Bachelor’s Thesis

Degree Programme in Information Technology

Internet Technology

2016

Milan Herrera Vargas

# INDOOR NAVIGATION USING BLUETOOTH LOW ENERGY (BLE) BEACONS

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology | Internet Technology

2016 | Total number of pages: 54

Instructor(s): Tero Virtanen, Jari-Pekka Paalassalo

Milan Herrera Vargas

# INDOOR NAVIGATION USING BLUETOOTH LOW ENERGY (BLE) BEACONS

The popularity of indoor navigation has significantly increased during the last decade. Nowadays, many studies aim to develop new indoor navigation systems and improve the accuracy of the already existing ones. Unfortunately, no definitive method for indoor navigation has been approved yet, which is why this type of systems are not as accessible as those used for outdoor navigation.

The purpose of this thesis is to introduce the concept of indoor navigation and demonstrate how common electronic devices can be used to create a simple indoor navigation system. In order to do this, a mobile Android application that helps users navigate inside a building was developed. Signal readings and other information broadcasted by Bluetooth Low Energy (BLE) devices, known as beacons, were used as reference for this application. In addition to this, adapters and sensors integrated into the mobile device were used to track the user's movement. The application is also capable of providing information to the user related to their location using additional data attachments associated to the beacons.

This application is only a prototype meant to provide an idea of how indoor navigation could be possible using no more than a smartphone and few BLE beacons. Even though the system proved to work quite well within the testing area, there are still plenty of aspects that can be improved.

KEYWORDS:

Indoor navigation, Bluetooth Low Energy, beacons, Android, mobile application, Google Eddystone

# CONTENTS

# PICTURES

## FIGURES

## TABLES

# LIST OF ABBREVIATIONS (OR) SYMBOLS

| | |
|---|---|
| BLE | Bluetooth Low Energy |
| GPS | Global Positioning System |
| IoT | Internet of Things |
| RF | Radio Frequency |
| RSSI | Received Signal Strength Indicator |
| IEEE | Institute of Electrical and Electronics Engineers |
| AP | Access Point |
| GHz | Gigahertz |
| MHz | Megahertz |
| dBm | Decibel-milliwatt |
| FSFM | Free Space Friis Model |
| API | Application Programing Interface |
| HTTP | Hypertext Transfer Protocol |
| JSON | JavaScript Object Notation |

# 1 INTRODUCTION

Street navigation systems that rely on Global Positioning System (GPS) satellites are used by many people every day. Unfortunately, this kind of technologies can only be used to navigate in open spaces and are not available indoors. It is because of this that indoor navigation techniques using Wi-Fi and Bluetooth signals, along with effective positioning algorithms, have been an object of study in recent years.

Ideally, deploying an indoor navigation system must be easy and cost effective. Most of the time, signals received from Wi-Fi devices present in a building are used as reference, however, these are not meant to be used for this purpose. It is because of this that other technologies better suited for indoor navigation, such as Bluetooth Low Energy (BLE), may be a good alternative.

BLE is a subsystem of the traditional Bluetooth technology capable of broadcasting data using a minimal amount of power. This makes it ideal for devices operating on small batteries which need to function uninterrupted for long periods of time. For the purpose of indoor navigation, BLE devices known as beacons seem to be the best choice.

Beacons are small devices that broadcast packets of data in short time intervals. These packets contain information about the beacon, as well as telemetry readings commonly used in distance calculations. Mobile devices such as tablets and smartphones can be used to pick up BLE signals and read the broadcasted data, which in turn can be used on indoor navigation applications.

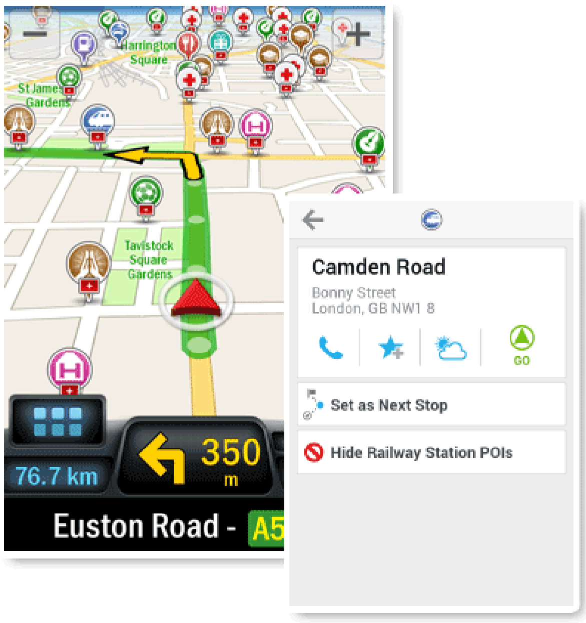
The popularity of beacons, as well as indoor positioning systems that rely on them, has increased notably during the last couple of years. Beacons are cheap when bought in large quantities and easy to set up. In addition to this, Apple and Google have developed dedicated beacon protocols which make managing and communicating with these an easy task.

For this project, an Android application that helps users navigate inside a building and receive contextual information, was developed. This application relies on information gathered from beacons and sensors integrated into the mobile device to function. The main reason for developing this application is to better understand indoor navigation and show how smartphones can be used for this purpose.

The structure of this document is as follows: Chapter 2 introduces the concept of micro-location, also referred as indoor navigation. In Chapter 3, the most common technologies used for this purpose, Wi-Fi and Bluetooth, are discussed. Special focus is given to the Bluetooth Low Energy (BLE) standard, which is the technology used in this project. In Chapter 4, an overview of BLE Beacons is given, including information about the most popular beacon protocols: iBeacon and Eddystone. Chapter 5 presents the most common indoor navigation methods as examples of how wireless signals can be used in this kind of systems. In Chapter 6, important aspects about the behaviour of radio signals and their impact on distance calculations are also covered. In Chapter 7, the mobile application is presented, along with an explanation of how the concepts mentioned throughout the document are implemented into it.

# 2 THE CONCEPT OF MICRO-LOCATION

During the last decade, the concept of location and navigation in outdoor areas has become fairly common. An example of this is the use of Global Positioning System (GPS) signals to navigate the streets of a city. With the large increase in mobile device production, more people have access to mobile applications developed for this purpose. These applications are usually free of charge and provide attractive and easy to use interfaces, aimed to everyone regardless of their technical knowledge. In addition to the navigation system, services designed to provide additional information, mostly related to places like shops and restaurants, are implemented into the application. One popular GPS-based application, with over 4 million downloads in the Android marketplace, is CoPilot (Picture 1).



Picture 1. Screenshot of GPS application CoPilot. Source: copilotgps.com

On the other hand, micro-location, also referred as indoor navigation, is still considered to be in its early development stages. One might think that by now, with all the advances in geolocation, such a simple concept as navigating inside a building would have been already addressed. This does not mean there has not been extensive research going on for several years, but no definite method for such purpose has been agreed upon yet.

In order to find out one’s position inside an enclosed space, some kind of reference is needed. GPS satellite signals cannot be received indoors, which means that indoor positioning systems must rely on other kinds of technologies to function. Usually, reference data in the form of radio signals, infrared, ultrasound or magnetic field readings [1] is used for this purpose. Once collected, this data can be interpreted and used in positioning algorithms, providing different levels of accuracy.

