UNIVERSIDAD CARLOS III DE MADRID

PROTOTYPING A LOW-COST PRESENCE TRACKER FOR THE ELDER

Author: José Ignacio Torralba Álvarez
Supervisor: Dr. Daniel Díaz Sánchez

Bachelor's Degree in Telecommunication Technologies Engineering

Telematic Engineering Department

Leganés, September 2016

"Whether you think you can, or think you can't – you're right."

– Henry Ford

Acknowledgements

Before starting, a little bit of recognition is in order. I want to express my gratitude to everyone that contributed even a little to make the most of these years.

First of all, I would like to thank my supervisor, Dani, for helping me choose what to do and giving me advice whenever I needed it.

I want to thank my lifelong friends too. This includes obviously my other family and those who kept asking me about the project even though they did not understand a word I said. Thanks for helping me take my mind off stressful stuff and for all the good times.

Next, I am grateful to Jesús, Santi and Víctor. Even though we are not finishing all together, you have been there from the very beginning and I know I can count on all of you if I need to. I hope you know you can count on me too. Thank you for all the laughs we have shared and for the many more to come.

Also, I want to thank Irene. The last year would not have been the same without you. You made it much easier and it turned out to be an awesome one. Thanks for putting up with me and for being my go-to person whenever I had a problem or something bothered me.

Lastly, I want to thank my family. In particular my parents, José Luis and Mara, as well as my sister, Marta. You have always been there for me and I hope I will be able sometime to give back to you at least half as much as you have given to me.

Abstract

Bachelor's Degree in Telecommunication Technologies Engineering

Prototyping a low cost presence tracker for the elder

by José Ignacio Torralba Álvarez

Bluetooth Low Energy technology has established itself as a key driver for the Internet of Things. In this field there exist an infinite number of opportunities, and the project described in this document is just one of them.

In this work the focus was set in helping people who suffer from mental illnesses and their relatives. It intends to take advantage of this technology to implement a presence tracking system able to detect whenever they are not in their corresponding place and alert the one responsible in case it happens. An analysis of the feasibility of this system and the performance that could be expected from it is carried out.

In the present document topics such as Bluetooth and BLE specification, most common uses and its application in beacons among others are discussed.

Contents

List of Figures

List of Tables

1 INTRODUCTION AND GOALS

This chapter will present the reader with a brief overview of the project described in this report. The initial goals for this project will be discussed.

1.1 Motivation

The release of Bluetooth version 4.0, introducing Bluetooth Low Energy for the very first time, opened the doors to a whole new set of possibilities. This new technology could be used to broadcast small pieces of information to nearby devices, and soon the idea of using it for location services came up. Its low power consumption made it a great candidate for that purpose.

Systems taking advantage of this technology can be deployed in hospitals to identify patients and instantly get an updated version of their medical history and health status. In fact, these systems already exist. There are also services to keep track of hospital equipment, which is quite expensive, to avoid costly losses. They may as well serve to make sure patients do not leave their corresponding rooms, if they are not supposed to; and locate them in case they do [1].

For this project, this idea was taken and pointed in a slightly different direction. The focus was set at a home environment and in a familiar context, instead of large hospitals. Mental illnesses such as dementia or Alzheimer's disease are a huge challenge in many people's lives. Not only in those who suffer from them, but also on those who know and care about the former. People affected by these illnesses do not know what they are doing and it may be the case that they left their home, room or any other place they should be at without anyone knowing it. Their relatives face a lot of struggle with this fact.

These relatives could get some peace of mind if they knew for sure that the ones they care about were actually where they should be. So now the question is, would it be feasible to build a similar system to the one hospitals use at a reduced cost? And, would this hypothetical system be reliable?

1.2 Objectives

The main goal of this project is to study the possibility of providing a low-cost solution to the issue described in the previous section, by prototyping a presence tracking system taking advantage of BLE technology.

This system would send a notification whenever the *tracked* person left home. It would also inform in case the person stayed in the exact same spot for a long time, as it could be a bad sign; or if he or she got too close to any *dangerous* zones, such as stairs.

Another key point for this project is in the words ”low-cost”. It is important to make it affordable for most people. The system is thought as something to help sick people that may live by themselves, which might be due to not being able to afford someone to keep company. It is a good idea to provide them with a less expensive alternative to those systems hospitals use. But also to obtain something that would not imply a great loss in the case that the *tracked* person decided to throw it away or lost it somewhere.

Also, along this project it was intended to keep a business-minded focus, resulting in a final product that could be launched to the market. To achieve that, aside from actually working, it should be easy to use by anyone. It was important not to add any extra barrier that might prevent people from benefiting from it.

1.3 Social context

The increasing of life expectancy has made the appearance of age related illnesses more and more common. Around 25 million people worldwide have dementia, and the number of people suffering from it is expected to surpass 80 million by 2040 [2]. In Europe, one of every three citizens will be over 65 by 2060. By that year the ratio of working people to retired people will be two to one, a big change compared to the current four to one ratio [3].

New technologies have the potential to help with such issues, and plans to promote their adoption have been deployed. There are projects like the EU-funded CommonWell that aim for a better integration between healthcare and social care services with the help of ICT solutions. Four different services were implemented: better emergency care through telecare integration; managed hospital admission for care clients; early intervention and telehealth for Chronic Obstructive Pulmonary Disease patients; and integrated support for heart failure patients [4].

The project this document describes fits in this idea of putting technology to the service of humans. In a time when everybody is often in a hurry to get to a place or get something done, it strives to help easing a bit the already difficult lives of people with loved ones suffering from dementia or similar conditions.

1.4 Regulatory requirements

This project has to fulfill two main legal compliance requirements. The first one has to do with radio-frequency output power. Bluetooth technology transmits in the ISM unlicensed band, which is regulated by the Internet Engineering Task Force in Europe. The regulation sets a 20dBm (100mW) limit for adaptative frequency hopping equipment that the maximum output power can never exceed [5]. This requirement is completely fulfilled since it is specified in the Bluetooth Core Specification [6] and, thus, manufacturers must satisfy it in order to certify their products.

The second one is related to personal data protection. Positioning techniques are a way to obtain location data, so the data protection laws of each region must be taken into account.

1.5 Contents

This document consists of seven chapters.

- **Chapter 1: Introduction and goals**

  The first chapter covers a brief introduction to the problem, situating it in its social context. The objectives of the project are stated as well.

- **Chapter 2: State of the art**

  This chapter takes a deep look into the technology that will be used for building the system, Bluetooth Low Energy.

- **Chapter 3: Design**

  The study of the problem and given solution are covered in this chapter. It ends with a description of the hardware chosen to build the prototype.

- **Chapter 4: Implementation**

  This chapter goes through the actual building of the prototype. Describing the steps taken and decisions made to achieve the desired performance.

- **Chapter 5: Testing**

  In this chapter, the tests the prototype was subjected to are described. An analysis of the results obtained from these tests is made as well.

- **Chapter 6: Project history**

  This chapter includes the planning made to complete the project in time. Problems faced during the implementation of the system and how they were solved or avoided. Budget including all the costs associated to the project.

- **Chapter 7: Conclusions**

  Conclusions after the realization the project. An analysis of the goals set at the beginning is made, pointing out those fulfilled by the prototype. This chapter also includes some personal thoughts of the author regarding the making of this project. Also, some ideas for future improvements are given.

2 STATE OF THE ART

This chapter will present the reader with some of the basics of the chosen technology to drive this project, Bluetooth Low Energy. Key points from the specification will be introduced as well as the main differences with classic Bluetooth. The chapter will end with a brief discussion of what to expect of this technology in the coming years.

2.1 Bluetooth

Bluetooth is a wireless technology standard that allows the exchange of data among electronic devices over short distances. It was conceived as an alternative to data cables. It was invented in 1994 by Ericsson.

The Institute of Electric and Electronic Engineers, IEEE, standardized Bluetooth as IEEE 802.15.1, although this standard is no longer maintained.

The Bluetooth SIG (Special Interest Group), an organization formed by different companies, is the one in charge of overseeing new developments of Bluetooth technology. Nowadays there are over 20000 members, being the most active and with most influence Ericsson Technology Licensing, Intel, Lenovo, Microsoft, Apple, Nokia, Toshiba and IBM.

Figure 2.1: Bluetooth SIG logo. Retrieved from www.wikipedia.com

2.1.1 Bluetooth basics

Bluetooth operates at frequencies between 2400 and 2483.5 MHz taking into account guard bands 2 MHz wide at the bottom end and 3.5 MHz wide at the top. There are 79 channels spaced 1 MHz. This is in the globally unlicensed Industrial, Scientific and Medical (ISM) 2.4 GHz short-range radio frequency band. Bluetooth uses frequency-hopping spread spectrum to fight against interference and fading.

There exists two modulation schemes. Basic Rate uses a shaped, binary frequency modulation. It can achieve data rates up to 1 Mbps. Enhanced Data Rate can use either $\pi/4$-DQPSK (Differential Quadrature Phase Shift Keying) or 8DPSK modulation, achieving rates of 2 Mbps with the former and 3 Mbps with the latter.

Bluetooth is a packet-based protocol with a master-slave structure. One master may communicate with up to seven slaves in a piconet.

There are three power classes according to the output power [6] and hence approximate range.

Class number	Maximum power	Range
Class 1	100 mW	100 m
Class 2	2.5 mW	10 m
Class 3	1 mW	1 m

Table 2.1: Bluetooth classes

2.1.2 Device discovery and pairing

Three progressive states [7] must be followed to establish a Bluetooth connection between two devices:

1. Inquiry – If two Bluetooth devices know absolutely nothing about each other, one must run an inquiry to try to discover the other. One device sends out the inquiry request, and any device listening for such a request will respond with its address, and possibly its name and other information.

2. Paging (Connecting) – Paging is the process of forming a connection between two Bluetooth devices. Before this connection can be initiated, each device needs to know the address of the other (found in the inquiry process).

3. Connection – After a device has completed the paging process, it enters the connection state. While connected, a device can either be actively participating or it can be put into a low power sleep mode.

    - Active Mode – The device is actively transmitting or receiving data.
    - Sniff Mode – The device will sleep and only listen for transmissions at a set interval. It is a power-saving mode
    - Hold Mode – The device sleeps for a defined period and then returns back to active mode when that interval has passed.
    - Park Mode – When in this mode, the slave will become inactive until the master tells it to wake up.

2.1.3 Common uses of Bluetooth

Bluetooth technology has been introduced in several different markets [8]:

- Automotive:

  The first example that comes to mind in this field are Bluetooth enabled hands-free calling systems, which are now included as standard equipment on millions of new cars. However, Bluetooth technology is also used in apps that connect to the vehicle's audio system and displays to receive route or weather updates while driving. Manufacturers are starting to use it to monitor and diagnose mechanical and electrical systems, getting rid of wires. There are even studies being carried out to monitor a person's vital signs while driving

- Consumer electronics:

  Nowadays Bluetooth is basically everywhere. It found its way to audio streaming making a reality all the wireless headphones and speakers we have now. But it did not stop there, newest products include toothbrushes, light bulbs and lawnmowners for example.

- Home automation:

  This field includes all the Bluetooth sensors for temperature, lights, doors, windows, motion detection, and more that are being introduced in the smart homes. These sensors connect with devices that users are already familiar with, like a tablet or a smartphone, making its adoption easier.

- Medical and health:

  In the medical field there have been great improvements thanks to Bluetooth. Products like asthma inhalers, blood glucose monitors, pulse oximeters or heart rate monitors are becoming more and more common. They connect to the Bluetooth compatible PCs, tablets and smartphones users already own. This allows them to control their health on their own, without needing to constantly go to the doctor.

- Smartphones

Nowadays it is very unusual to find a mobile phone in the market without Bluetooth capabilities. It is used to provide connectivity with a wide variety of accessories. It started with wireless headphones and speakers, and now there are billions of Bluetooth products on the market that work with our phones.

- PC and peripherals:

  With the irruption of tablets and the trend to make everything thiner, Bluetooth enabled keyboards gained more and more popularity. Aside from keyboards, this technology is implemented in speakers, stereo headphones, and many other wireless computer accessories. All of them have the advantage that no extra cables are needed.

- Retail and location-based services:

  BLE beacons are being introduced in the stores. Some of their possible applications are in-store analytics, proximity marketing, indoor navigation and contactless payments.

- Sports and fitness:

  Fitness tracking bands and smart watches are becoming more popular among consumers. Small sensors in these devices track aspects of athletic performance such as steps, distance, pace, heart rate, and calories burned, and communicate that information to consumer-friendly mobile apps. Also, motion sensors are finding their way into equipment such as basketballs, golf clubs, softball, baseball bats... Bluetooth technology makes all that data accessible so it can be later analyzed and used to improve performance.

- Wearables:

  All wearables take advantage of Bluetooth technology. This is a market with a huge potential, which will grow rapidly in the coming years.

2.1.4 Bluetooth vs. other technologies

Aside from Bluetooth there are several other short-range technologies in the market, including RFID, IrDa, ZigBee or NFC. Each of these have their own features and work in a different way [9].

- RFID

  The receiver gets data from a passive tag by powering it wirelessly. Its most common use is inventory tracking.

- IrDa

  It is a point-to-point, ad-hoc, wireless transmission over infrared light. It is commonly used in remote control systems or for payments.

- ZigBee

  This technology offers a high range at low data rates and power consumption. It is targeted for remote control in the industrial field.

- NFC

  Based on RFID technology, NFC has low power consumption, range and data rate. Its main applications are payments and ticketing.

Table 2.2 shows a quick comparison among them and Bluetooth [10].

	Bluetooth	**RFID**	**IrDa**	**ZigBee**	**NFC**
Maximum Data Rate	2.1 Mbps	128 kbps	16 Mbps	250 kbps	424 kbps
Maximum Range	100 m (typically less than 10 m)	1 m	2 m	100 m (typically 10 m – 20 m)	0.1 m

Table 2.2: Short-range technologies comparison

2.2 Bluetooth Low Energy

Compared to Classic Bluetooth, BLE is intended to provide considerably reduced power consumption and cost while maintaining a similar communication range. Its power-efficiency makes it perfect for devices that run for long periods on power sources such as coin cell batteries.

When the Bluetooth SIG announced the formal adoption of Bluetooth® Core Specification version 4.0, it included the hallmark Bluetooth Smart (low energy) feature. This final step in the adoption process opened the door for qualification of all Bluetooth product types to version 4.0 and higher.

Bluetooth Smart (low energy) wireless technology features:

- Ultra-low peak, average and idle mode power consumption
- Ability to run for years on standard coin-cell batteries
- Low cost
- Multi-vendor interoperability
- Enhanced range

2.2.1 Main differences with classic Bluetooth

While Bluetooth focused on exchanging high volumes of data in a limited period of time, BLE attempted to make these exchanges in a quick and periodical way. The transmitted data would be in much smaller quantities but making it last longer. Some of the main changes are the following:

- BLE has 40 2 MHz wide channels where classic Bluetooth has 79 1 MHz wide.

- BLE supports an unlimited number of nodes, instead of just 7.
- The larger modulation index used in BLE (0.5 against 0.35), allows the range to be increased above 100 m.
- Application throughput is much lower in BLE.
- Authentication needs to be per packet.
- 24 bit CRC against the 8 or 16 bit CRC present in classic Bluetooth.

2.2.2 Architectural overview

The Link Layer operates as a state machine with five possible states:

- Standby State

  The Link Layer does not receive or transmit any packets. This state can be entered from any other state
- Advertising State

  The Link Layer is transmitting advertising channel packets and possibly listening to and responding to responses triggered by these advertising channel packets.
- Scanning State

  The Link Layer is listening for advertising channel packets from devices that are advertising.
- Initiating State

  The Link Layer is listening for advertising channel packets from a specific device(s) and responding to these packets to initiate a connection with another device.

- Connection State

  The Connection State can be entered either from the Initiating State or the Advertising State. Within this state, two roles are defined:

  – Master Role: when entered from the Initiating State. A device assuming this role will communicate with slave devices and will define the timings of transmissions

  – Slave Role: when entered from the Advertising State. A device in this role will communicate with a single device in the Master Role.

