, 10)
```



---

```
#returnedList = [{ 'uuid': 'hola4', 'rssi': 85 }, { 'uuid': 'hola8', 'rssi': 88 } ] #testing purpose only
with open("/home/pi/Desktop/KidsScanner/iBeaconmaster/config.json") as data_file:
    self.config = json.load(data_file)
print "-----"
for beacon in returnedList:
    print beacon
    self.updateBeaconList(beacon)
    if beacon['rssi'] > self.threshold and beacon['uuid'] not in [x[0] for x in list(self.alreadyAck)]:
        now = datetime.datetime.now()
        self.__out(str(now)[:19] + ": Detected new beacon in range UUID: " + str(beacon['uuid']) + " with signal strength:
" + str(beacon['rssi'])),
                "BLEScan.log")
        print "Detected new beacon in range"
        self.alreadyAck.append([unicode(beacon['uuid']), beacon['rssi'], now.strftime("%Y-%m-%d %H:%M:%S")])
        self.playWelcome(beacon['uuid'])
    elif beacon['rssi'] > self.threshold and beacon['uuid'] in [x[0] for x in list(self.alreadyAck)]:
        print "Already detected above threshold"
        with open("/home/pi/Desktop/KidsScanner/iBeaconmaster/config.json", "w") as jsonFile:
            self.config["BeaconList"] = self.beaconList
            self.config["alreadyAck"] = self.alreadyAck
            json.dump(self.config, jsonFile)
            time.sleep(1)
def __out(self, text, filename):
    with open("/home/pi/Desktop/KidsScanner/iBeaconmaster/log/" + filename, "a") as f:
        f.write(text.encode("latin1") + "\n")
        f.flush()
def updateBeaconList(self, beacon):
    if beacon['uuid'] not in [x[0] for x in list(self.beaconList)]:
        print str(beacon['uuid']) + " not in beacon list"
        now = datetime.datetime.now()
        self.beaconList.append([unicode(beacon['uuid']), now.strftime("%Y-%m-%d %H:%M:%S")])
        print "Adding to beacon list..."
        self.__out(
            str(now)[:19] + ": Adding to beacon list UUID: " + str(beacon['uuid']) + " with signal strength: " + str(beacon['rssi'])),
            "BLEScan.log")
    elif beacon['uuid'] in [x[0] for x in list(self.beaconList)]:
        now = datetime.datetime.now()
        search = beacon['uuid']
        for sublist in self.beaconList:
            if sublist[0] == search:
                self.beaconList.remove(sublist)
                self.beaconList.append([unicode(beacon['uuid']), now.strftime("%Y-%m-%d %H:%M:%S")])
```

---



---

```
        self.__out(
        str(now)[:19] + ": Updating element in beacon list UUID: " + str(
            beacon['uuid'] ),
        "BLEScan.log")
for uuid,dtime in self.beaconList:
    atime = datetime.datetime.strptime(str(dtime), '%Y-%m-%d %H:%M:%S')
    now = datetime.datetime.now()
    elapsedtime = ((now - atime).total_seconds()) / 60
    if elapsedtime > self.timeout:
        self.beaconList.remove([uuid,dtime])
        self.__out(
            str(now)[:19] + ": Removing element in beacon list UUID: " + str(
                beacon['uuid']),
            "BLEScan.log")

#UPDATING ALREADY ACK LIST
for uuid in self.alreadyAck:
    if uuid[0] not in [x[0] for x in list(self.beaconList)]:
        #atime2 = datetime.datetime.strptime(str(uuid[2]), '%Y-%m-%d %H:%M:%S')
        #now = datetime.datetime.now()
        #elapsedtime2 = ((now - atime2).total_seconds()) / 60
        #if elapsedtime2 > self.timeout:
            self.alreadyAck.remove(uuid)
            print "Removing element in already ack list UUID: " + str(uuid)
            self.__out(
                str(now)[:19] + ": Removing element in already ack list UUID: " + str(
                    uuid),
                "BLEScan.log")
def playWelcome(self,user):
    if user in self.users:
        print "Playing welcome audio file " + str(self.users[user]) + " for user: " + str(user)
        pygame.mixer.init()
        pygame.mixer.music.load(str(self.users[user]))
        pygame.mixer.music.play()
        while pygame.mixer.music.get_busy() == True:
            continue
        now = datetime.datetime.now()
        self.__out(
            str(now)[:19] + ": Playing welcome audio file: " + str(self.users[user]) + "for user: " + str(user),
            "BLEScan.log")
def exit_gracefully(self,signum, frame):
    signal.signal(signal.SIGINT, self.original_sigint)
```



```
try:
    if raw_input("\nReally quit? (y/n)> ").lower().startswith('y'):
        self.__out("Removing beacons from list before exiting, Program closed.",
            "BLEScan.log")
        self.config["BeaconList"] =[] #Cleaning BeaconList
        self.config["alreadyAck"] =[] #self.alreadyAck
        with open("/home/pi/Desktop/KidsScanner/iBeaconmaster/config.json", "w") as jsonFile:
            json.dump(self.config, jsonFile)
        sys.exit(1)
except KeyboardInterrupt:
    print("Ok ok, quitting")
    sys.exit(1)
signal.signal(self.signal.SIGINT, self.exit_gracefully)
a = testBLEScan()
a.scan()
```