For the purpose of developing an indoor navigation system that is easy to implement, as well as widely available to the public, one must consider using common and simple technologies. Positioning methods such as those using magnetic field readings [2] offer great accuracy but their implementation is difficult and pricy, not to mention, they cannot be easily made available to the public. On the other hand, Radio Frequency (RF) signals, such as those emitted by Wi-Fi Access Points (AP) and Bluetooth devices, are fairly common and may be also used as reference for indoor navigation systems. Nowadays most mobile devices are equipped with both Wi-Fi and Bluetooth adapters which enable them to pick up these signals and connect, if necessary, to the transmitting devices. By taking advantage of this reality, mobile applications aimed to indoor navigation can be developed and made available to everyone through mobile marketplaces.

Just as those relaying on GPS, indoor navigation applications can be also designed to provide additional information, either on request or automatically, related the user’s location. A company building or university, for example, can offer guidance to visitors about the location of a classroom or conference room, including schedules, features, etc. Another good example is an airport that chooses to provide information about flights as soon as the clients walk through the door, as well as ways to check-in and find their luggage. The list goes on and on and the possibilities become more interesting.

Being able to provide contextual information can be highly valuable not just for the user but also for companies and services that choose to implement this system. Stores, for instance, can provide information about certain products on sale or available promotions. They can also track, not intrusively, customers to learn about their shopping habits and identify popular products in order to provide a better service. Whether they are for commercial purposes or not, the mobile applications should always ask permission from the user and inform them about any information being collected.

The next chapter provides an introduction to the two most used technologies in indoor navigation, Wi-Fi and Bluetooth. Understanding these technologies is essential in order to choose which one to implement into a mobile application, or any other kind of micro-location system.

# 3 TECHNOLOGIES USED FOR INDOOR NAVIGATION

## 3.1 Wi-Fi

Wi-Fi is a wireless technology that enables users to connect to the network through devices known as Access Points (AP). It operates mainly over the 2.4 GHz and 5 GHz “Industrial Scientific and Medical” (ISM) frequency bands and has been standardized by the Institute of Electrical and Electronics Engineers (IEEE) as 802.11. An example of a Wi-Fi AP is shown in Picture 2.

Picture 2. Linksys WAP54G Wireless Access Point. Source: Linksys.com

Throughout the years, many different standards of the Wi-Fi protocol have been developed, providing network throughput improvements over their successors. The first official standard of the protocol was the 802.11b followed by 802.11g, 802.11n and 802.11ac. Both “b” and “g” versions function on the 2.4 GHz ISM band and are the most widely used. They use 20 MHz of bandwidth to transmit over 14 overlapping channels 22 MHz in length, three of which (1, 6 and 11) do not overlap and are the most commonly used. All available 2.4 GHz channels are depicted in Figure 1.

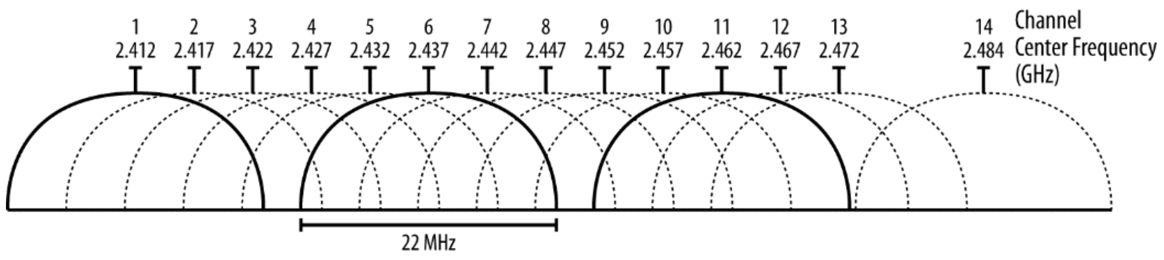


Figure 1. Wi-Fi channels on the 2.4 GHz frequency band. Source: wikipedia.org

The remaining Wi-Fi standards work only on the 5 GHz band, except for the newest 802.11ac which can work on both 2.4 GHz and 5 GHz bands. They double the bandwidth from 20 MHz to 40 MHz per channel and support multiple antennas [3] but are not as widespread as the 2.4 GHz standards.

The maximum range Wi-Fi can achieve is determined by several factors including: type of antenna, signal interference caused by obstacles present in the environment and the output power of the AP. The output power achieved by an AP, or any other device that uses RF signals, is measured in dBm (decibel-milliwatt) which is the amount of power referenced to one milliwatt. This value usually ranges from -30 dBm to 30 dBm and determines how far transmitted signals can travel. For example, the range of an 802.11b/g AP can reach up to 100 m [4] in an open space transmitting at 30 dBm, the maximum transmit output power allowed by the Federal Communications Commission (FCC) [5] in the USA.

Due to their high availability, Wi-Fi signals are the most used in indoor navigation. Usually, a large quantity of wireless APs are already present inside places where these kinds of systems are implemented, providing enough reference points. One fact to have in mind, though, is that originally Wi-Fi has not been designed to be used in this way. Every time a scan occurs, queries are sent to the APs requesting the necessary information which may hinder the network performance. In addition, APs might belong a third party which does not allow their use as such due to security policies. Due to these disadvantages, other kinds of devices, more adaptable to the purpose of indoor navigation, could be used. Bluetooth devices, for example, might offer a better solution.

## 3.2 Bluetooth

Bluetooth is a wireless communication technology that allows electronic devices to communicate over small distances and, just as Wi-Fi, operates on the 2.4 GHz ISM frequency band. Despite operating on an open band, the Bluetooth technology is regulated by the “Bluetooth Special Interest Group” (SIG) and was standardized by IEEE as IEEE 802.15.1, even though it no longer maintains the standard. Common examples of Bluetooth devices are mobile headsets, game controllers, wireless keyboards and mice.

Bluetooth uses 79 channels to transmit data, starting with the first channel at a frequency of 2402 MHz and continuing up to the last one at a frequency of 2480 MHz in 1 MHz increments. In order to avoid interference from other RF signals, a technique known as frequency hopping [6], where data is transmitted over one of the available channels for a small period of time and resent over another channel in case of interference, is used.

In order to transmit data, Bluetooth devices must first stablish a connection. One single device is capable of connecting to up to 7 devices and communicating with each one of them simultaneously. This is done by using a connection model known as “master-slave”, in which the device that initiates the connection takes the role of master over the other devices. Whenever a master and a slave establish a connection, a bond is created, enabling them to transmit and receive data.

Designed as a low-power technology, Bluetooth is meant to function on battery power devices over relatively short distances. Just as with Wi-Fi, the maximum distance over which a Bluetooth device can stablish a connection depends on its output power, as well as other factors such as signal reflection caused by obstacles. Three different classes are used to classify Bluetooth devices according to their maximum output power [6] and their corresponding maximum range. These can be seen in Table 1.