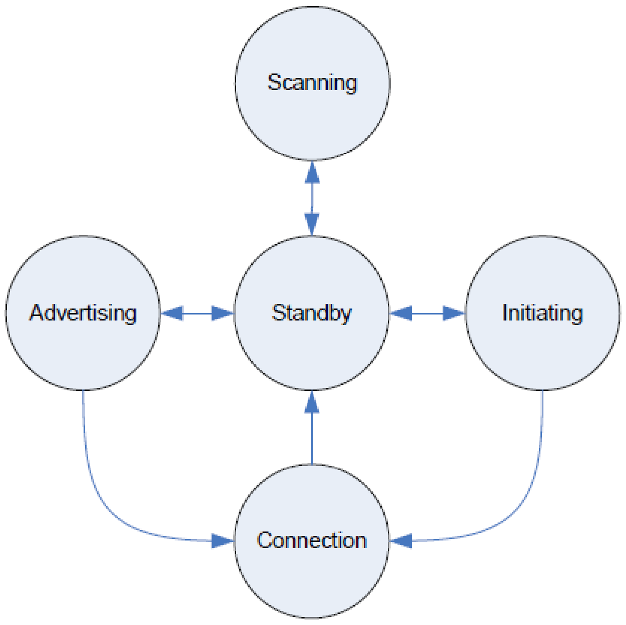


Figure 2.2: State diagram of the Link Layer state machine. Retrieved from www.bluetooth.com

2.2.3 BLE protocol stack

BLE protocol stack is very similar to that of classic Bluetooth. They both consist of two parts, the Controller and the Host, which can be connected by the Host Controller Interface. Despite this similarity, both types of controllers are not compatible. That is the reason why there exist the known as dual-mode devices. These implement both stacks, the classic Bluetooth one and the BLE one. Otherwise, a device with only the BLE stack, known as Bluetooth Smart devices, could not connect with a device implementing just the classic Bluetooth stack [11]. Dual-mode devices are also referred to as Bluetooth Smart Ready devices. Figure 2.3 shows a comparison among the protocol stacks of the three, while Figure 2.4 depicts the Bluetooth Low Energy or Smart stack.

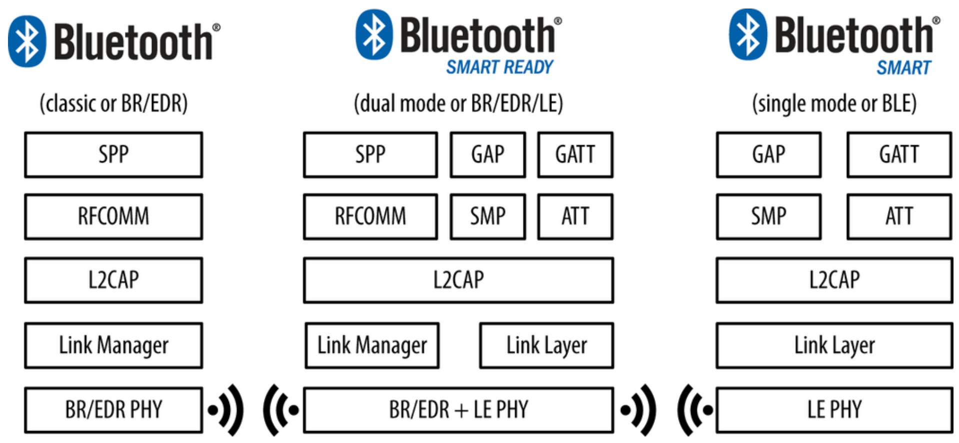


Figure 2.3: Bluetooth, Smart Ready and Smart protocol stacks. Retrieved from www.safaribooksonline.com

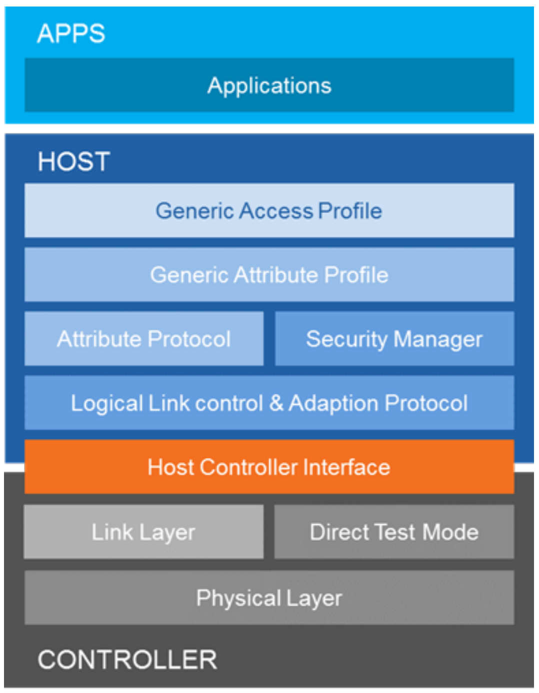


Figure 2.4: BLE stack. Retrieved from www.bluetooth.com

Controller

The controller is in charge of radio signal transmission and reception as well as connection management.

- Physical layer

  BLE transmits in the 2.4 GHz ISM band. There exists 40 RF channels 2 MHz wide, which can be either advertising or data channels. The former are used for device discovery, connection establishment and broadcast transmission, whereas the latter are used for bidirectional communication between connected devices.

  Channels 38, 39 and 40 are the three advertising channels. They have been assigned center frequencies that minimize overlapping with IEEE 802.11 channels 1, 6 and 11, which are commonly used in several countries.

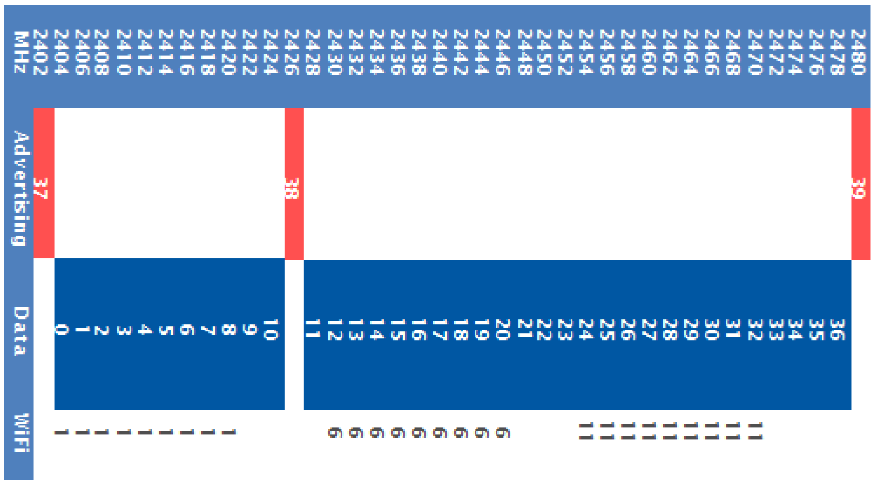


Figure 2.5: BLE channels. Retrieved from www.bluetooth.com

  Adaptive frequency hopping is used in the data channels as to avoid interference and wireless propagation issues, such as fading and multipath. This mechanism selects one of the 37 available data channels for communication during a given time interval.

  Physical channels use Gaussian Frequency Shift Keying (GFSK) modulation.

Usually, the modulation index ranges between 0.45 and 0.55, which helps reducing the peak power consumption. Data rate in the physical layer is 1 Mbps.

The receiver sensitivity is the signal level at the receiver that produces a Bit Error Rate (BER) of 10^3. The BLE specification mandates a sensitivity of no less than –70 dBm. Typical coverage range is over various tens of meters.

- Link layer

  A device acting as an advertiser transmits its data through the advertising channels in intervals of time called advertising events. Within an advertising event, the advertiser sequentially uses each advertising channel for packet transmission.

  In order to establish a connection, an advertiser announces through the advertising channels that it is a connectable device. The other device listens for such advertisements. When an initiator finds an advertiser, it may transmit a Connection Request message to the advertiser, which creates a point-to-point connection between the two devices. Both devices can then communicate by using the physical data channels. The packets for this connection will be identified by a randomly generated 32-bit access code.

  Slaves are in sleep mode by default and wake up periodically to listen for possible packet receptions from the master, which helps saving energy. The instants in which slaves are required to listen are determined by the master, who also coordinates the medium access by using a Time Division Multiple Access scheme. The master provides the slave with the information needed for the frequency hopping algorithm and for the connection supervision.

  When none of the devices has more data to transmit, the connection event will be closed and the slave will not be required to listen until the beginning of the next connection event. Other circumstances that force the end of a connection event include the reception of two consecutive packets with bit errors by either the master or the slave, and the corruption of the access address field of a packet sent by any device. In order to allow bit error detection, all data units include a 24-bit Cyclic Redundancy Check (CRC) code.

A supervision timeout happens when the time since the last received packet exceeds the connSupervisionTimeout parameter. The purpose of this mechanism is to detect the loss of a connection due to severe interference or the movement of a device outside the range of its peer.

Link Layer connections use a stop-and-wait flow control mechanism based on cumulative acknowledgments, which at the same time provides error recovery capabilities. Each data channel packet header contains two one-bit fields called the Sequence Number (SN) and the Next Expected Sequence Number (NESN). The SN bit identifies the packet, whereas the NESN indicates which packet from the peer device should be received next. If a device successfully receives a data channel packet, the NESN of its next packet will be incremented, and that packet will serve as an acknowledgment. Otherwise, if a device receives a packet with an invalid CRC check, the NESN of the received packet cannot be relied upon. This forces the receiving device to resend its last transmitted packet, which serves as a negative acknowledgment.

There exists the option of filtering. There will be a *white list* of allowed devices and all requests from others will be ignored. Aside from the security aspect, this also helps to reduce power consumption [12].

Host to Controller Interface

Optional standard interface between the Bluetooth controller subsystem (bottom three layers) and the Bluetooth host.

Host

It provides a series of protocols to enable upper layer functionalities.

- Logical Link Control and Adaptation Protocol (L2CAP)

  L2CAP is in charge of multiplexing the data of three higher layer protocols, Attribute Protocol (ATT), Security Manager Protocol (SMP) and Link Layer control signaling, on top of a Link Layer connection. The data of these services are handled by L2CAP in a best-effort approach and without the use of retransmission and flow control mechanisms, which are available in other Bluetooth versions. The maximum payload size is 32 bytes. Since upper layer protocols pass only data fitting this size no segmentation or reassembly capabilities are needed.

- Attribute Protocol (ATT)

  The ATT defines the communication between two devices playing the roles of server and client, on top of a dedicated L2CAP channel. The server keeps a set of attributes, which are data structures that store the information managed by the Generic Attribute Profile (GATT). The client or server role is determined by the GATT.

  Server's attributes can be accessed by the client by sending requests, which trigger response messages from the server. For greater efficiency, a server can also send to a client two types of unsolicited messages that contain attributes: (i) notifications, which are unconfirmed; and (ii) indications, which require the client to send a confirmation. A client may also send commands to the server in order to write attribute values. Request/response and indication/confirmation transactions follow a stop-and-wait scheme.

- Generic Attribute Profile (GATT)

  The GATT defines a framework that uses the ATT for the discovery of services, and the exchange of characteristics from one device to another. A characteristic is a set of data which includes a value and properties. The data related to services and characteristics are stored in attributes. For example, a server that runs a 'temperature sensor' service may account with a 'temperature' characteristic that uses an attribute for describing the sensor, another attribute for storing temperature measurement values and a further attribute for specifying the measurement units.

- Security Manager

  The SMP is responsible of the message exchange in the pairing process, which consists of the three phases. In the first one, the two devices announce their capabilities and an appropriate method for the next phase is chosen. In the second phase the pairing devices first agree on a Temporary Key (by means of the Out Of Band, the Passkey Entry or the Just Works methods). From this TK and some random values generated by each device, the Short-Term Key (STK) is obtained. This STK will be used in the next phase to secure the distribution of key material. In the third phase, the STK is used to secure the exchange of three 128-bit keys: the Long-Term Key (LTK), to generate Link Layer 128-bit encryption key; the Connection Signature Resolving Key (CSRK), for signing data at the ATT layer; and the Identity Resolving Key (IRK), to generate a private address based on a device public address.

- Generic Access Profile (GAP)

  The GAP specifies device roles, modes and procedures for the discovery of devices and services, the management of connection establishment and security.

  The BLE GAP defines four roles with specific requirements on the underlying controller: Broadcaster, Observer, Peripheral and Central. A device in the

Broadcaster role only broadcasts data (via the advertising channels) and does not support connections with other devices. The Observer role is complementary for the Broadcaster, i.e., it has the purpose of receiving the data transmitted by the Broadcaster. The Central role is designed for a device that is in charge of initiating and managing multiple connections, whereas the Peripheral role is designed for simple devices which use a single connection with a device in the Central role. In consequence, the Central and Peripheral roles require that the device's controller support the master and slave roles, respectively. A device may support various roles, but only one role can be adopted at a given time.

Finally, since certain types of applications may benefit from reusing common functionality, additional profiles can be built on top of the GAP. Bluetooth follows a profile hierarchy, whereby a new profile including all the requirements of an existing profile can be defined. A highest-level profile that specifies how applications can interoperate is called an application profile. Application profiles, which are also specified by the Bluetooth SIG, favor interoperability between devices from different manufacturers.

2.2.4 Link layer packet format

The same packet format is used for both advertising and data channels [6]. Their length may go from 80 bits to 2120 bits. Each packet has four fields, shown in the next figure.

- Preamble: the receiver uses this to perform frequency synchronization, symbol timing estimation, and Automatic Gain Control training. Its value must be 10101010b for advertising channel packets and either 01010101b or 10101010b for data channel packets.

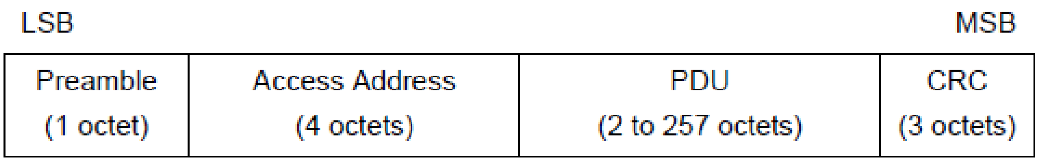


Figure 2.6: BLE link layer packet format. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

- Access Address: for advertising channel packets it is the same, 0x8E89BED6. For data channel packets it will be a random 32-bit value (with some restrictions), different for every connection.
- Protocol Data Unit (PDU): it will be either an advertising channel PDU or a data channel PDU, depending on in which type of channel the packet is being transmitted.
- Cyclic Redundancy Check (CRC): a 24-bit value calculated from the PDU.

Advertising channels

The advertising channel PDU consists of a 16-bit header and a variable size payload as shown in the figure.

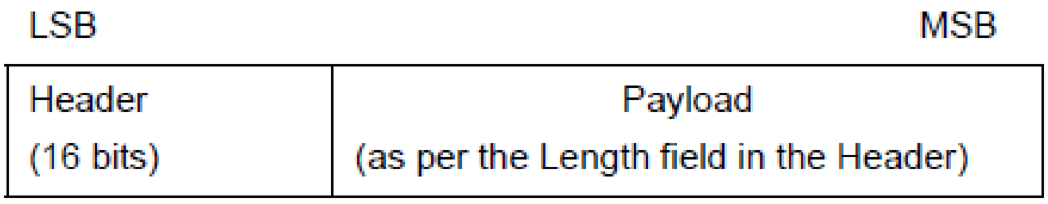
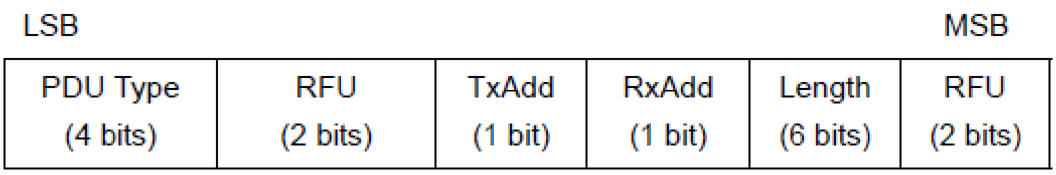


Figure 2.7: Advertising channel PDU. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

The PDU type field defines the PDU type following the information in Table 2.3. The TxAdd and RxAdd fields carry particular information of the PDU type. The length field indicates the payload length in bytes, ranging from 6 to 37 octets.