Table 1. Bluetooth classes and their corresponding ranges

| Class | Maximum Output Power | Range |
|---|---|---|
| 1 | 20 dBm | up to 100 m |
| 2 | 4 dBm | up to 10 m |
| 3 | 0 dBm | 10 cm |

Bluetooth has been also known to work for indoor navigation purposes [7], offering less power consumption than Wi-Fi and a wider range of devices. Even though its main purpose is to transmit data between connected devices, in the case of indoor navigation, a connection might not be always necessary. Since no data is needed to be exchanged between devices, having reference devices that only broadcast information related to their position would be enough for this purpose.

During the last decade, Bluetooth sensors designed to only broadcast data without the need of a connection have become very popular, especially in the fields of healthcare and sports. Common examples of these are activity wristbands like the FitBib Charge HR, shown in Image 3, normally used as step counters and heart rate monitors in sports.

Picture 3. FitBib Charge HR Activity wristband

In addition to these kinds of devices, a new communication trend known as the Internet of Things (IoT) began encouraging the use of small sensors equipped on everyday objects, enabling them to communicate with each other and their users as an integral part of the Internet [8]. Smaller sensors running on smaller power sources started emerging and the need for an even lower power consumption option became evident. In answer to this need, a new Bluetooth standard named Bluetooth Low Energy (BLE), also known as Bluetooth Smart, was introduced.

### 3.2.1 Bluetooth Low Energy (BLE)

BLE is a subsystem of the conventional Bluetooth 4.0 protocol, designed to provide device communication with minimal power consumption. The number of transmission channels has been reduced from 79 to 40 2 MHz wide channels, 3 of which (37, 38 and 39) are used for advertising. These three channels have been chosen to not collide with the most commonly used Wi-Fi channels (1, 6 and 11), as shown in Figure 2, decreasing interference considerably [9].

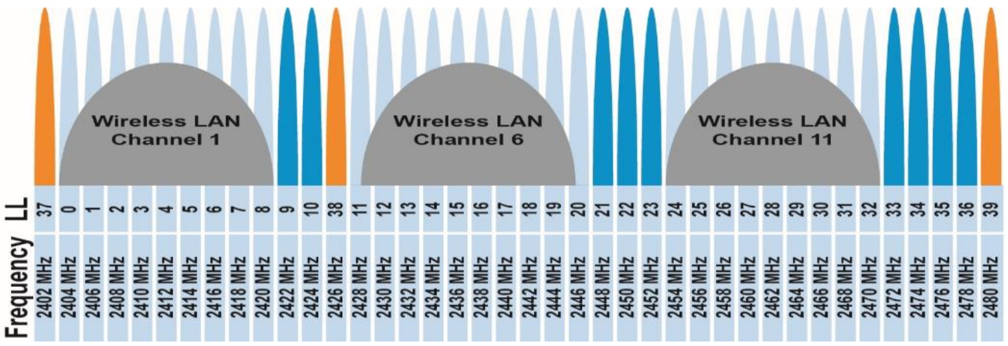

Wireless LAN Channel 1
Wireless LAN Channel 6
Wireless LAN Channel 11
LL
37 0 1 2 3 4 5 6 7 8 9 10 38 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 39
Frequency
2402 MHz 2404 MHz 2406 MHz 2408 MHz 2410 MHz 2412 MHz 2414 MHz 2416 MHz 2418 MHz 2420 MHz 2422 MHz 2424 MHz 2426 MHz 2428 MHz 2430 MHz 2432 MHz 2434 MHz 2436 MHz 2438 MHz 2440 MHz 2442 MHz 2444 MHz 2446 MHz 2448 MHz 2450 MHz 2452 MHz 2454 MHz 2456 MHz 2458 MHz 2460 MHz 2462 MHz 2464 MHz 2466 MHz 2468 MHz 2470 MHz 2472 MHz 2474 MHz 2476 MHz 2478 MHz 2480 MHz

Figure 2. BLE channels

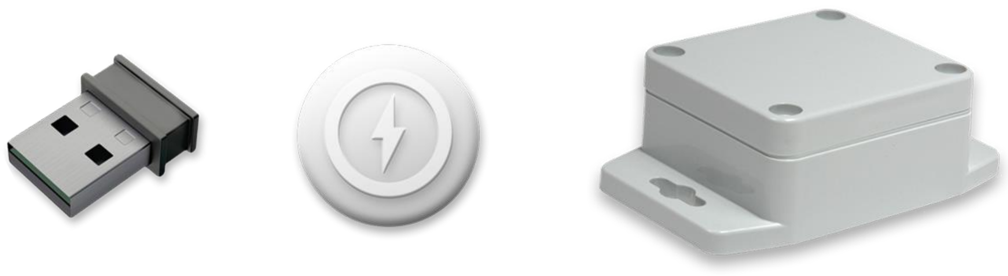
Just as classic Bluetooth, BLE devices can operate using several output power settings which can be adjusted according to the range that is needed. A low output power (between -30 dBm and -12 dBm) is normally chosen if the distance between the broadcaster and receiver is not too long (between 1 – 15 m). On the other hand, higher power might be needed if the distance increases (over 20m) although this will also decrease the battery charge much faster.  The most common value used by BLE devices is 0 dBm which offers a range of around 50 m [10] in a non-obstacle environment. Higher values increase the signal range considerably, up to a theoretical 300 m with a 10 dBm value, the maximum supported by BLE [11]. These optimized ranges differed from the classic Bluetooth ones, mentioned in Table 1, and are achieved thanks to a different modulation technique used by BLE.

BLE was developed for the purpose of broadcasting data without the need of a connection. BLE devices are normally used in one-way communications where receivers pick up data passively and are unable to stablish connections with the broadcaster. Due to this, data on the device cannot be manipulated, except for specific cases in which dedicated USB cables can be used to upload firmware. As the production of BLE devices increased, the need for configurable devices became a priority. There are many cases in which the device must be configured in order to fulfil a certain purpose, which is why most of BLE devices nowadays are connectable as well. This allows other devices to connect to them, modify their configuration, and update their firmware wirelessly. In cases where this feature might cause any security problem, most devices can be protected with passwords required to modify their configuration.

Taking into account all the features BLE has to offer, devices using this technology seem like the best option for indoor navigation. Furthermore, they have been proven to work fairly well for this purpose [12, 13]. In these examples, small BLE devices, known as beacons, are normally used. These devices are limited to broadcasting data that can be used as reference for positioning, making them ideal for this kind of applications. More information about these is provided in the next chapter.

# 4 BLUETOOTH LOW ENERGY (BLE) BEACONS

Beacons are BLE devices that broadcast small data packets at regular time intervals and commonly operate on coin-cell batteries although other power options, such as AA batteries and USB, are also available. Relying on these power sources and thanks to the BLE technology, beacons can operate uninterrupted for months or even years. In addition to this, their small size makes them easy to carry and install. Examples of the beacons available in the market are shown in Picture 4.



Picture 4. Examples of beacons. Source: radiusnetworks.com

Beacons achieve optimal power consumption by remaining asleep most of their operating time, only waking up to broadcast at predefined intervals [9]. These intervals can range from a couple of seconds to 100 ms or less and can be adjusted depending on the beacon's purpose. For example, a shorter broadcast interval increases the amount of packages being broadcasted, improving the accuracy of the readings but at the cost of higher power consumption. Advertising intervals values, as well as other beacon configurations, can be configured using software tools provided by the beacon manufacturer.