The payload field are specific to each PDU type:



LSB					MSB
PDU Type (4 bits)	RFU (2 bits)	TxAdd (1 bit)	RxAdd (1 bit)	Length (6 bits)	RFU (2 bits)

Figure 2.8: Advertising channel PDU header. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

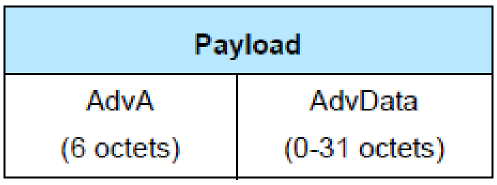
PDU type	Packet name
0000	ADV_IND
0001	ADV_DIRECT_IND
0010	ADV_NONCONN_IND
0011	SCAN_REQ
0100	SCAN_RSP
0101	CONNECT_REQ
0110	ADV_SCAN_IND
0111-1111	Reserved

Table 2.3: Advertising channel PDU type

- Advertising PDUs: those which are sent by a Link Layer in the Advertising State and received by another one either in the Scanning or the Initiating State.

  - *ADV_IND*:

    Used in connectable undirected advertising events. It consists of two fields. The AdvA carries the advertiser's public address or a random device address. The AdvData can contain advertising data from the advertising's host.

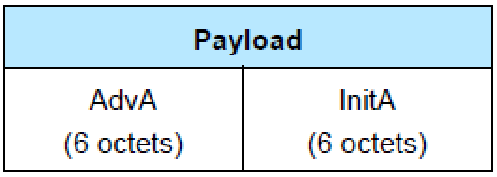


Payload	
AdvA (6 octets)	AdvData (0-31 octets)

Figure 2.9: *ADV_IND* payload. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

  - *ADV_DIRECT_IND*

Used in connectable directed advertising events. The payload carries two fields. The AdvA contains the advertiser's public or random device address. The InitA will be the address of the device to which the PDU is addressed. It can also be either a public or random device address.

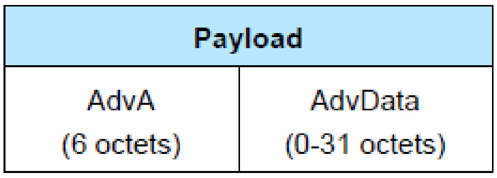


Payload	
AdvA (6 octets)	InitA (6 octets)

Figure 2.10: *ADV_DIRECT_IND* payload. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

– *ADV_NONCONN_IND*

Used in non-connectable undirected advertising events. The AdvA field is the same as before. The AdvData can contain advertising data from the advertising's host.

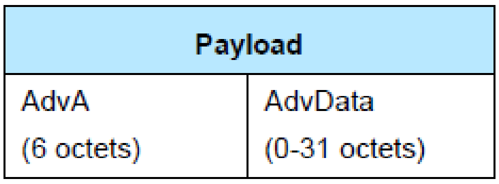


Payload	
AdvA (6 octets)	AdvData (0-31 octets)

Figure 2.11: *ADV_NONCONN_IND* payload. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

– *ADV_SCAN_IND*

Used in scannable undirected advertising events. The AdvA and AdvData field carry the same information as in the previous cases.



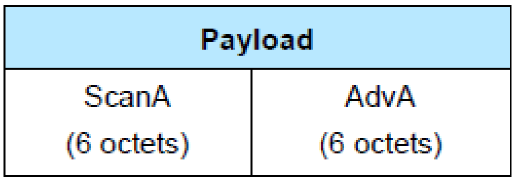
Payload	
AdvA (6 octets)	AdvData (0-31 octets)

Figure 2.12: *ADV_SCAN_IND* payload. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

- Scanning PDUs:

  - *SCAN_REQ*

    Sent by Link Layer in the Scanning State, received by a Link Layer in the Advertising State. The ScanA field carries the scanner's public or random device address. The AdvA field contains the advertiser's public or random device address.

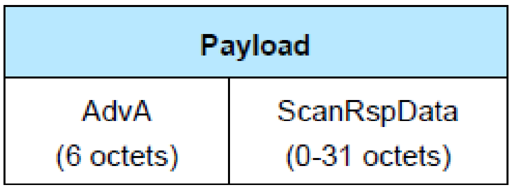


Payload	
ScanA (6 octets)	AdvA (6 octets)

Figure 2.13: *SCAN_REQ* payload. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

  - *SCAN_RSP*

    Sent by Link Layer in the Advertising State, received by a Link Layer in the Scanning State. The content of the AdvA field is the advertiser's public or random device address. The ScanRspData may contain any data from the advertiser's host.



Payload	
AdvA (6 octets)	ScanRspData (0-31 octets)

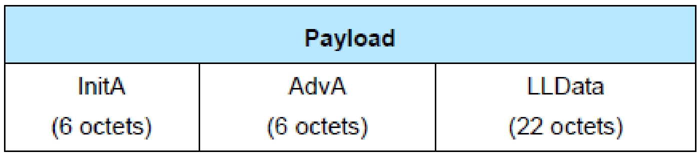
Figure 2.14: *SCAN_RSP* payload. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

- Initiating PDUs: sent by the Link Layer in the Initiating State and received by a Link Layer in the Advertising State.

  - *CONNECT_REQ*

    The InitA field is the initiator's public or random device address. The

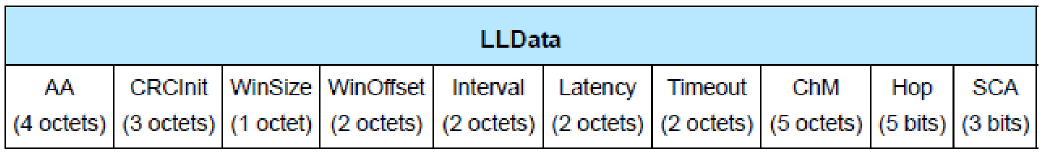
AdvA field contains the advertiser's public or random device address.



Payload		
InitA (6 octets)	AdvA (6 octets)	LLData (22 octets)

Figure 2.15: $CONNECT\_REQ$ payload. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

The LLData contains ten fields which specify some connection parameters: link layer connection's address, CRC initialization value, window size and offset, time interval between connection events, number of connection events the slave is not required to listen, timeout, channel map, hop value for the data channel selection algorithm and the sleep clock accuracy.



LLData									
AA (4 octets)	CRCInit (3 octets)	WinSize (1 octet)	WinOffset (2 octets)	Interval (2 octets)	Latency (2 octets)	Timeout (2 octets)	ChM (5 octets)	Hop (5 bits)	SCA (3 bits)

Figure 2.16: LLData field structure. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

Data channels

The data channel PDU is formed by a 16-bit header, a variable size payload and it could contain a message integrity check field too, which may only be included in encrypted connections whenever the payload length is different from zero.

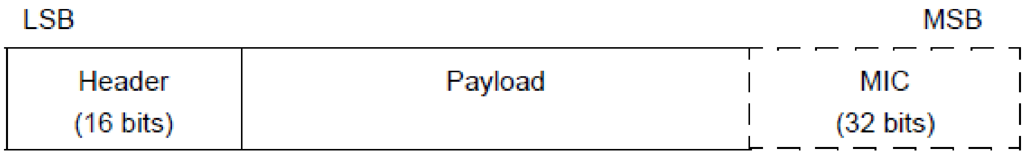
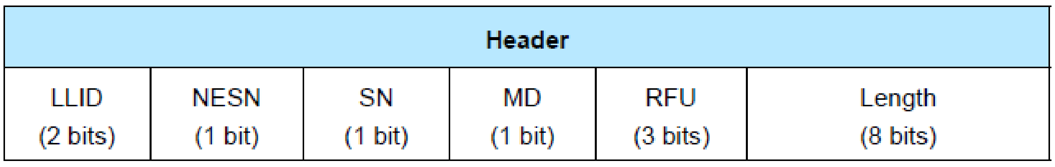


Figure 2.17: Data channel PDU. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

The LLID field determines whether the payload contains an LL Data PDU (this field value is 01b or 10b) or an LL Control PDU (its value is 11b). The NESN denote the next expected sequence number and the SN does the proper with the sequence number. The MD indicates whether the device has more data to send. The Length field contains the length of the payload, including the MIC in case it exists.

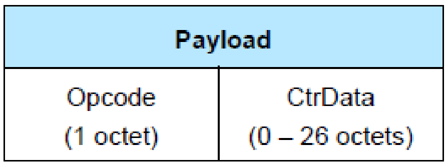


Header					
LLID (2 bits)	NESN (1 bit)	SN (1 bit)	MD (1 bit)	RFU (3 bits)	Length (8 bits)

Figure 2.18: Data channel PDU header. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

- LL Data PDU: This data channel PDU is used to send L2CAP data. They are referred to as empty PDU whenever the LLID field in the header is set to 01b and the Length field is all zeros. The Length field cannot contain only zeros if the LLID field's value is 10b.

- LL Control PDU: This data channel PDU is used to control the link layer connection. Its payload contains two fields. The Opcode distinguishes different types of LL control PDU. The CtrData is specified by the Opcode. Table 2.4 includes the possible values for this field and their meaning.



Payload	
Opcode (1 octet)	CtrData (0 – 26 octets)

Figure 2.19: LL control PDU payload. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

Opcode	Control PDU name
0x00	LL_CONNECTION_UPDATE_REQ
0x01	LL_CHANNEL_MAP_REQ
0x02	LL_TERMINATE_IND
0x03	LL_ENC_REQ
0x04	LL_ENC_RSP
0x05	LL_START_ENC_REQ
0x06	LL_START_ENC_RSP
0x07	LL_UNKNOWN_RSP
0x08	LL_FEATURE_REQ
0x09	LL_FEATURE_RSP
0x0A	LL_PAUSE_ENC_REQ
0x0B	LL_PAUSE_ENC_RSP
0x0C	LL_VERSION_IND
0x0D	LL_REJECT_IND
0x0E	LL_SLAVE_FEATURE_REQ
0x0F	LL_CONNECTION_PARAM_REQ
0x10	LL_CONNECTION_PARAM_RSP
0x11	LL_REJECT_IND_EXT
0x12	LL_PING_REQ
0x13	LL_PING_RSP
0x14	LL_LENGTH_REQ
0x15	LL_LENGTH_RSP
0x16-0xFF	Reserved for future use

Table 2.4: LL control PDU Opcodes

2.3 Market forecast

On June 2016, Bluetooth 5.0 was announced. It is expected for late 2016, early 2017. With this new version, the range of connections will be quadrupled and their speed doubled without any extra power consumption [13]. These features among others will help BLE to widen its horizons.

The latest market forecast from ABI Research indicates that BLE devices will represent 27% of total Bluetooth shipments by 2021. Given its characteristics it did not take long for BLE to establish as the leading wireless connectivity standard for the Internet of Things. The use of Bluetooth in markets such as smart homes, location related services or wearables, which have a huge potential, also helps reinforcing its position.

Another characteristic that works in favor of Bluetooth against other technologies is the dual-mode solution. It is specially important in the smart home environment, as it will contain a variety of hybrid networks.

By 2021 the beacon market is expected to be the third largest market for Bluetooth devices. Smartphones will still represent 40% of Bluetooth product shipments. However this would mean a 12% drop in volume share from 2015 [14].

3 DESIGN

This chapter will describe the first approach to defining how the system should be and work. It will cover the study carried out to that end and will finish with a design of the prototype to be implemented and a description of the hardware chosen to do so.

3.1 BLE beacons

This project will take advantage of beacon technology. A brief description of its main features is presented next to understand further designing decisions.

3.1.1 What is a beacon?

A beacon is a device configured to broadcast data. This data may serve to give information about the area the beacon is located at. A BLE beacon does so using BLE technology.

BLE beacons are configured to periodically broadcast this information. When they are not broadcasting they are in sleep mode, allowing the well-known low power consumption of this technology. Devices searching for it will be in scanning mode.

3.1.2 Parameters

The main configurable parameters, common to every BLE beacon specification are the following [15]:

- Advertising interval: this is the time the beacon will sleep between advertising events. It is the way of achieving a low power consumption. They should last for at least 100 ms for non-connectable events, and a random delay is added ranging from 0 to 10 ms.

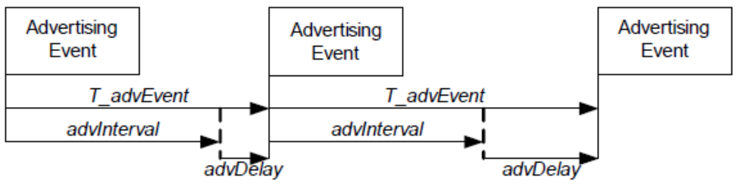


Figure 3.1: Advertising interval. Retrieved from *Specification of the Bluetooth System, Core Version 4.2*

- Transmission power: this is the power with which the signal will be broadcasted. The higher the power, the larger the range. However, it would make the beacon need more power shortening its lifetime. There are three possibilities for powering a beacon:
  - Batteries: able to provide a lifetime suitable for most applications, batteries offer the possibility of designing small, low-cost final products.
  - DC power supply: usually implies that the power consumption is not a critical parameter.
  - Energy harvesting: least common option, although being progressively introduced into low-power wireless solutions.

  The transmitted power is often included in a field so that the receiver can compare it to the received one and estimate the distance separating them.

- Identifier: different beacon specifications support different identification fields. These are important when using beacons to locate users within a room or perform a particular action.

3.2 System architecture

As mentioned in the previous section, the way location-based services are usually implemented with BLE beacons is by having a number of them as *fixed stations* advertising and the node to locate scanning. When receiving the signal from the beacons, this node can estimate the distance to each beacon and triangulate its position in the room. For the project being carried out, this configuration would imply that in case the tracked person left the room, the notification should be sent from its own device. A possible way to do so would be to develop an app for a smartphone that scanned for the beacons' signal and in the case this signal was not picked, it sent a notification over the Internet. The problem with this approach is that a smartphone would be needed, and it would not be the least expensive option. The *tracked* person would need to always carry the phone for the system to work. It may then be the case as well that the person lost the phone, implying this a great loss. These scenarios are not unrealistic at all considering the targets of the system (people suffering from mental illnesses).

To avoid these issues, it was decided that the optimal solution would be to send the notification from the *fixed stations*. To make this possible, they would need to be the ones in scanning mode, while the person's device would need to be the advertiser. The intelligence of the system will be provided by the receiving stations, as the beacon limits itself to advertising.

The idea was to have several *stations* spread around the area to cover. The person would carry a *tracking device* acting as a *transmitter*. While any of the stations received the transmitter's signal, there would be no problem as it would mean the person was still in range. The moment none of the stations were able to pick up the signal, a notification would have to be sent alerting that the person left the controlled area.

3.2.1 Dividing the area

The first step would be to divide the area to be controlled. To do so, some considerations need to be taken into account. Each station can cover an area of a fixed radius. This radius can be adjusted by setting a RSSI threshold. Stations should be within range from one another, so that they can inform the rest whether the tracked device is in their radius of coverage. An example of this is represented in Figure 3.2. There is just one station in the range of the tracking device's signal, so this station needs to be able to let the rest know it is picking up the signal. Otherwise a notification could be sent without the person actually leaving the covered area.