The data broadcasted by a BLE beacon is contained into packets formatted according to the Bluetooth Core Specification [14]. How these packets are formatted, along with all its components is shown in Figure 3.

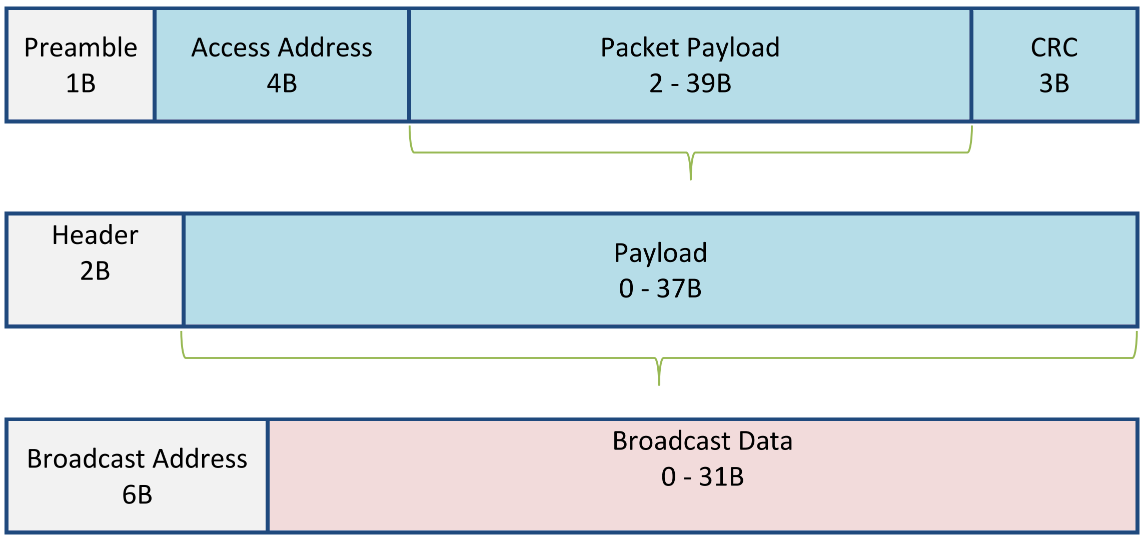


Figure 3. BLE packet structure

Looking at the BLE packet, we can see that the data being broadcasted is limited to only 31 bytes, consisting mostly on short text strings and numerical values. Due to this constraint, special beacon protocols are used to format the data field, optimizing as much as possible the available space. In addition to this, these protocols offer additional tools that can be used to attach additional data to the beacon using cloud services.

## 4.1 Beacon Protocols

Beacon protocols format the data field by splitting it into small segments containing beacon information, such as signal strength, output power, etc. This information is represented in hexadecimal values which can be interpreted using code and tools provided by the protocol developers. The most popular beacon protocols are iBeacon and Eddystone.

### 4.1.1 iBeacon

The iBeacon protocol, developed by Apple in 2013, was the first beacon technology available. It started as way of turning iOS devices into advertisers that can transmit data to other listening iOS devices. This system was also used to estimate the distance between broadcasting and receiving devices [15]. Later on, dedicated BLE Beacons were introduced, eliminating the necessity of a mobile device.

The iBeacon protocol makes use of the data field in a BLE packet as shown in Figure 4.

| iBeacon Data Field 31 Bytes | | | | |
|---|---|---|---|---|
| iBeacon Prefix 9B | UUID 16B | Major 2B | Minor 2B | TX Power 1B |

Figure 4. iBeacon data field

The iBeacon frame contains a field called UUID used to identify a specific application related to the beacon. This means that every beacon related to one application will have the same ID.  Major and Minor values are mainly used to identify beacons when deployed into small subgroups. In this case, the Major value is used to identify each subgroup and the Minor one to identify specific beacons within the subgroup. The TX Power field represents the beacon transmit power calibrated at 1 m, used in distance calculations. This value depends on the hardware and it is calibrated by measuring the received signal strength at the distance of 1 m in an obstacle-free environment.

When implementing the iBeacon protocol certain requirements must be followed. For instance, iBeacon requires the beacon to transmit its data every 100 ms (ten times every second) assuring their availability. Unfortunately this is not always necessary and might increase the beacon's power consumption considerably, thus decreasing its battery life. Also, a unique application must also be coded for each beacon deployment. This is the case, for example, for a clothes store that has its own application that works only with its associated beacons. However, this is not always good since users must download a new application for each place they visit [16].

Another disadvantage of iBeacon is its private distribution. This means that, in order to use the protocol and its related APIs, including those used to calculate the approximate distance between devices, one must create a paid Apple developer account and register to their MFi program [17]. Moreover, despite their claims of being cross platform, the development of mobile applications implementing iBeacon is somehow limited to Apple products only.

### 4.1.2 Eddystone

Eddystone is an open beacon protocol developed by Google which features numerous advantages over iBeacon, most important of all being its open source license.

The Eddystone data field, shown in Figure 5, contains the same information as the iBeacon one, only formatted in a different way.

| Eddystone Data Field 31 Bytes | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Len 0x02 | Type 0x01 | Flags 0x06 | Len 0x03 | Type 0x03 | Eddystone UUID 0xAA, 0xFE | Len 0x?? | Type 0x16 | Eddystone UUID 0xAA, 0xFE | Eddystone Frames Up to 20B |

Figure 5. Eddystone data field

Additionally, different kinds of Eddystone frames are available, depending on the beacon's purpose [18].

There are three kinds of Eddystone frames, the most common of them being the UID frame [19], shown in Figure 6.

| Eddystone UID Frame 20 Bytes | | | | |
|---|---|---|---|---|
| Frame Type 0x00 1B | TX Power 1B | Namespace ID 10B | Instance ID 6B | RFU 2B |

Figure 6. Eddystone UID frame

The first field, included in all Eddystone frames, indicates the frame type. Just as with iBeacon, the TX Power field is also present but, in case of Eddystone, this value is calibrated at a distance of 0 m. The two fields called Namespace ID and Instance ID represent the same data as the iBeacon Major and Minor values. In the case of Eddystone, however, the Namespace ID will be unique per beacon deployment and the Instance ID will be unique for each beacon device.