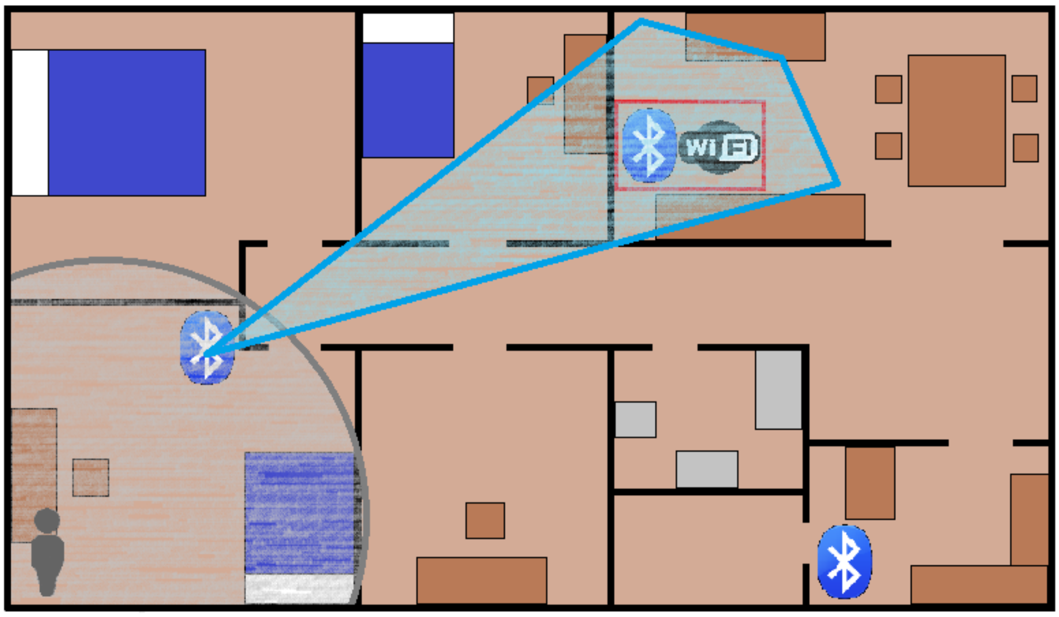


Figure 3.2: Communication between stations

In the simplest case, no divisions would be needed. A single *station* would manage to cover the desired area. This was the case used in this project, as will be explained in the next chapter.

3.2.2 Central node

The central node has special relevance. It will be the one sending the notifications. To do so, it must have a way to connect to the Internet.

It would also be in charge of the user's configuration. It should be able to facilitate a simple and easy setup process.

As mentioned in the previous section, all the other nodes should be able to communicate with the central one. For distant ones, setting a higher transmission power might be necessary, as it would increase their range. Depending on the information received from the rest of the stations, the central one will decide whether a notification should be sent or not.

If a *station* receives the *tracking device's* signal, it will switch to advertising mode so that the central node can receive its signal and be aware that the person is still within range. Then it should switch back to scanning mode to make sure that has not changed. The notification will only be sent when the central *station* does not receive any signal, neither from the *tracking device* nor from any other *station*.

Figure 3.3 shows an schematic of the configuration for the proposed scenario. The central node in this case is the one highlighted in the red square.

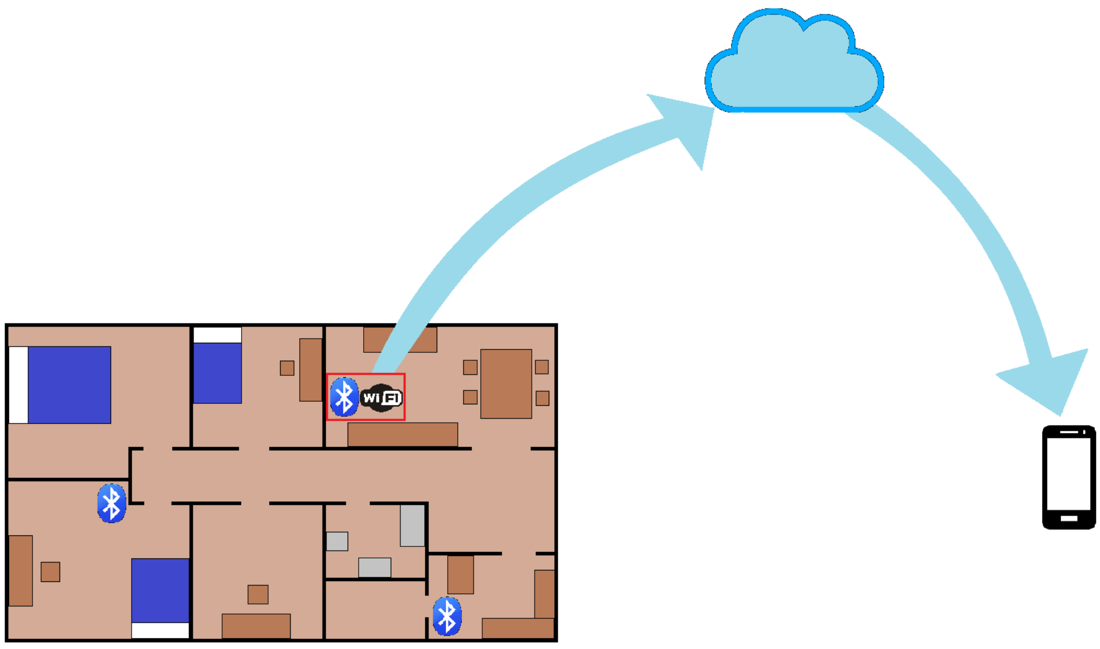


Figure 3.3: System architecture

3.2.3 Proximity estimation

With the signal each station receives, an estimate of the distance between the transmitter and each of them can be computed. This would allow to approximately triangulate the position of the subject if the area map is known.

When implementing a proximity application, the RSSI may be used. The use of its absolute value should be avoided, and try to replace it with the trend when possible [16].

Transmitting at the maximum power would yield better estimations, but it would also increase power consumption. A trade-off should be made between accuracy and power consumption.

There is noise that has to be taken into account as well. Increasing the frequency of transmissions is another way to improve accuracy in this sense. The more samples available, the better proximity estimation, as the noise could be filtered much more easily [17].

To get a proximity estimation the log-distance path model [18] may be used.

$$PL = PL_0 + 10n\log_{10}\left(\frac{d}{d_0}\right) + \chi \tag{3.1}$$

PL : *path loss value*

PL_0 : *path loss at* d_0

n : *path loss exponent*

d : *distance from beacon to device*

d_0 : *distance at which* PL_0 *was measured.*

χ : *environmental factors*

This equation can be adapted to work with RSSI values, resulting in the following:

$$RSSI = RSSI_0 - 10n\log_{10}\left(\frac{d}{d_0}\right) - \chi \tag{3.2}$$

From which an expression for the distance can be obtained:

$$d = d_0 10^{\frac{RSSI_0 - RSSI - \chi}{10n}} \tag{3.3}$$

With this, an estimate for the distance to the tracking device can be computed. For a known topology, *dangerous* areas like stairs can be controlled. With three stations knowing the distance between them and the person, his position could be triangulated and a notification could be sent whenever he got too close to any of these zones.

3.3 Privacy concerns

This is obviously a hot topic in applications using location techniques. They collect personal data, and thus, this kind of services may incur in violations of users' privacy.

There is a special sensibility in this project, as it is a presence tracker, name that may have negative connotations to some people. The aim of this project is to help people that need to be taken care of, not to spy or keep track of anyone against their will. A reasonable use has to be made at all times, only in justified cases, and respecting always other's privacy.

It is important to warn users about how their data will be used and who, if any, will have access to it. In the case of the system under development, no personal data will be stored. The information will only be accessible to the one receiving the notifications.

3.4 Designing the prototype

Due to time and budgetary limitations, a simpler version of the design described was implemented in the prototype. It will count with just one *station* and a *tracking device*.

No positioning techniques will be able to be tested as there will be only one station. For building prototype, the complexity of the communication between *stations* will not have to be faced either for the same reason.

However, it is intended to obtain a fully working system, with the mentioned limitations, but ready to be used in real cases.

3.4.1 Hardware choice

After designing the system it was time to decide what components could perform the required tasks. Taking into account the goals set for the project and the design described in this chapter, several development boards and modules were compared. For the notification sending options such as Wipy or ESP8622 boards were considered.

Comparing prices, availability and main features the final choices were RedBearLab's BLE Nano board for the *station* and *tracking device*; and NodeMCU devkit, which is a development board that includes the previously mentioned ESP8622 board, for connecting to the Internet and sending the notifications. The prototype will be built with these boards.

RedBearLab's BLE Nano

BLE Nano is a Bluetooth 4.1 Low Energy development board commercialized by RedBearLab [19]. Some of its most important characteristics are its reduced size (it is only 18.5mm x 21mm) and its low power consumption. It comes with 256KB of flash memory and 16KB of SRAM, and accepts voltages ranging between 1.8V and 3.3V. Both boards used in this project were programmed in C++ making use of the online mbed Developer Platform.

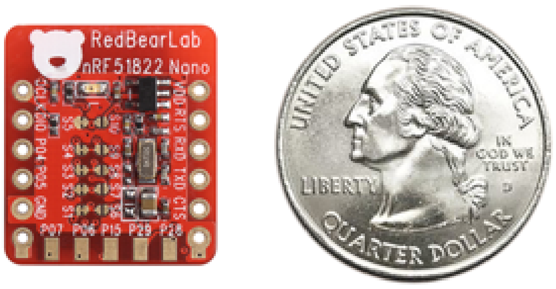
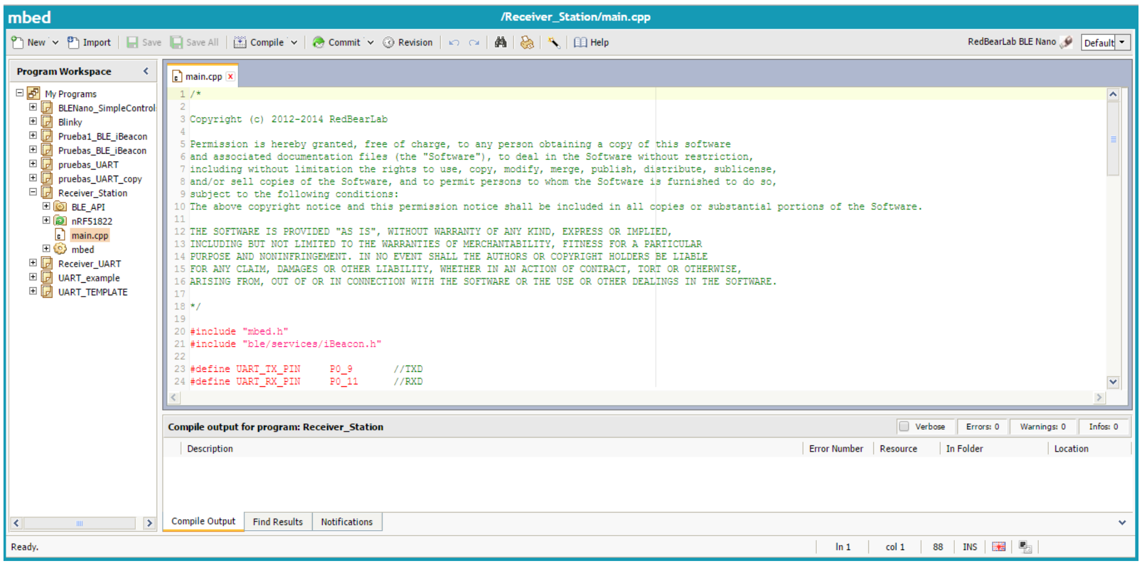


Figure 3.4: BLE Nano. Size comparison with an US quarter. Retrieved from http://redbearlab.com/blenano/



```
/*

Copyright (c) 2012-2014 RedBearLab

Permission is hereby granted, free of charge, to any person obtaining a copy of this software
and associated documentation files (the "Software"), to deal in the Software without restriction,
including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR
PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE
FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE,
ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

*/

#include "mbed.h"
#include "ble/services/iBeacon.h"

#define UART_TX_PIN     P0_9       //TXD
#define UART_RX_PIN     P0_11      //RXD
```

Figure 3.5: mbed compiler interface

<u>NodeMCU devkit</u>

NodeMCU devkit is a development board equipped with the ESP8622 board with Wi-Fi capabilities [20] . It is programmed in Lua. It can work both as an Access Point and as a Station. Unlike the ESP8622 board itself, NodeMCU is breadboard friendly (which means the separation between pins is designed to allow it fitting in a breadboard, making it much easier to work with). It also includes a USB-to-serial module, simplifying its use, as it is no longer needed to buy a separate converter and connect them.

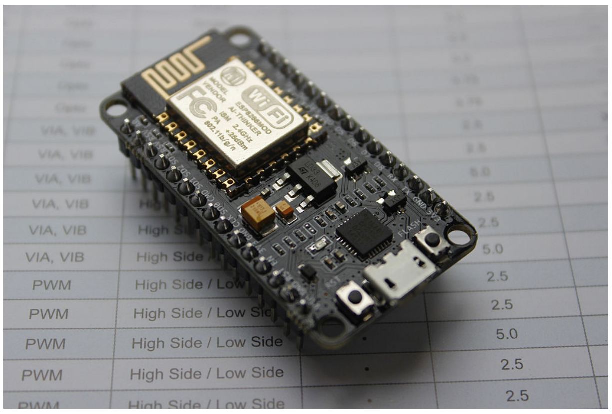
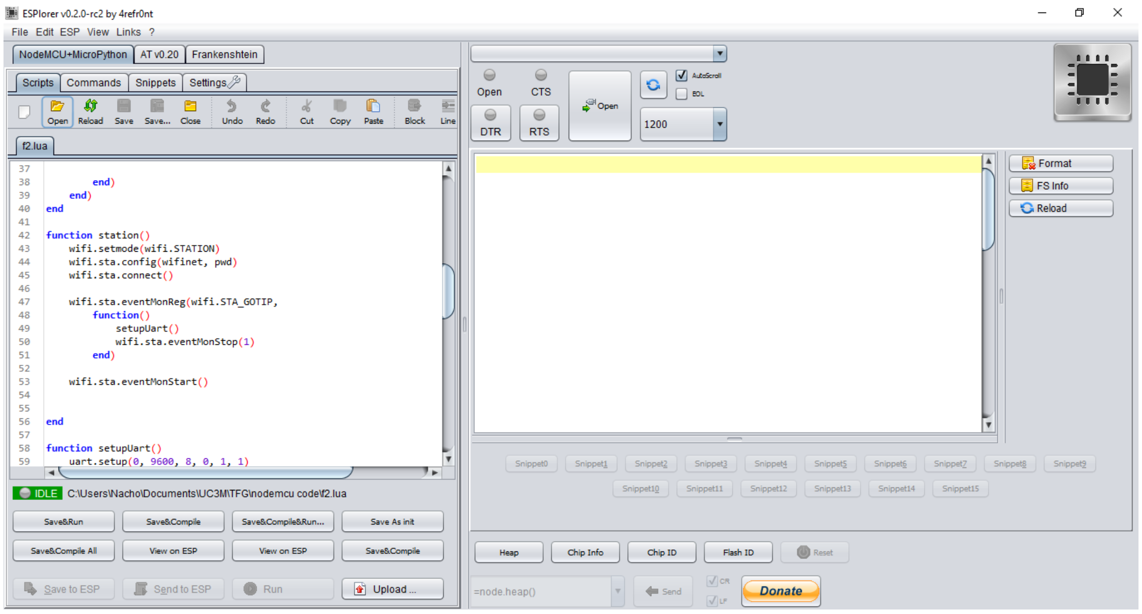


Figure 3.6: NodeMCU devkit. Retrieved from https://github.com/nodemcu/nodemcu-devkit-v1.0

This board can be programmed using ESPlorer. Connecting the NodeMCU board to the USB port of the computer, this program allows to write the Lua code and directly upload it to the board. It also offers the possibility of displaying a list of the files stored in the board's memory, and running or deleting any of them via simple commands.



```
        end)
    end)
end

function station()
    wifi.setmode(wifi.STATION)
    wifi.sta.config(wifinet, pwd)
    wifi.sta.connect()

    wifi.sta.eventMonReg(wifi.STA_GOTIP,
        function()
            setupUart()
            wifi.sta.eventMonStop(1)
        end)

    wifi.sta.eventMonStart()


end

function setupUart()
    uart.setup(0, 9600, 8, 0, 1, 1)
```

C:\Users\Nacho\Documents\UC3M\TFG\nodemcu code\f2.lua

Figure 3.7: ESPlorer work environment

4 IMPLEMENTING THE PROTOTYPE

This chapter will go through the building of the actual prototype. The events are presented in a chronological order so that the reader gets a better understanding of the decisions taken along the process.

4.1 BLE signal detection

To implement the system, simplest scenario, with one *station* and a transmitter. For that purpose, two BLE Nano boards were purchased to use one as an advertiser and the other as a scanner.

4.1.1 Advertising

The one acting as an advertiser periodically sends its signal. This is the one that the tracked person will be carrying around. It is powered with a CR2032 coin cell.

Some of the parameters discussed in Section 3.1.2 need to be adjusted in the code. The `setAdvertisingInterval()` method sets the advertising interval in ms. A reduced value for this interval will allow a scanning device to detect the advertiser faster, but it will also increase the power consumed by the latter, as more frequent transmissions would be needed. The `setTxPower()` method sets the transmission power in dBm.

4.1.2 Scanning

The scanning one will be the *fixed station*. It will be the first to notice when the tracked person abandons the area. It will carry out scans of the surroundings and as long as the advertiser's signal is picked it will do nothing. When this signal disappears or the RSSI drops below a fixed threshold, in this case −80 dBm (the reason for choosing this value will be explained in Section 5.1.2, as well as the change made to it), it means that the person is no longer in the area and a notification should be sent. After that, whenever the signal is recovered another notification should be sent letting the receiver know that the person returned home.

The `setScanParams()` method sets the scan interval and window in ms. The `scanCallback` passed as a parameter to the `startScan()` method will contain the code that needs to be executed when the results of a scan are obtained.

4.2 Sending the notification

In this first design, the notification was sent to the user by the NodeMCU via the Pushingbox API.

4.2.1 Serial port connection

The NodeMCU is connected to the scanning BLE Nano via their serial ports. The BLE Nano, by using the `Serial`'s `printf()` method, writes a different string depending on whether the advertiser device is in range or not. In case there has been a state change, the NodeMCU should send a notification. For this, it needs to be connected to a Wi-Fi network.

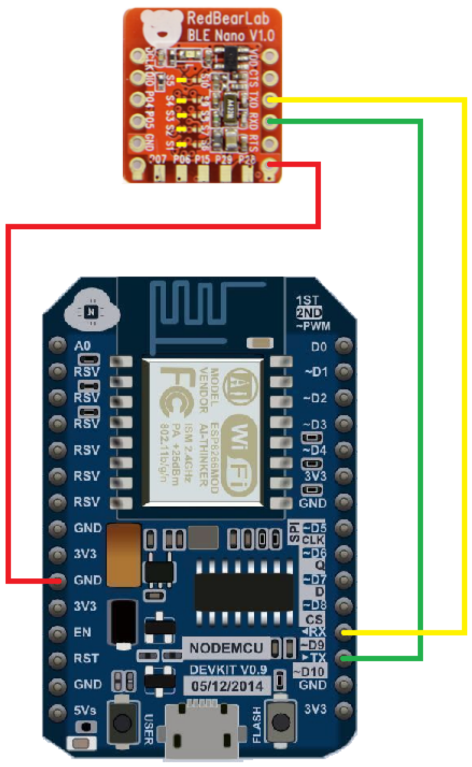
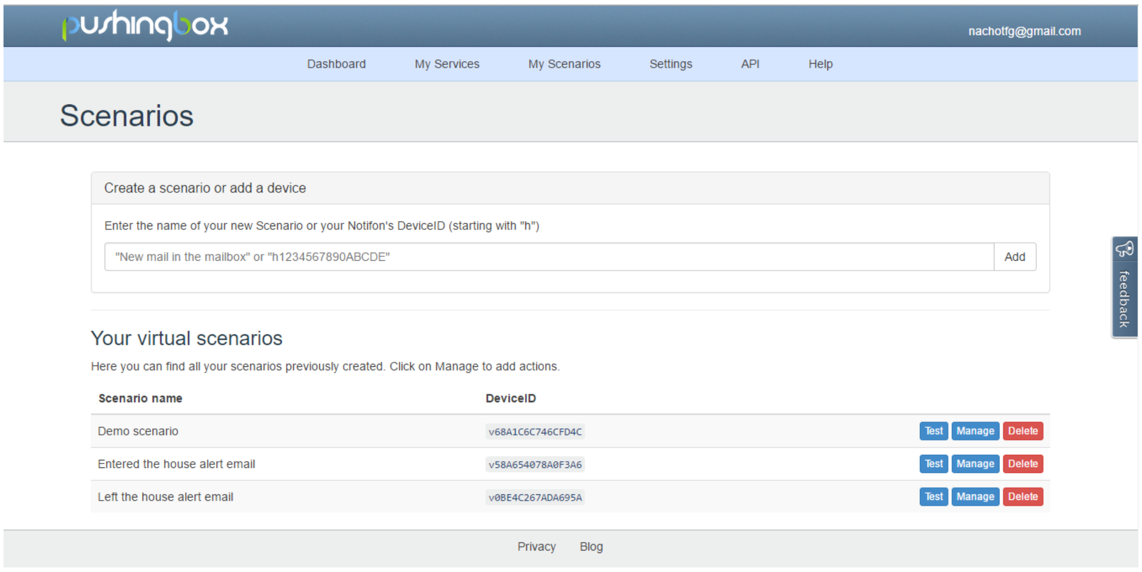


Figure 4.1: BLE Nano and NodeMCU devkit UART connection

4.2.2 Pushingbox API

Pushingbox API offers the ability to send notifications via simple HTTP requests. The first thing that needs to be done is create a scenario. Each scenario has what is called a DeviceID that identifies it. This DeviceID is sent in the HTTP request so that the appropriate scenario is triggered.

A Google account is needed in order to log in and start setting up scenarios. Once logged in, one can add services such as email, Twitter or several different push notification services for Android, iOS, Windows... These services will be used when setting up scenarios. At this point, what is left to be done is configure the scenario itself. Actions are added to the scenario. These actions will be triggered whenever an HTTP request with the DeviceID of the corresponding scenario is received. The action could be for example to send an email to a specific address saying that the HTTP request has been received.



Scenarios

Create a scenario or add a device

Enter the name of your new Scenario or your Notifon's DeviceID (starting with "h")

"New mail in the mailbox" or "h1234567890ABCDE" Add

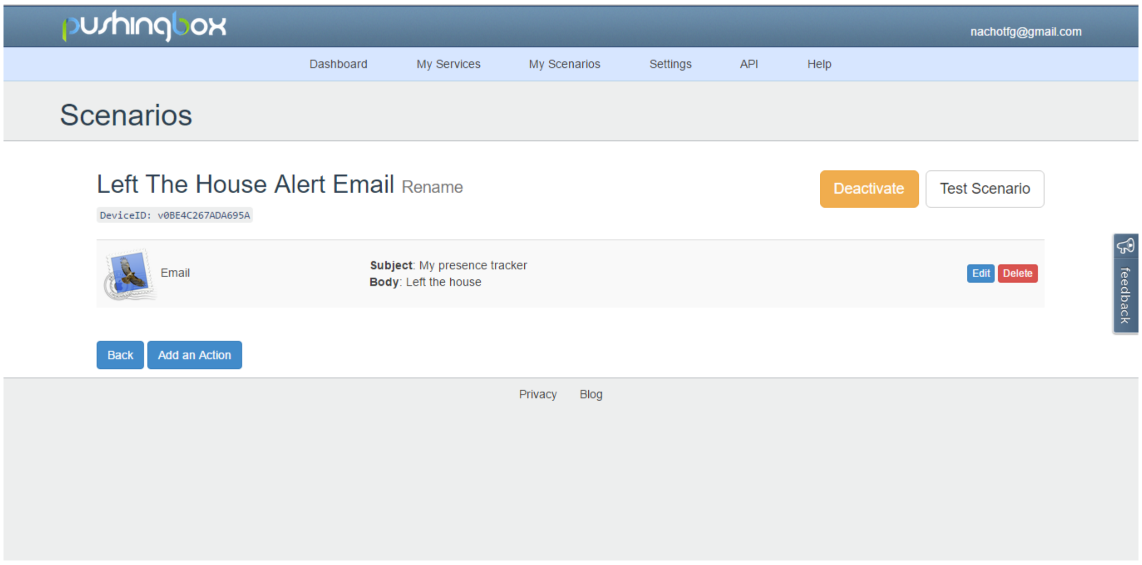
Your virtual scenarios

Here you can find all your scenarios previously created. Click on Manage to add actions.

Scenario name	DeviceID	
Demo scenario	v68A1C6C746CFD4C	Test Manage Delete
Entered the house alert email	v58A654078A0F3A6	Test Manage Delete
Left the house alert email	v0BE4C267ADA695A	Test Manage Delete

Figure 4.2: *My scenarios* tab in *Pushingbox* API

In the case of the presence tracker under consideration, more than one scenario would be needed, each one sending an email with a different content. Some of these emails could be of the kind "Grandpa left the house", "Mom arrived home"...



Scenarios

Left The House Alert Email Rename

DeviceID: v0BE4C267ADA695A

Email **Subject**: My presence tracker
Body: Left the house

Figure 4.3: An example scenario in *Pushingbox* API

At this point two scenarios were created, one for when the person left the covered area and the other for when reentering. The NodeMCU board sends the proper HTTP request upon change of state detection.

4.3 User friendly configuration

Up to this point it has been demonstrated that the notification sending can be triggered by an event such as the subject leaving the covered area. However, the prototype is not launchable to the market as is yet. Pushingbox API is really simple to use to send notifications, but trouble comes when attempting to make the presence tracker user friendly. For each scenario there is a DeviceID that the NodeMCU devkit needs to be aware of. It would be a tough time for an average user to set everything up. This is why it was decided to use Firebase Cloud Messaging to try to offer a smoother experience to the user. This has also the advantage of opening the possibility of creating a personalized interface aimed to present the user the received notifications.

4.3.1 Firebase Cloud Messaging

Firebase Cloud Messaging is a service offered by Google that allows to reliably deliver messages [21]. An FCM implementation includes an app server that interacts with FCM via HTTP or XMPP protocol, and a client app. For the latter there are currently three possible options, Android, iOS and web versions.

4.3.2 Implementing an HTTP app server

The implementation of an HTTP server over an XMPP one was chosen for two main reasons: the wider knowledge on HTTP protocol at the time and the possibility

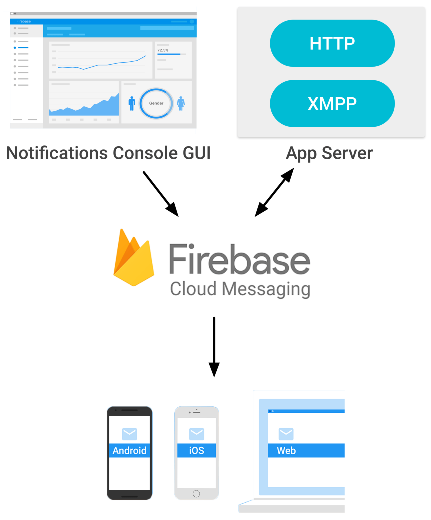


Figure 4.4: Firebase Cloud Messaging architecture. Retrieved from https://firebase.google.com/docs/cloud-messaging

of sending a message to multiple registration tokens, which is not supported in the XMPP server [22]. Even though this last feature is not used in the project, it is left as one of the possible future improvements, which will be discussed in more depth in Section 7.3.1. It also forced the decision to use JSON objects instead of plain text in the content of the requests. To send a notification, the app server issues a POST request which must contain two mandatory headers:

- Authorization: its value has to be `key=SERVER_KEY`
- Content-Type: it can be either `application/json` for JSON objects, which were used in this project; or `application/x-www-form-urlencoded;charset=UTF-8` for plain text

A final request could look like something similar to the following:

```
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA
Content-Type:application/json

{ "data" : {
     "text" : "Mark left home"
  },
  "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1..."
}
```

In this example JSON objects are used, as in the final prototype. The `"to"` field takes the value of the registrationId of the client the message is being sent to. Inside the `"data"` field the information that needs to be sent is encapsulated. In this case a `"text"` field is added with the message that the client will display in the notification to the user.

The NodeMCU board will host the HTTP app server, as it is in charge of sending the notifications. To do so, it needs to be aware of the registrationId of the client app it will be sending those notifications to.

On boot up, the NodeMCU will work in AP mode, so that the client app can connect to it and send that registrationID. At this moment, the client will also send the Wi-Fi network information (SSID and password) so that the NodeMCU can connect to the Internet and send the notifications. When all this data is received, the NodeMCU will switch to Station mode, connect to the Wi-Fi network and be ready to start receiving data from the BLE Nano.

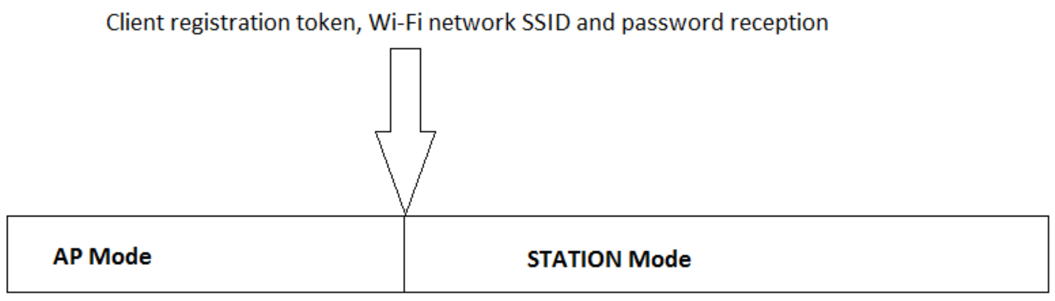
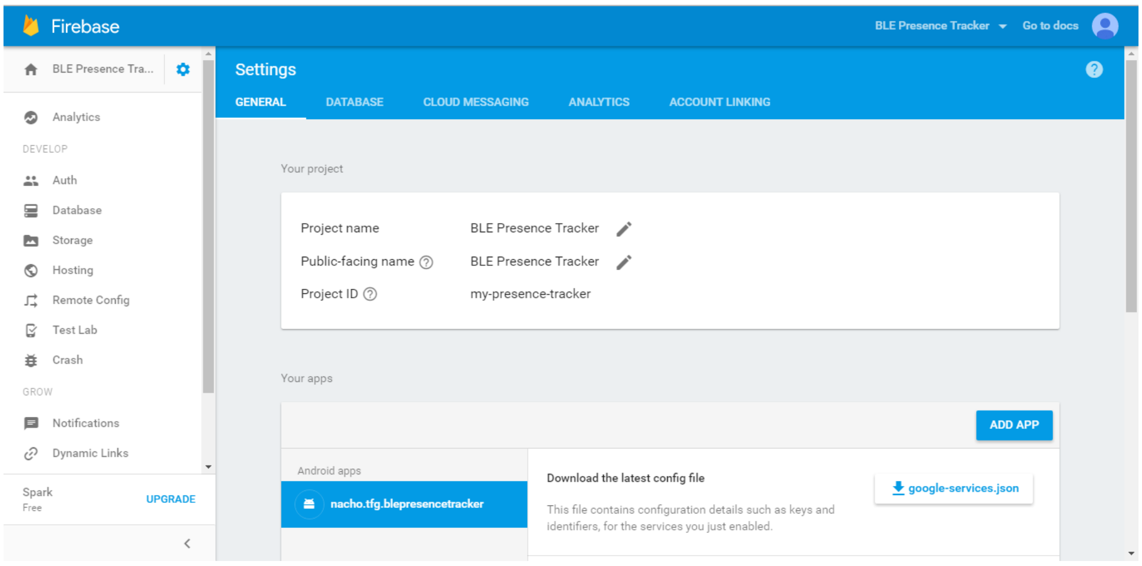

Client registration token, Wi-Fi network SSID and password reception
AP Mode
STATION Mode

Figure 4.5: NodeMCU working mode timeline

For this project an Android app was implemented.