Eddystone also has another kind of frame type called URL [20]. The data in this kind of frame, as its name suggests, is a URL of up to 17 bytes in length. Any device passing by, with the necessary functionality built in to it, can pick and interact with this URL. This field is represented in Figure 7.

| Eddystone URL Frame 6 - 20 Bytes | | | |
|---|---|---|---|
| Frame Type 0x10 1B | TX Power 1B | URL Scheme Prefix 0x00 = http://www. 0x01 = https://www. 0x02 = http:// 0x03 = https:// 1B | Encoded URL up to 17B |

Figure 7. Eddystone URL frame

The third packet format is called TLM [21] and can be seen in Figure 8. This frame is intended for broadcasting telemetry data related to the beacon device. This can be battery voltage, beacon temperature, advertisement count and elapsed seconds since power-on. Because this packet does not contain any beacon ID, it must be paired with another frame that contains this information (either UID or URL). In this way the frames containing the ID and telemetry frames can be matched up using the device address.

| Eddystone TLM Frame 14 Bytes | | | | | |
|---|---|---|---|---|---|
| Frame Type 0x20 1B | Version 0x00 1B | Battery Voltage (mV) 2B | Beacon Temperature 2B | Advertisement PDU Count since boot 4B | Time Since Boot (milliseconds) 4B |

Figure 8. Eddystone TLM frame

As mentioned before, one of the advantages of Eddystone over iBeacon is licensing. An iBeacon licensed device is not truly cross platform because the code needed to understand that iBeacon format is exclusive of iOS devices, despite some beacon manufacturers claiming to have reverse-engineered the protocol to find a way around this. Eddystone is intentionally cross platform and all the necessary code needed to take full advantage of this protocol is provided for free by Google in their GitHub repository [18].

As opposed to iBeacon, Eddystone does not dictate how often frames should be transmitted. Typically Eddystone devices broadcast in intervals of between 1 and 2 seconds, which is especially important if the beacon is battery powered. Another important point is that users do not need to install a new application for each place they visit. Eddystone makes it possible for a single application to manage multiple beacon deployments and with the use of additional services, such as Google Places, information related to popular venues can be delivered and user reviews collected [16]. There is also available a specific beacon Application Programing Interface (API) which allows developers to associate data with their beacons and access it easily [22].

The next chapter provides examples of how the previously discussed technologies and devices are used for indoor navigation.

# 5 INDOOR NAVIGATION METHODS

RF signals broadcasted by wireless devices are used differently depending on the positioning method being implemented. During the last years, several of these methods have been developed [23], the most common of these being Fingerprinting and Triangulation. Before moving on to explaining these positioning methods, it is important to understand how RF signals can be measured in order to be used as reference.

Signal strength measurement is made by using a signal parameter known as Received Signal Strength Indicator (RSSI). As it name suggests, RSSI is the measurement of the power present on a received radio signal expressed in dBm [24] ranging from -100 dBm for a very low signal level to -50 dBm or more for a strong one. This value should not be confused with output power, which is measured using the same units but represents something totally different (see Chapter 3).

Some positioning methods collect RSSI values as they are while others use them, along with additional information, to calculate the distance between transmitting and receiving devices. The results of these calculations tend to be inaccurate, since the relationship between RSSI and distance is not always the same and changes according to many factors.

For the purpose of this thesis, RSSI is used to measure the distance between the transmitting beacons and the user. As mentioned before, these calculations are not very accurate but give an idea of how far one is from the other. Due to the complexity of these calculations, Chapter 6 is completely dedicated to them.

## 5.1 Fingerprinting

Fingerprinting uses the RSSI values of a group of devices to create a signature (Fingerprint) of a specific location. This is done by storing these values, along with the addresses of their corresponding devices, in a database. Once several Fingerprints of different locations are created, continuous scans are performed and a runtime Fingerprint is generated every time. This last one is then compared to each one of the saved Fingerprints in order to obtain the closest match which represents the location where the user is.

Fingerprinting offers reasonable accuracy and is easy to implement. In the case of Wi-Fi [25], existing APs in a building are used. One disadvantage, however, is that the availability of the APs used to create the Fingerprint cannot be assured. Even though this issue can be easily addressed by replacing missing RSSI values with arbitrary measurements, this could decrease the accuracy significantly, especially if more than one of these values are missing.

BLE Beacons are also used for Fingerprinting [12]. They can be positioned strategically in order to obtain as many necessary readings as possible and offer a lower power consumption. Having dedicated devices for this purpose is definitely favourable, however, a large quantity of them must be purchased in order to cover a large area.

Regardless of the type of signal being used, Fingerprinting has some drawbacks. First of all, the collection of Fingerprints around a building is time-consuming. When using Beacons, a survey of the building is needed to evaluate the best locations to place them and, in the case of Wi-Fi, the places where the samples for the Fingerprint must be collected. Changes in the building can also affect the signal readings and many times the Fingerprints need to be re-calibrated.

## 5.2 Triangulation

The triangulation method uses RSSI readings in a different way from Fingerprinting. Instead of only limiting to collect these values, these are also used to calculate the distance between the broadcasting and receiving devices. For this method to work, three or more reference devices must be positioned in a way they cover an area. Next, distances to the reference devices are roughly calculated by the receiving device and an interception point between these is found, which is usually where the user is located [7]. The simplified concept of this method is shown in Figure 9.

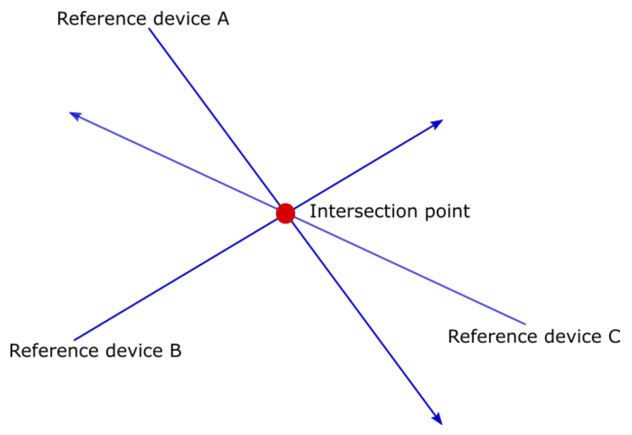


Figure 9. Concept of Triangulation by distance interception

Another version of this method uses imaginary circles drawn around the reference devices in order to find interception points between them. The radii of these circles is calculated based on the RSSI readings picked up from each of the reference points and the time it takes for the signal to travel to the receiving device [7]. Once three or more of these circles overlap, the interception points between them can be found and it is between these points were the user usually is. A simple representation of this method is shown in Figure 10.

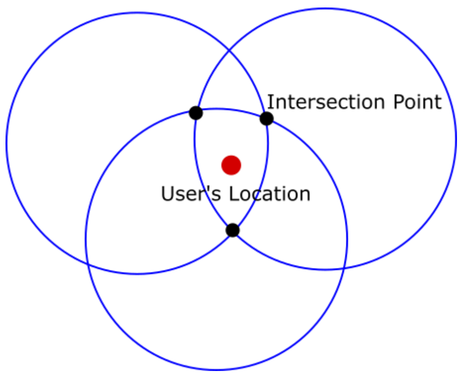


Figure 10. Concept of Triangulation using multiple interception points

Both the Fingerprinting and Triangulation methods were considered for this thesis, since they are easy to implement and have been proven to work in other positioning systems referenced at the beginning this chapter. However, one disadvantage is that the necessary amount of reference devices would have to be quite high. In Fingerprinting, for example, as many RSSI samples as possible will be needed, otherwise the accuracy will decrease considerably. As for Triangulation, as the user moves along, let us say a corridor in a university, many devices would have to be positioned and the complexity of the algorithm will increase in order to calculate the user’s position at all times.

Even though neither of the previously explained methods will be used for this project, some of their characteristics will still be taken into account to develop a new and simple positioning algorithm. This will be achieved by using data provided by beacons and the sensors integrated into the phone. All aspects related to this algorithm, as well as detailed information about the application in which it will be used, is covered in Chapter 7.