4.3.3 Developing an Android app

There are some steps that need to be followed in order to start implementing an Android client [23]. First of all, a Firebase project needs to be created in the Firebase console. The app's packet name will be required to add Firebase, and a `google-services.json` file will be downloaded. This file needs to go into the project's module folder, usually `app/`.



Project name BLE Presence Tracker
Public-facing name BLE Presence Tracker
Project ID my-presence-tracker

nacho.tfg.blepresencetracker
Download the latest config file
This file contains configuration details such as keys and identifiers, for the services you just enabled.
google-services.json

Figure 4.6: Firebase Project settings in the Firebase Console

Then, in the Android Studio project the following needs to be added to the `build.gradle` file:

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:3.0.0'
    }
}
```

And in the `app/build.gradle` file the FCM dependency must be added. The `apply plugin` line needs to be appended at the bottom of the same file:

```
dependencies {
      compile 'com.google.firebase:firebase-messaging:9.0.0'
}

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

The minimum SDK version needs to be set to 8 in order for FCM to work, and in the Android manifest the following two services should be created:

```
<service android:name=".MyFirebaseMessagingService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>

...

<service android:name=".MyFirebaseInstanceIDService">
    <intent-filter>
        <action android:name="com.google.firebase.INSTANCE_ID_EVENT" />
    </intent-filter>
</service>
```

The first one will allow the app to receive and handle messages and the second one will serve to obtain a registration token.

In the first launch of the app, this ID token is created. It will be needed to be able to send messages to this device, so it should be sent to the server. Its value can be accessed by extending the `FirebaseInstanceIdService`. It is retrieved just by calling `getToken` in the `onTokenRefresh()` callback.

To be able to receive messages, a service extending `FirebaseMessagingService` has to be created. In this service, the `onMessageReceived` callback should be overriden to give the desired functionality [24] .

In the app under development, the `onMessageReceived` contains a call to the `sendNotification()` method, which builds a notification to show to the user and adds it to a list of notifications in the main screen of the app. All notifications are stored in this list (as seen in Figure 4.9c), in case the user misses the notification at first or wants to keep a record of them. The user will also be able to delete any notification from the list.

4.4 Final result

In the end, this prototype is conformed by three items. A tracking device, a station and an Android app.

To configure the system, the station must be powered and the smartphone that will receive the notifications within its range. In the app, click the *Add* button select the NodeMCU's network and then select the network the NodeMCU should connect to and introduce its password, when there is one. Figures 4.7 and 4.8 show this process. Once this steps are followed, a message will inform the user whether the setup was successful or it should be done again. In the former case, everything will be ready to start receiving notifications. The system will be running until the batteries are removed or run out of power.

When a new notification arrives, it will be stored in a list with all the notifications chronologically ordered. The user may delete any notification he is not interested in.

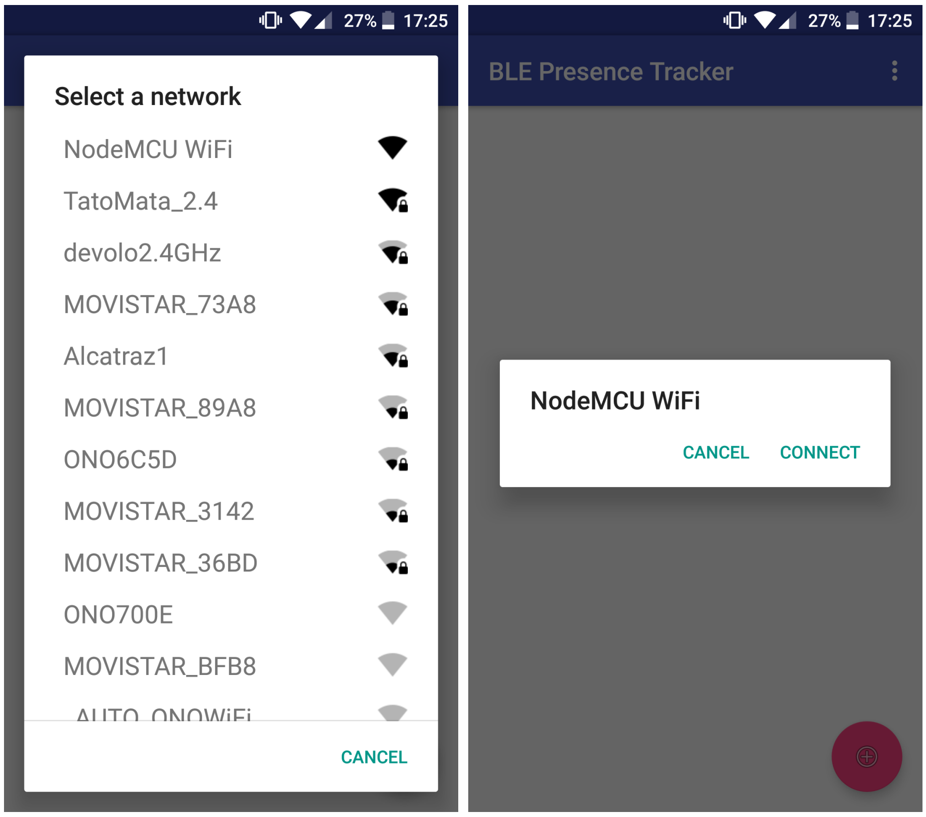
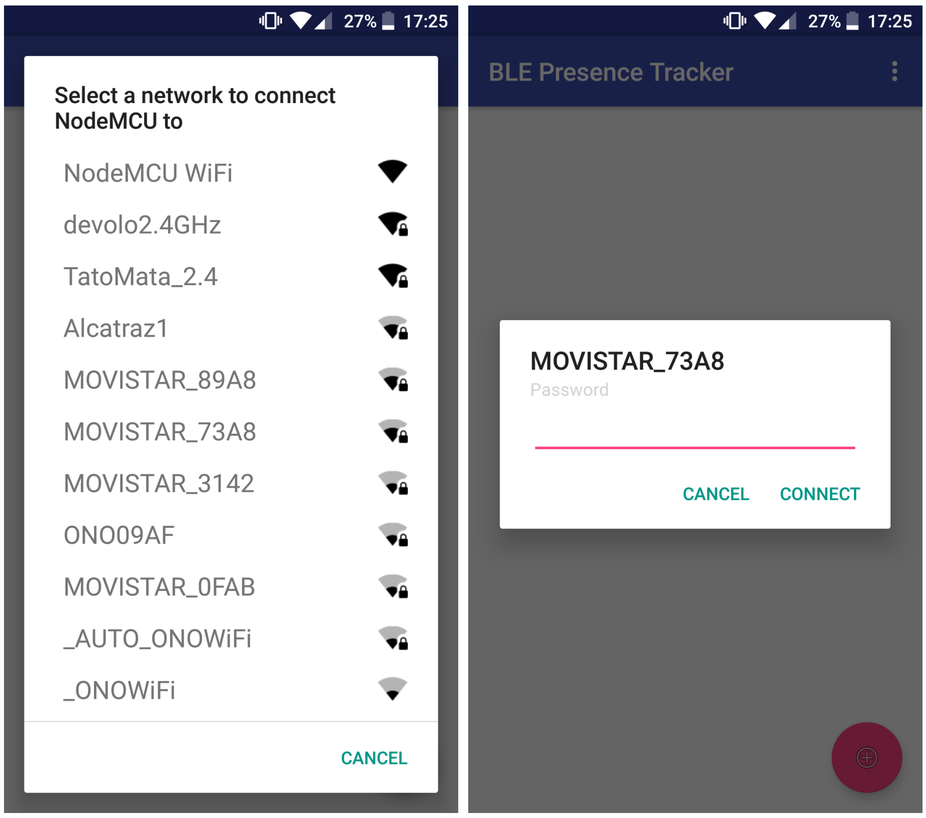


Figure 4.7: Connecting to the NodeMCU network


27% 17:25

Select a network to connect NodeMCU to

NodeMCU WiFi
devolo2.4GHz
TatoMata_2.4
Alcatraz1
MOVISTAR_89A8
MOVISTAR_73A8
MOVISTAR_3142
ONO09AF
MOVISTAR_0FAB
_AUTO_ONOWiFi
_ONOWiFi

CANCEL

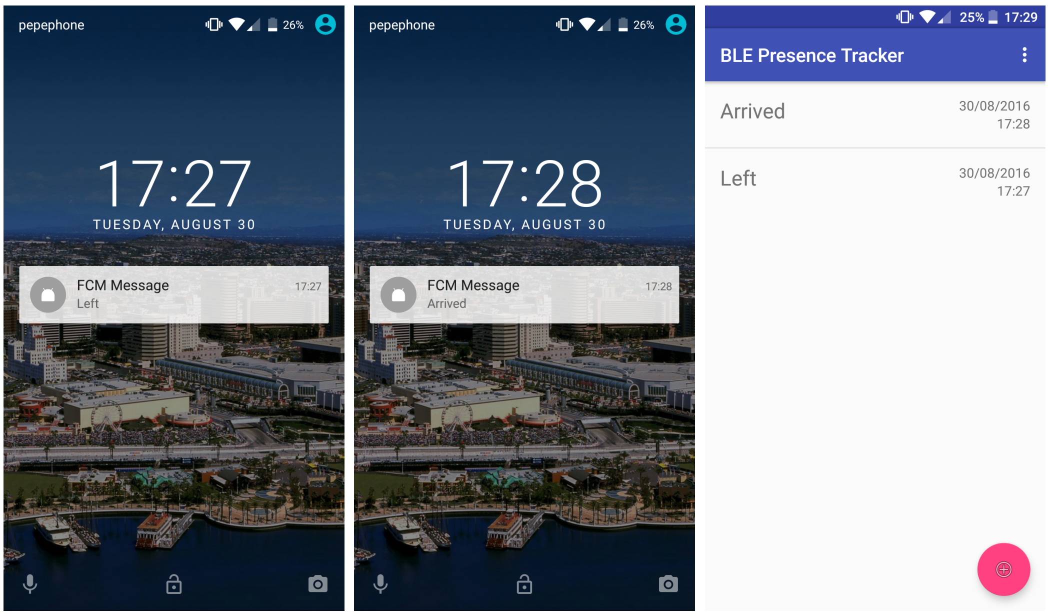
27% 17:25

BLE Presence Tracker

MOVISTAR_73A8
Password

CANCEL CONNECT

Figure 4.8: Selecting the network NodeMCU should connect to


pepephone 26%

17:27
TUESDAY, AUGUST 30

FCM Message 17:27
Left

pepephone 26%

17:28
TUESDAY, AUGUST 30

FCM Message 17:28
Arrived

25% 17:29

BLE Presence Tracker

Arrived	30/08/2016 17:28
Left	30/08/2016 17:27

(a) *Left* notification (b) *Arrived* notification (c) Notifications list

Figure 4.9: Receiving notifications

5 TESTING

This chapter describes the tests the built prototype was subjected to. An analysis of the obtained results is made in order to determine the reliability of the system.

5.1 Testing

Numerous tests were carried out during the implementation of the prototype. Every step described in the previous chapter was properly checked for the correct behavior before moving on to the next one.

The final prototype was subjected to two different tests. In the first one it was intended to detect the loss of the signal. The second one checked whether the signal's strength dropping below a given value triggered the sending of a notification.

5.1.1 Total loss of signal

This first scenario was very similar to the one used while developing. All changes in the code – either NodeMCU or Android app– were tested in this manner. The BLE Nano acting as advertiser was disconnected to check if this was detected by the scanning one and a notification was actually sent.

Every test made in this scenario with the final prototype resulted in the expected outcome, as noted in Table 5.1. The proper notification was sent in all cases, depending on whether the user left (when the advertiser was disconnected) or arrived (when it was reconnected).

Measurement	**1**	**2**	**3**	**4**	**5**
Result	OK	OK	OK	OK	OK
Measurement	**6**	**7**	**8**	**9**	**10**
Result	OK	OK	OK	OK	OK

Table 5.1: Total loss of signal test results

When this was accomplished, it was time to move on to the next scenario.

5.1.2 RSSI drops below threshold value

This test is a closest simulation to real life usage. Now, instead of disconnecting the *tracking device*, it was carried while moving around. At a given distance from the *station* a *Left* notification should be sent.

The area to cover was reduced to observe the response of the system. To determine the correct RSSI threshold value to cover the desired area several measures of the RSSI at the edge of the zone were taken and the average was set as the threshold. These values, which are shown in Table 5.2 were measured with an app that displayed the value of the RSSI of the received signals.

Measurement	**1**	**2**	**3**	**4**	**5**
RSSI	–79 dBm	–75 dBm	–85 dBm	–82 dBm	–76 dBm
Measurement	**6**	**7**	**8**	**9**	**10**
RSSI	–81 dBm	–88 dBm	–76 dBm	–82 dBm	–79 dBm

Average
–80 dBm

Table 5.2: RSSI values at the edge of coverage

With the threshold value set to –80 dBm, the prototype was tested to see if the notification was actually sent when the tracking device left the area, and how accurately this was detected. Figure 5.1 shows a representation of the scenario used for this test. The ideal response would be to send the *Left* notification when the person was in the green area, and the *Arrived* one when in the red area.

The number of samples for this test was tripled in order to get a clearer impression of the real reliability of the system.

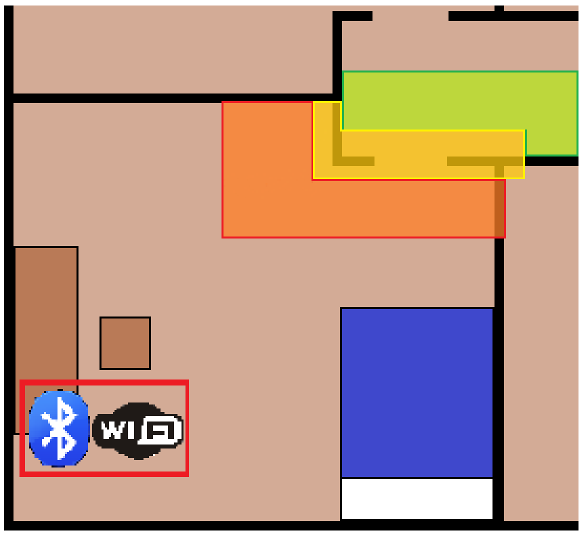


Figure 5.1: Test scenario

Area	Green	Yellow	Red
Left notifications	6	11	13
Arrived notifications	8	10	12

Table 5.3: RSSI threshold test results

Values given in Table 5.3 indicate that something was wrong with the system. Too many *Left* notifications were sent when standing in the red zone. A little research was done and everything pointed to the fact that there are variations on the RSSI value depending on the receiving system. To try to correct this behavior, new measures were taken with the values received by the BLE Nano. A program to send the RSSI value to the NodeMCU and printing it was written for that purpose. The new values are shown in Table 5.4.

Measurement	**1**	**2**	**3**	**4**	**5**
RSSI	–90 dBm	–79 dBm	–82 dBm	–81 dBm	–88 dBm
Measurement	**6**	**7**	**8**	**9**	**10**
RSSI	–85 dBm	–88 dBm	–84 dBm	–85 dBm	–87 dBm

Average
–85 dBm

Table 5.4: RSSI updated values at the edge of coverage

The results of the test with this new threshold were as shown in Table 5.5.

Area	Green	Yellow	Red
Left notifications	9	14	7
Arrived notifications	12	10	8

Table 5.5: Second RSSI threshold test results

As it can be seen, there were still too many *Left* notifications sent from the red zone. To improve the response a bit more, a final tweak was added. Instead of setting a threshold and whenever the RSSI was lower sending the *Left* notification and when it was larger the *Arrived* one, it was decided to set two thresholds. The *Left* notification would be sent if the RSSI value was below the lower threshold, and the *Arrived* one when the RSSI value was above the larger threshold. This way, there is a *safety margin* to compensate the big variations detected in the RSSI value. The values used were –82 dBm and –88 dBm for the upper and lower threshold respectively. Table 5.6 contains the results of the test using this configuration.