The next chapter covers some important concepts regarding RF signal behaviour, and proposes a way to calculate distances using RSSI readings. These calculations are the core of the application being developed and need to be understood before moving on to the actual application.

# 6 DISTANCE CALCULATIONS USING RF SIGNALS

As mentioned in the previous chapter, RSSI values can be used in distance calculations. These values normally change depending on how far the transmitting device is. If we compare the transmitted signal strength to the received one, we can somehow measure how much the value decreased over the travelled distance and use this information to know the distance between devices. Nevertheless, due to many factors that may affect the signal on its way from one point to another, the relation between distance and RSSI is not always constant.

For the purpose of this thesis, a method to calculate the distance between the reference devices and the user must be found. Unfortunately, this is not as simple as it sounds and several factors must be taken into account when trying different methods. Due to the characteristics of RF signals, distance calculations are somehow inaccurate and vary depending on factors such as signal attenuation, reflection among others.

The most basic concept to understand is signal propagation. First of all, signals attenuate as they travel (propagate) between the transmitting and receiving device. This attenuation, also known as path loss, increases with distance and is also affected by other factors such as obstacles present in the environment, temperature, weather, etc. Signals might also collide against obstacles and be diffracted or absorbed depending on the obstacle’s material.

Different models have been created in order to understand signal propagation. These models take into account diverse factors, including path loss, and can be used in order to calculate distances [13]. One of the most popular propagation models is known as the Free Space Friis Model (FSFM).

FSFM makes a comparison between the transmitted and received RSSI to know how much the signal attenuates over distance. As it names suggests, this model can be applied only to calculations made in free space, that is, in an environment without obstacles that may interfere with the signals.

FSFM also takes into account another factor known as antenna gain, a measurement of the antenna's ability to concentrate radio frequency energy in a particular direction [26], from both transmitting and receiving devices. Both RSSI and antenna gain values can be used to calculate distances using the following formula (1),

$$d = \sqrt{\frac{Pt}{Pr}} \cdot \left(\sqrt{Gt \cdot Gr}\right) \cdot \left(\frac{c}{4\pi f}\right) \qquad (1)$$

where P*t* and P*r* are the transmitted and received powers, G*t* and G*r* the antenna gain values of the transmitter and receiver, *c* is the speed of light and *f* the signal frequency.

A simplified graph representing the relation between distance and RSSI, according to FSFM [13], is shown in Figure 11. There we can see how the RSSI value decreases as the distance increases. This change is not always linear, however, and it is at distances over 7 meters where the RSSI values seem to stagnate.

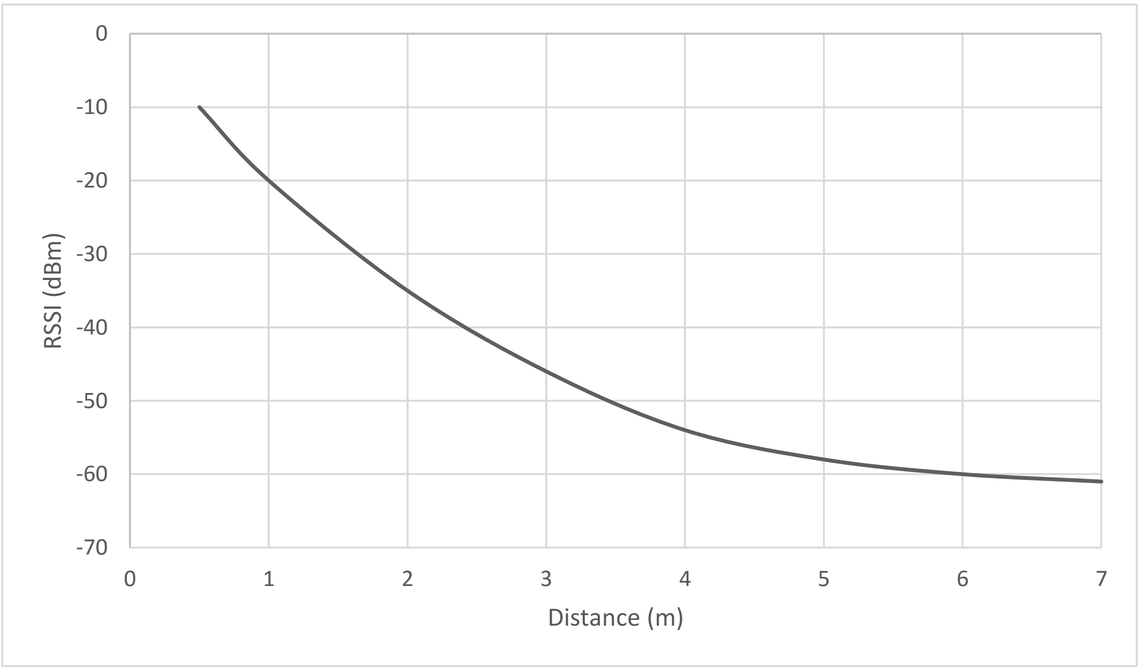


Figure 11. Relation between RSSI and distance according to FSFM

Even though this model seems simple enough for its use in this project, values such as antenna gain and exact signal frequencies cannot be easily found by using tools such as mobile phones and beacons. Furthermore, this model is designed to work only in open spaces. Due to these reasons, a more suitable method is needed.

An even simpler propagation model proposed in [27] seems like a formidable option for this thesis work. This same model has been used in other systems [7, 28, 29, 30, 31] and has proven to work relatively well for distance calculations based on RSSI. This model is represented by the following formula,

$$RSSI = P_{tx} + G + 20\log\left(\frac{c}{4\pi f}\right) - 10\log(d) \qquad (2)$$

where $P_{tx}$ is the beacon's transmit power in dBm, *G* is the combined antenna gain of both the transmitting and receiving device; *c* is the speed of light or 300,000 km/s; *f* is the frequency of 2.44 GHz; *d* is the distance between the transmitter and receiver and *n* is the attenuation exponent. This exponent ranges from 4 to 2 and is calculated using the following formula,

$$n = -\left(\frac{RSSI - A}{10log_{10}d}\right) \qquad (3)$$

where *A* is the beacon's RSS at 1m distance in a non-obstacle environment [27]. The attenuation exponent must be calculated for each reference device and will change depending on the environment. That is, the obstacles and interferences affecting the signal. In an environment where no obstacles are present (free space) the value of the attenuation exponent is 2 [27].

As mentioned before, values such as antenna gain and exact frequencies cannot be easily acquired since they are not transmitted by the beacon. It is because of this that (2) can be further simplified as shown in (4).

$$RSSI = A - 10n\log(d) \qquad (4)$$

Thanks to this model, distances between devices, using only the information broadcasted by the beacons and a mobile device, can be easily calculated.

# 7 IMPLEMENTATION

At this point, all necessary concepts related to indoor navigation have been explained. Information about the technologies used in this thesis project as well as a way to use them in positioning algorithms have been also covered. In this chapter all this information is put together and implemented into a navigation system consisting on a group of beacons and an Android application.

The system works as follows: first, few beacons are positioned around an area. Then, the application scans for them and gathers their RSSI readings. These readings are, in turn, used to calculate the distance between the user and each beacon. Since no information being broadcasted by the beacons indicates their position relative to the receiver, the application makes use of both the gyroscope and compass integrated in the mobile device to know in which direction the user is moving. Both the beacons' and the user's position, as well as the distance between them, are shown in a map. In order to provide the user with additional information related to their position, tools provided by Google as part of the Eddystone protocol will be used to add small data attachments to the beacons.

## 7.1 Testing environment

The system was tested inside a 50 m$^2$ apartment where two beacons, separated by a distance of approximately 5 m, were placed. The map of the apartment as well as the placement of the beacons is shown in Figure 12.

The reason for setting this kind of testing environment is to test the accuracy of the system based on its capability to calculate the distance from beacons that are positioned close to each other. This situation will give an idea of how beacons should be configured regarding their output power, which will affect the range they can achieve and the RSSI readings collected by the application. For example, if the area is no more than 100 m$^2$ there is no point using several beacons operating with an output power of 0 dBm (around 50 m range). This will only cause the beacons to consume unnecessary power and affect the accuracy of the distance calculations as RSSI values will barely change.

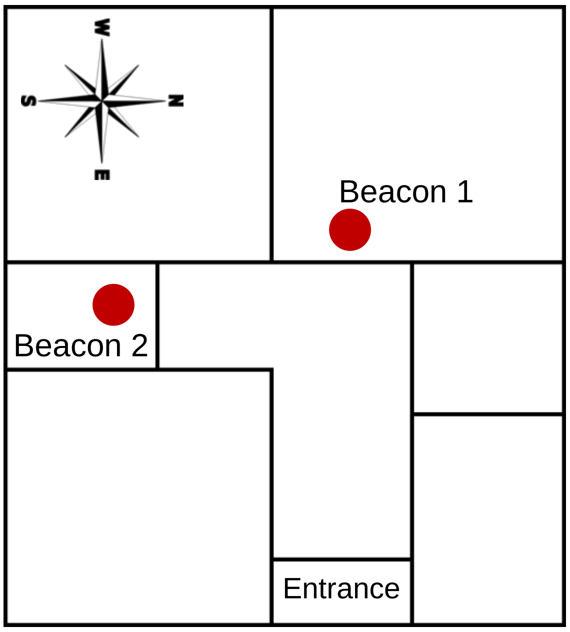


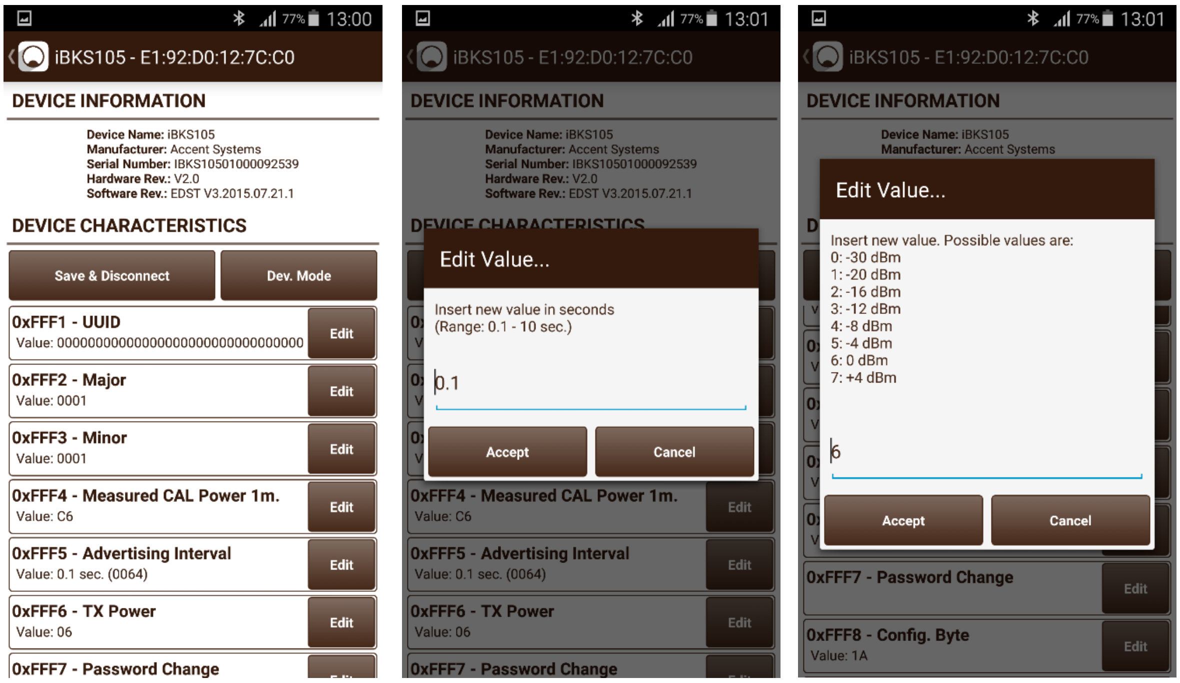
Figure 12. Test environment

## 7.2 Hardware

A Samsung Galaxy S5 smartphone using Android version 5.0 (Lollipop) was used to host the application. It is important to mention that BLE capabilities are only available on Android devices using version 4.3 (Jelly Bean) or newer. Few iBKS BLE beacons, manufactured by Accent Systems [32] were used to provide the reference signals. An example of these beacons is shown in Picture 5.

Picture 5. iBKS BLE beacon

In order to achieve optimal performance, beacons must be configured using the configuration application provided by the beacon manufacturer (Picture 6).


iBKS105 - E1:92:D0:12:7C:C0

DEVICE INFORMATION

Device Name: iBKS105
Manufacturer: Accent Systems
Serial Number: IBKS10501000092539
Hardware Rev.: V2.0
Software Rev.: EDST V3.2015.07.21.1

DEVICE CHARACTERISTICS

Save & Disconnect | Dev. Mode

0xFFF1 - UUID
Value: 00000000000000000000000000000000

0xFFF2 - Major
Value: 0001

0xFFF3 - Minor
Value: 0001

0xFFF4 - Measured CAL Power 1m.
Value: C6

0xFFF5 - Advertising Interval
Value: 0.1 sec. (0064)

0xFFF6 - TX Power
Value: 06

0xFFF7 - Password Change

0xFFF8 - Config. Byte
Value: 1A

Edit Value...
Insert new value in seconds
(Range: 0.1 - 10 sec.)
0.1
Accept | Cancel