Area	Green	Yellow	Red
Left notifications	15	12	3
Arrived notifications	4	13	13

Table 5.6: Final RSSI threshold test results

Difficult to establish a fixed distance due to the fluctuations in RSSI at the *station.*

5.2 Results analysis

The first test results demonstrate that the system is able to detect when the *tracking device* is out of range. It does so with a 100% reliability. So this proves that it is indeed possible to implement a reliable presence tracker taking advantage of BLE technology.

However, the results obtained from the second test denote that there is a true difficulty when trying to determine the radius of action. Because of the unpredictable changes in the signal's strength, it is almost impossible to guarantee that the system will limit its coverage to a specific area. Results in Table 5.6 support this statement.

The *Left* notification is sent a 100% of the times the *tracking device* leaves the area. The problem is that there is a big variation of the position at which the *tracking device* is when the notification is sent. Also, sometimes the notification is sent without the person actually leaving the area, just when standing close to the edge of it. The same happens for the *Arrived* notification.

A possible workaround for this, could be to allow a safety margin by increasing a bit the RSSI thresholds. This increase should be small enough to not let the person go too far without sending the corresponding *Left* notification, but also not sending the notification without the person actually leaving the area.

Also, if the person decided to stand in the edge of coverage (the yellow zone in Figure 5.1), a lot of alternating *Left* and *Arrived* notifications would be sent, which can be very annoying for the receiver.

6 PROJECT HISTORY

This chapter covers everything related to the project development. The stages it was divided into, the planning made to carry it out, some of the problems faced while working on it and the budget covering the costs associated to it.

6.1 Project stages

6.1.1 Research

When the objectives of the project were set, some research needed to be carried out in order to be able to make good decision regarding how to proceed. BLE technology and its uses in location services were deeply studied. After that, it was time for investigating and comparing different BLE and Wi-Fi enabled development boards.

6.1.2 Beacon detection

Once the BLE boards were acquired. During this stage the learning of some basic notions of the C++ programming language was mandatory, as the boards purchased were programmed in that language.

6.1.3 Sending notification

At this point, the communication between the BLE Nano acting as a *station* and the NodeMCU was set. As in the previous stage, a new programming language was studied, in this case Lua. The notification was sent via the Pushingbox API.

6.1.4 Android application

The prototype was improved to make it user friendly. Firebase Cloud Messaging allowed it. This stage included the implementation of an HTTP server and the development of the Android application.

6.1.5 Testing the prototype

This stage included all the tests the prototype had to go through. It is a crucial stage, as the results of those tests would validate or not the good performance of the built system.

6.1.6 Documentation

This last stage consisted on the elaboration of this document describing the work done throughout the project.

6.2 Planning

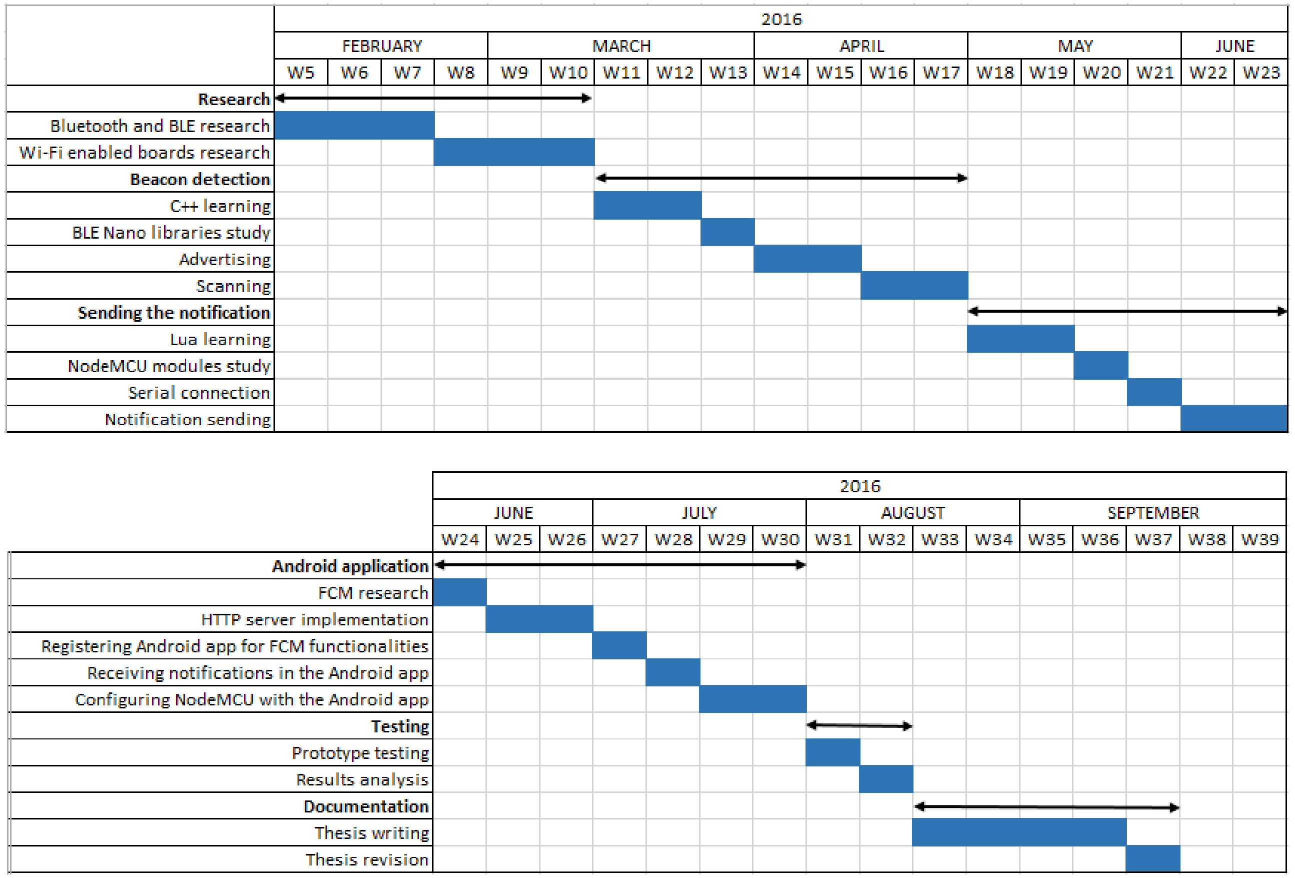
Planning is very important in any project. A thoroughly made planning can prevent many future possible problems. Special care needs to be taken with respect to available resources (mainly time and budget for this particular project). A breakdown of the tasks to complete it has to be done, defining their start and ending dates and the group or person responsible. As this project was carried out by just one person, the latter was not included in the planning.

The project was divided into the smaller tasks listed in Table 6.1. The work was done part-time dedicating 20 hours a week, so the planning was done based on weeks.

To get a clearer impression of the tasks to perform and the way they are ordered in time, they were also represented in a Gantt chart depicted in Figure 6.1.

Stage	**Task**	**Start**	**End**	**Weeks**
1	Bluetooth and BLE research	01/02/2016	21/02/2016	W5-W7
	Wi-Fi enabled boards research	22/02/2016	13/03/2016	W8-W10
2	C++	14/03/2016	27/03/2016	W11-W12
	BLE Nano documentation	28/03/2016	03/04/2016	W13
	Advertising	04/04/2016	17/04/2016	W14-W15
	Scanning	18/04/2016	01/05/2016	W16-W17
3	Lua	02/05/2016	15/05/2016	W18-W19
	NodeMCU documentation	16/05/2016	22/05/2016	W20
	Serial connection	23/05/2016	29/05/2016	W21
	Notification sending	30/05/2016	12/06/2016	W22-W23
4	FCM research	13/06/2016	19/06/2016	W24
	HTTP server implementation	20/06/2016	03/07/2016	W25-W26
	Registering Android app for FCM functionalities	04/07/2016	10/07/2016	W27
	Receiving notifications in the Android app	11/07/2016	17/07/2016	W28
	Configuring NodeMCU with the Android app	18/07/2016	31/07/2016	W29-W30
5	Prototype testing	01/08/2016	07/08/2016	W31
	Results analysis	08/08/2016	14/08/2016	W32
6	Thesis writing	15/08/2016	11/09/2016	W33-W36
	Thesis revision	12/09/2016	18/09/2016	W37

Table 6.1: Tasks breakdown



	2016																		
	FEBRUARY				MARCH					APRIL				MAY				JUNE	
	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15	W16	W17	W18	W19	W20	W21	W22	W23
Research	↔	↔	↔	↔	↔	↔													
Bluetooth and BLE research	🟦	🟦	🟦																
Wi-Fi enabled boards research				🟦	🟦	🟦													
Beacon detection							↔	↔	↔	↔	↔	↔	↔						
C++ learning							🟦	🟦											
BLE Nano libraries study									🟦										
Advertising										🟦	🟦								
Scanning												🟦	🟦						
Sending the notification														↔	↔	↔	↔	↔	↔
Lua learning														🟦	🟦				
NodeMCU modules study																🟦			
Serial connection																	🟦		
Notification sending																		🟦	🟦

	2016															
	JUNE			JULY				AUGUST				SEPTEMBER				
	W24	W25	W26	W27	W28	W29	W30	W31	W32	W33	W34	W35	W36	W37	W38	W39
Android application	↔	↔	↔	↔	↔	↔	↔									
FCM research	🟦															
HTTP server implementation		🟦	🟦													
Registering Android app for FCM functionalities				🟦												
Receiving notifications in the Android app					🟦											
Configuring NodeMCU with the Android app						🟦	🟦									
Testing								↔	↔							
Prototype testing								🟦								
Results analysis									🟦							
Documentation										↔	↔	↔	↔	↔		
Thesis writing										🟦	🟦	🟦	🟦			
Thesis revision														🟦		

Figure 6.1: Gantt chart

6.3 Problems encountered

6.3.1 C++ documentation for BLE Nano

At the time the project was being done, the documentation for the BLE Nano's C++ libraries was unclear and incomplete. This was an issue when trying to understand the proper way to make the boards either advertise or scan.

Whereas the NodeMCU documentation was quite complete and easy to understand, for the BLE Nano it was hard to figure out what each method did, which parameters were needed and what would be returned.

It is understandable as it was a fairly new board, and everything was still under development. When in the testing stage, the documentation had to be revisited and it was already presented in a much nicer way than before. Every method with its corresponding explanation.

6.3.2 C++ library for BLE Nano

Aside from the documentation discussed in the previous section, there were some important issues with the behavior of the libraries themselves.

Whenever a scan starts all the values assigned to the variables reset. This made it impossible to keep track of whether the state of the tracked person had change (whether he was in the area and left or vice versa). An exhaustive research was done to dive deep into it and it was found that it was a common issue, so a workaround needed to be found.

The solution was to make the NodeMCU be the one to control those changes. The

BLE Nano acting as a scanner will always inform the NodeMCU if the advertiser's signal was received and the NodeMCU will only send the notification whenever there was a state change.

6.3.3 NodeMCU devkit memory management

This was probably the biggest issue faced during the building of the prototype. Sometimes, when running a Lua script on the board, it gave the expected results while some other times it just did not work. It seemed as if the board had run out of free memory. However, there was actually a lot of free memory left. This was an important problem, because the NodeMCU had strange behaviors leading to irregular performances of the system.

To solve it, the code was optimized, eliminating lines written for test purposes. Also some features were removed, like sending a notification when the tracked person stayed for a long time in the same spot. This required a counter and the NodeMCU could not manage it.

As it is still under development and there are firmware updates, hopefully this issue could get solved in the future.

6.4 Resources

This section gives a list of the resources used for the completion of the project

6.4.1 Hardware

- Acer Aspire S3-951-2464G34iss laptop (Intel Core i5-2467M CPU @ 1.6GHz, 4GB RAM memory, 320GB hard disk)
- BQ Aquaris M5 (Qualcomm® Snapdragon™ 615 @ 1.5GHz, 3GB RAM memory, Android Marshmallow 6.0.1)
- 2 BLE Nano development boards
- MK20 USB Board: to program the BLE Nano boards
- NodeMCU devkit
- 2 CR 2032 batteries
- 30AWG Wrapping Wire
- 2 Breadboards

6.4.2 Software

- mbed Developer Platform: to write the code for the two BLE Nano boards.
- ESPlorer: to write the code for the NodeMCU.
- Android Studio: to develop the Android app.
- LaTeX: to write this document.

6.5 Project budget

The project budget should include a detailed description of the costs associated to the project from start to the end. It should take into account staff and material related expenses, as well as any other extra costs that there may be.

For the personnel associated costs, the number of hours for the supervisor was computed as a 10% of the number of hours worked by the author.

Full name	**Professional category**	**Devoted time (hours)**	**Worker costs per hour (€)**	**Total cost (€)**
Torralba Álvarez, José Ignacio	Engineer	660	20	13,200
Díaz Sánchez, Daniel	Senior Engineer	66	25	1,650
			Total	14,850

Table 6.2: Personnel budget

Description	**Cost (€)**	**Project devoted time (%)**	**Devoted time (months)**	**Deprecia-tion period (months)**	**Attributable cost (€)**
Acer Aspire S3-951	600	100	7	60	70
BQ Aquaris M5	280	100	6	36	46.67
BLE Nano kit (BLE Nano + MK20 USB board)	30.35	100	6	36	5.06
BLE Nano	16.75	100	6	36	2.79
NodeMCU devkit board	16.90	100	6	36	2.82
2 CR 2032 batteries	5	100	6	24	1.25
2 breadboards	10	100	6	60	1
Wrapping Wire (1 m)	1	100	6	60	0.1
				Total	135.69

Table 6.3: Equipment budget

To compute the attributable cost the following formula was applied:

$$\frac{Devoted\ time}{Depreciation\ period} \times Cost \times Project\ devoted\ time\ (\%) \tag{6.1}$$

All the software used for the realization of this project was free, so there is no cost associated to software expenses.

Aside from the costs listed in Tables 6.2 and 6.3, a 20% of the total was added as indirect costs to cover for shipping rates and utilities (Internet connection, power...). The 21% added value tax in Spain was computed as well.

Description	**Cost (€)**
Personnel costs	14,850
Equipment costs	135.69
Software costs	0
Indirect costs	2,997.14
Taxes	3,776.39
TOTAL	21,759.22

Table 6.4: Total cost

After all of this is done, the total cost of the project adds up to **twenty-one thousand, seven hundred and fifty-nine euro and twenty-two cents**.

7 CONCLUSIONS

This chapter contains the final conclusions reached when the project was completed and the prototype tested. An analysis of the fulfilled objectives is made. It also gives some personal thoughts of the whole process and different ideas to improve the system in the future.

7.1 General conclusions

With the realization of this project, it has been proven that it is indeed possible to implement a presence tracking system taking advantage of BLE technology. At least in its simplest form, with only one *station*.

An important point in the realization of this project was to be able to build a system like this at a low cost. This was also fulfilled, as the hardware's cost went to a bit over 60€. The tracking device itself is less than 20€. This part of the system is the most critical one in terms of cost. It is the one the sick person will be carrying around, so it is the most likely to be lost or damaged. In case that happened and it needed to be replaced, 20€ would be an acceptable price to pay.

Another of the goals set at the beginning was to come up with something easy to use. This was satisfied, as the resulting prototype is rather user friendly and could be utilized by anyone, without needing any special knowledge to do so. The setup process is very similar to connecting to a Wi-Fi network. All that is needed is downloading the app in an Android smartphone and tapping a few buttons. The system takes care of storing the registrationIDs and similar, which are transparent for the user.

However, it has to be said that, as the results of the tests discussed in Section 5.2 showed, sometimes it can provide varying outcomes. Fluctuations in RSSI may trigger the notification sending at a moment that it normally would not. This only happened when standing on the edge of the covered area. In these cases an *Arrived* notification was received right after a *Left* one. If the person actually left the zone, the response of the system is the expected one.

Despite this, the prototype as is does work.

Due to the NodeMCU memory limitations discussed in Section 6.3.3, the best way to implement functionalities described in the design (Section 3.2.3) would probably be to send all data to the phone and let it be the one to do the calculations.

7.2 Personal conclusions

The realization of this project turned out to be a really good experience to me. It was the first time I faced a challenge of this characteristics and at first I really did not know what to expect.

I liked the fact that this particular project was an opportunity to put together some of the knowledge acquired in rather different courses during the past four years, aside from learning new things. Another thing that was appealing to me was that it consisted in something tangible. It is not as theoretical as other projects may be, but in this case you see the results of your work.

I specially enjoyed the building of the prototype itself. Programming the boards and developing the Android app to get the desired outcomes. The writing of this document was just the opposite of it. It took me quite longer than what I expected. I would have liked to have more time to try and improve the prototype, but finishing the document in time was more important.

7.3 Further development

After the completion of the project with the built prototype, there is room for future improvements. Some of them are related to the objectives set at the beginning that could not be satisfied by the prototype, while others are new ideas that came up during the process thanks to the way things evolved.

7.3.1 Sending the notification to more than one device

In the current prototype it is only possible to register one device to receive notifications. However, it may be the case that more than one person was interested in being notified whenever the tracked person left the controlled area.

As previously mentioned, decisions to facilitate the realization of this improvement were made while developing the initial prototype. However, some more adjustments would be needed, as the NodeMCU would have to be able to store (and receive) more than one registration token.

Aside from the known Access Point and Station modes there is another one called Softap mode when it acts both as and Access Point and a Station, which could be a possible solution for this issue.

7.3.2 App improvements

The user interface of the app could be redesign to offer a better experience. Tons of extra functionalities could be added, like allowing to configure the notifications settings, making it possible to delete all old notifications at once.

An app for iOS could also be implemented. This would be an obvious step forward when attempting to widen the target market. For the project an Android version was chosen because it was what was available at the moment, but Firebase Cloud Messaging also offers the option of developing and iOS client app.

7.3.3 Add more *tracked events*

The working prototype alerts the user whenever the tracked person leaves or enters the covered area. Initially, it was intended to also notify in case the tracked person stayed for a disturbingly extensive period of time in the same spot, which could imply a fall or a fainting among others. Unfortunately this led to problems with the NodeMCU memory, as will be explained in section 6.3.3, so it was decided to leave them on hold and try to find a way to make it possible in the future.

7.3.4 Add more *stations*

Due to limitations in resources (especially time and budget) this first prototype consists only of one *station*. Ideally, adding more of them would open the door to more possibilities. For example, proximity to *dangerous* zones, such as stairs, could be detected with three *stations* as the person's position could be triangulated.

To implement a way of communication between *stations* would be essential, in order to synchronize to send the proper notification according to the situation.

As a first thought, two possibilities for this: a central *station* connected to a NodeMCU responsible of sending all the notifications and computing the distances or every *station* connected to its NodeMCU sending their notifications and the receiving smartphone would be the one to make the computations. The first option would probably lead to the aforementioned NodeMCU memory problems.

7.3.5 Add more *tracking devices*

When doing this, the system could control more than just one person. This could be useful in a scenario where two people living together were not a 100% capable. Each of the *tracking devices* would have an unique identifier so that the proper notification could be sent when any of them left the covered area.

Either the central node or the receiving smartphone (as there are limitations with the NodeMCU memory) would need to store a list of the known subjects, as to differentiate which notification to send/display depending on the *tracking device* that triggered it.

A Acronyms

AP	**A**ccess **P**oint
API	**A**pplication **P**rogramming **I**nterface
ATT	**Att**ribute Protocol
BER	**B**it **E**rror **R**ate
BLE	**B**luetooth **L**ow **E**nergy
CRC	**C**yclic **R**edundancy **C**heck
FCM	**F**irebase **C**loud **M**essaging
GAP	**G**eneric **A**ccess **P**rofile
GATT	**G**eneric **Att**ribute Profile
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
ICT	**I**nformation and **C**ommunication **T**echnology
IEEE	**I**nstitute of **E**lectric and **E**lectronic **E**ngineers
IETF	**I**nternet **E**ngineering **T**ask **F**orce
IrDA	**I**nfra**r**ed **D**ata **A**ssociation
ISM	**I**ndustrial **S**cientific and **M**edical
L2CAP	**L**ogical **L**ink **C**ontrol and **A**ccess **P**rotocol
NFC	**N**ear **F**ield **C**ommunication
PDU	**P**rotocol **D**ata **U**nit
RFID	**R**adio **F**requency **Id**entification
RSSI	**Received** **S**ignal **S**trength **I**ndicator
SDK	**S**oftware **D**evelopment **K**it
SIG	**S**pecial **I**nterest **G**roup
SMP	**S**ecurity **M**anager **P**rotocol
SSID	**S**ervice **S**et **Id**entifier
UART	**U**niversal **A**synchronous **R**eceiver-**T**ransmitter
XMPP	Extensible **M**essaging and **P**resence **P**rotocol

B Code

https://github.com/Nacho-tfg/TFG

C Summary

This annex is included following the university regulation, which as of the academic year 2015/2016 requires that students pursuing a bachelor's degree in Telecommunication Technologies Engineering must include an extended summary of their final project appended as an annex.

Motivation and objectives

As the title of the project (Prototyping a low cost presence tracker for the elder) may suggest, it was done thinking about old people. More particularly, it aims to help people living with mental illnesses, many of them age related, such as dementia or Alzheimer's disease as a part of their lives. This includes many people, whether it is them who actually suffer from the disease or a person they care about.

For those belonging to the second group, the ones with loved ones suffering from the disease, it could be a great relief to know for sure that the person they care about was at home and did not leave without anyone knowing. It would take the burden of not knowing off of them and they could focus more in their lives. This is where the idea for the project described in this document started.

The existence of Bluetooth Low Energy technology and the possibility to use it for location services helped in deciding which direction to take for the project. There are systems in hospitals able to track medical equipment using this technology. Now, instead of this, the focus was set at more personal scenario. If a similar system could be developed for a home environment, it could be used to *track* the sick person so that their relatives could be certain they were actually at home.

When taking a look at the statistics, it is observed that the increasing of life expectancy has made the appearance of age related illnesses more and more common. Around 25 million people worldwide have dementia, and the number of people suffering from it is expected to surpass 80 million by 2040 [2]. In Europe, one of every three citizens will be over 65 by 2060. By that year the ratio of working people to retired people will be two to one, a big change compared to the current four to one ratio [3].

New technologies can be taken advantage of to help with these issues, and plans to promote their adoption have been deployed. There are projects like the EU-funded CommonWell that aim for a better integration between healthcare and social care services with the help of ICT solutions.

The project this document describes can be included as one that tries to put technology to the service of people.

Regarding the objectives set for this project, the main one was to study the possibility of providing a low-cost solution to the issue above described, by prototyping a presence tracking system taking advantage of BLE technology.

This system would send a notification whenever the *tracked* person left home. It would also inform in case the person stayed in the exact same spot for a long time, as it could be a bad sign; or if he or she got too close to any *dangerous* zones, such as stairs.

Taking into account the economic aspect, it is important to make it affordable for most people. It is the most likely scenario that people who may benefit from this system can not afford to expend a big amount of money in it. The intention was to provide them with a less expensive alternative to those systems hospitals use. Not only for the whole system, but special emphasis should be placed in the device the *tracked* person would be carrying, as it may be the case that the he or she decided to throw it away or lost it somewhere.

Another objective was to come up with something that was easy to use, making it launchable to the market targeting the general audience. It was important not to add any extra barrier that might prevent people from benefiting from it.

Prototype

Before deciding how to proceed with the solution to the problem, it is important to take a look at the regulatory limitations that may apply. This project has to fulfill two main legal compliance requirements. The first one is related to radio-frequency output power. Bluetooth technology transmits in the ISM unlicensed band. The regulation states that for adaptative frequency hopping equipment the maximum output power can never exceed the 20dBm (100mW) limit [5]. This requirement is completely fulfilled since manufacturers must satisfy it in order to certify their products, as it is specified in the Bluetooth Core Specification [6].

The second one is related to personal data protection. The data protection laws of each region must be taken into account as positioning techniques are a way to obtain location data, which is personal data.

After designing the system, it was time for the actual building of a prototype. Three separate parts or elements can be distinguished in the implemented final prototype: the *tracking device*, a *station* and an Android app. The *tracking device* is a BLE Nano board acting as an advertiser, the *station* consists of another BLE board, this one in scanning mode, and a NodeMCU devkit. This last one is the one in charge of sending the notification whenever the *tracked person* leaves or arrives home. The reason why the prototype counted with just one *station* is the obvious time and budget limitation present during the development.

The system needs to be configured before using it, as to connect the NodeMCU to a Wi-Fi network. This is easily done from the same Android app the notifications will be sent to. Once setup, every time the *tracking device* leaves or arrives to the covered area a notification will be sent informing the interested user about it. In the app, a list of all the received notifications is maintained, being the user able to delete any of them as he or she will.

Tests and results

Before finalizing the project, the prototype had to undergo two different tests to check its proper behavior. The first one consisted on checking that the *station* was able to detect when the *tracking device*'s signal completely disappeared. The second one was designed as to simulate a real life situation. The area to cover needed to be reduced and the Received Signal Strength Indicator was used to that end.

After taking several measures of the RSSI at the edge of coverage, it was decided that the best option would be to set two thresholds. An upper one, that when surpassed an *Arrived* notification would be sent; and a lower one, that would trigger the sending of the *Left* notification.

The first test outcomes demonstrated that the system was able to detect when the *tracking device* was out of range. It did so with a 100% reliability. So this proved that implementing a reliable presence tracker taking advantage of BLE technology can be done.

However, the results obtained from the second test show that there is a real difficulty when trying to determine the radius of coverage. Because of the continuous changes in the signal's strength, it was almost impossible to guarantee that the system would always cover the same area.

The *Left* notification was sent everytime the *tracking device* left the area. The problem resided in the big variation of the position at which the *tracking device* was when the notification was sent. Also, sometimes the notification was sent when standing close to the edge of the covered area, without the person actually leaving. This same thing happened for the *Arrived* notification.

Also, if the person stood in the edge of coverage, a lot of alternating *Left* and *Arrived* notifications would be sent, which can be very annoying for the receiver.

Conclusions

With the realization of this project and analyzing the commented results, one can see that it has been proven that it is indeed possible to implement a presence tracking system taking advantage of BLE technology. At least one counting with only one *station*, as the built prototype.

Reviewing the goals set at the beginning, most of them were fulfilled. Those that were not was due to the limitations during the development and are left as future improvements.

An important point in the realization of this project was to manage to build a system like this at a low cost. This was also satisfied, as the hardware's cost went to a bit over 60€. The tracking device itself, whick is probably the most critical in terms of cost, is less than 20€. This is an acceptable price to pay in case it got lost or damaged, which is not unlikely as the sick person would carry it around and could forget or lose it anywhere.

Another of the objectives set at the beginning was to obtain a final product that was easy to use. This was fulfilled too, as the resulting prototype is quite user friendly and could be used by anyone, without needing to learn anything new. The setup process is very similar to connecting to a Wi-Fi network. All that is needed is downloading the Android app and tapping a few buttons. The system takes care of storing the registrationIDs and similar, allowing the user to have a smoother experience.

However, it has to be said that, as the results of the tests showed, the outcomes it provides may not be the same always. Variations in RSSI may make the system send the notification at a moment that it normally would not. This only happened when standing on the edge of the covered area. In these cases an *Arrived* notification was received right after a *Left* one. If the person left the zone, the system responded as expected.

Nevertheless, the prototype as is does work. It can be concluded that the project was a success even though there are a few improvements left for future development.

Bibliography

[1] IKNAIA. Healthcare. http://www.iknaia.co.uk/iknaia-tracking-patients-and-staff/. [Last access: February 2016].

[2] Cleusa P Ferri. Global prevalence of dementia: a delphi consensus study. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2850264/. [Last access: September 2016].

[3] European Comission. Policies for ageing well with ict. https://ec.europa.eu/digital-single-market/en/policies-ageing-well-ict. [Last access: September 2016].

[4] European Comission. The eu-funded project commonwell: how to integrate health and social care with ict. https://ec.europa.eu/digital-single-market/en/news/eu-funded-project-commonwell-how-integrate-health-and-social-care-ict. [Last access: September 2016].

[5] ETSI. Final draft etsi en 300 328. http://www.ietf.org/mail-archive/web/6tsch/current/pdfd3d1acPkgu.pdf, 2012. [Last access: July 2016].

[6] Bluetooth SIG. *Specification of the Bluetooth System*, December 2014. Core version 4.2.

[7] JIMB0. Bluetooth basics. https://learn.sparkfun.com/tutorials/bluetooth-basics. [Last access: June 2016].

[8] Bluetooth technology website. markets. https://www.bluetooth.com/marketing-branding/markets, 2016. [Last access: June 2016].

[9] Bluetooth SIG. Compare with other technologies. https://developer.bluetooth.org/TechnologyOverview/pages/compare.aspx, 2016. [Last access: June 2016].

[10] Ibrahim F. Tarrad Ahmed H. Ali, Reham Abdellatif Abouhogail and Mohamed I. Youssef. Assessment and comparison of commonly used wireless technologies from mobile payment systems perspective. http://www.sersc.org/journals/IJSEIA/vol8_no2_2014/25.pdf, 2014. [Last access: June 2016].

[11] Joaquim Oller Carles Gomez and Josep Paredells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. http://www.mdpi.com/1424-8220/12/9/11734/htm, 2012. [Last access: June 2016].

[12] Mikhail Galeev. Bluetooth 4.0: An introduction to bluetooth low energy—part ii. http://www.eetimes.com/document.asp?doc_id=1278966&, 2011. [Last access: June 2016].

[13] Bluetooth SIG. Bluetooth® 5 quadruples range, doubles speed, increases data broadcasting capacity by 800 https://www.bluetooth.com/news/pressreleases/2016/06/16/-bluetooth5-quadruples-rangedoubles-speedincreases-data-broadcasting-capacity-by-800, 2016. [Last access: August 2016].

[14] ABI research. Bluetooth 5 evolution will lead to widespread deployments on the iot landscape. https://www.abiresearch.com/press/bluetooth-5-

evolution-will-lead-widespread-deploym/, 2016. [Last access: September 2016].

[15] Joakim Lindh. Bluetooth® low energy beacons. http://www.ti.com/lit/an/swra475/swra475.pdf, 2015. [Last access: June 2016].

[16] Vincent Gao. Proximity and rssi. http://blog.bluetooth.com/proximity-and-rssi/, 2015. [Last access: June 2016].

[17] Mathuranathan. Fundamentals of beacon ranging. http://developer.radiusnetworks.com/2014/12/04/fundamentals-of-beacon-ranging.html, 2013. [Last access: June 2016].

[18] David G. Young. Log distance path loss or log normal shadowing model. http://www.gaussianwaves.com/2013/09/log-distance-path-loss-or-log-normal-shadowing-model/, 2014. [Last access: June 2016].

[19] RedBearLab. Ble nano. http://redbearlab.com/blenano/, 2015. [Last access: April 2016].

[20] NodeMCU Team. Nodemcu connect things easy. http://nodemcu.com/index_en.html, 2014. [Last access: April 2016].

[21] Google. Firebase cloud messaging. https://firebase.google.com/docs/cloud-messaging/, 2016. [Last access: July 2016].

[22] Google. About firebase cloud messaging server. https://firebase.google.com/docs/cloud-messaging/server, 2016. [Last access: July 2016].

[23] Google. Set up a firebase cloud messaging client app on android. https://firebase.google.com/docs/cloud-messaging/android/client, 2016. [Last access: July 2016].

[24] Google. Receive messages in an android app. https://firebase.google.com/docs/cloud-messaging/android/receive, 2016. [Last access: July 2016].