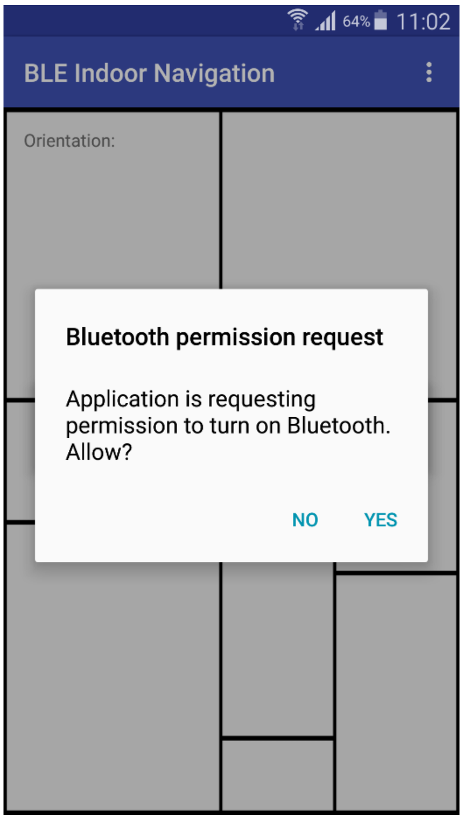
Edit Value...
Insert new value. Possible values are:
0: -30 dBm
1: -20 dBm
2: -16 dBm
3: -12 dBm
4: -8 dBm
5: -4 dBm
6: 0 dBm
7: +4 dBm
6
Accept | Cancel

Picture 6. iBKS Beacon Configuration Tool screenshots

For the purpose of this application an advertisement interval of 100 ms was chosen, ensuring the collection of enough RSSI readings (~10 readings) per second. As for the output power, it must be selected depending on the area that needs to be covered and the number of beacons used to do so. As explained before, this must be done according to the required range. Based on the testing environment, the output power value was set to -12 dBm which enables the beacons to cover an area of approximately 10 – 15 m, depending on the environmental interferences.

## 7.3 The application

At the start, the application screen looks as shown in Picture 7. A dialog is displayed on the screen asking permission from the user to turn on the Bluetooth adapter. Once this dialog is cleared, the application immediately starts scanning for beacons.



Bluetooth permission request

Application is requesting permission to turn on Bluetooth. Allow?

Picture 7. Application initial screen

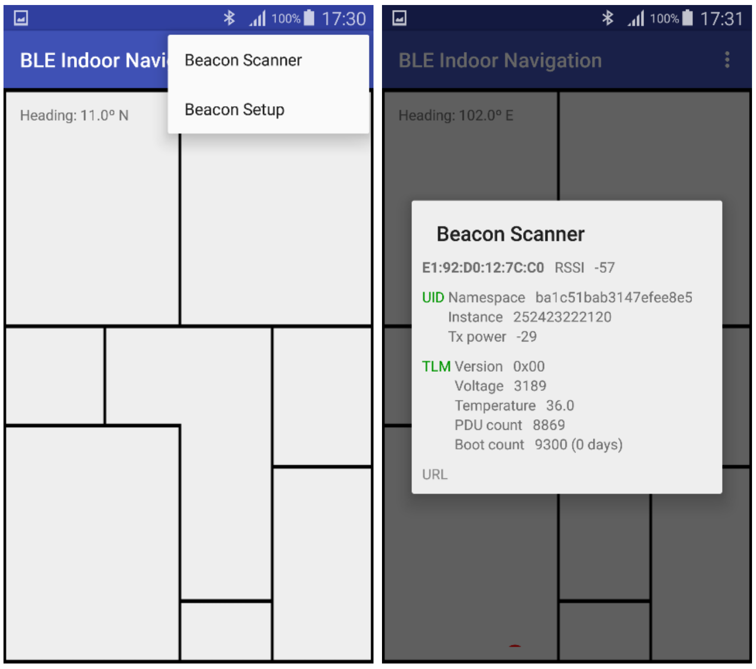
### 7.3.1 Scanning

The BLE adapter scans for nearby BLE devices and the results of every scan are passed through an Eddystone filter. This filter is used to detect if the beacon uses the Eddystone protocol and, if it does, it validates its frames and reads the information contained in them. These filters and other utilities necessary for reading Eddystone frames are provided by Google at their GitHub repository [33].

The application will search for all Eddystone frames that might be included in the BLE packet. Both Telemetry and URL frames are optional and can be left out, however, the UID frame must be present at all times. If no UID frame is found, the filter will ignore the device.

An additional screen where users are able to see all beacons in range and the contents of their frames is available from the menu bar at the top of the screen.

A screenshot of this screen is shown in Picture 8, where we can see an Eddystone beacon that contains both a UID and a TLM frame in its advertisement.



```
Beacon Scanner
E1:92:D0:12:7C:C0   RSSI  -57
UID Namespace  ba1c51bab3147efee8e5
    Instance  252423222120
    Tx power  -29
TLM Version  0x00
    Voltage  3189
    Temperature  36.0
    PDU count  8869
    Boot count  9300 (0 days)
URL
```

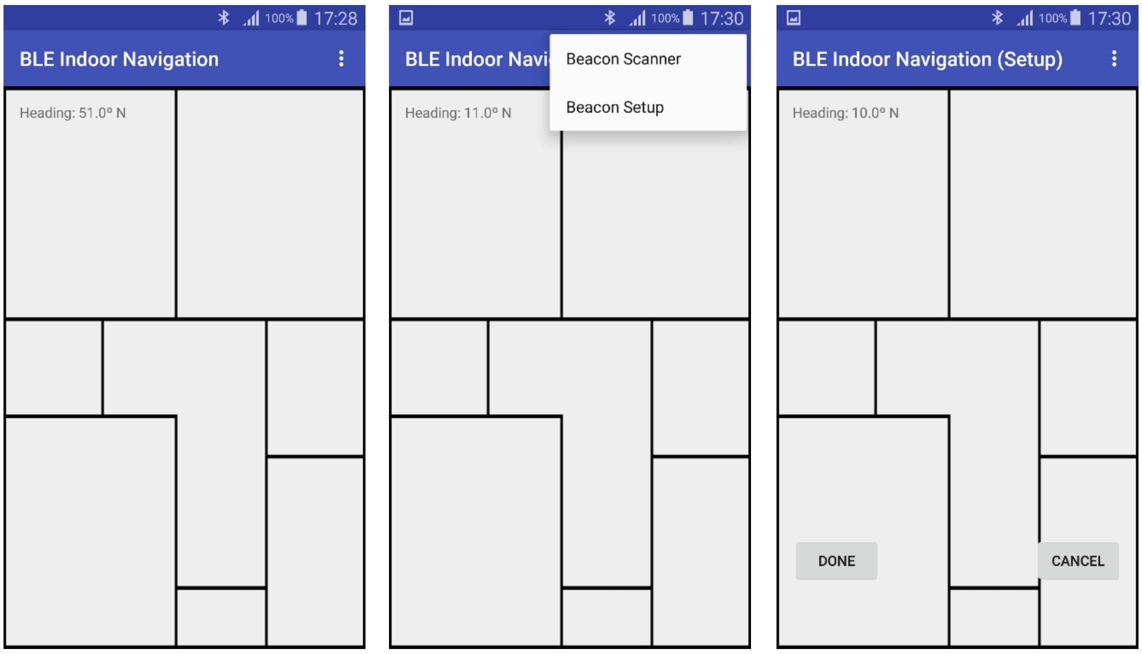
Picture 8. Beacon scanner screen

### 7.3.2 Setup

In order to use them as reference, the application must know where each beacon is within the map, therefore, their positions must be set by the user. Beacons within the application are generic objects, contained inside their own instances of a Java class, which can be added and configured independently by whoever is deploying the system.

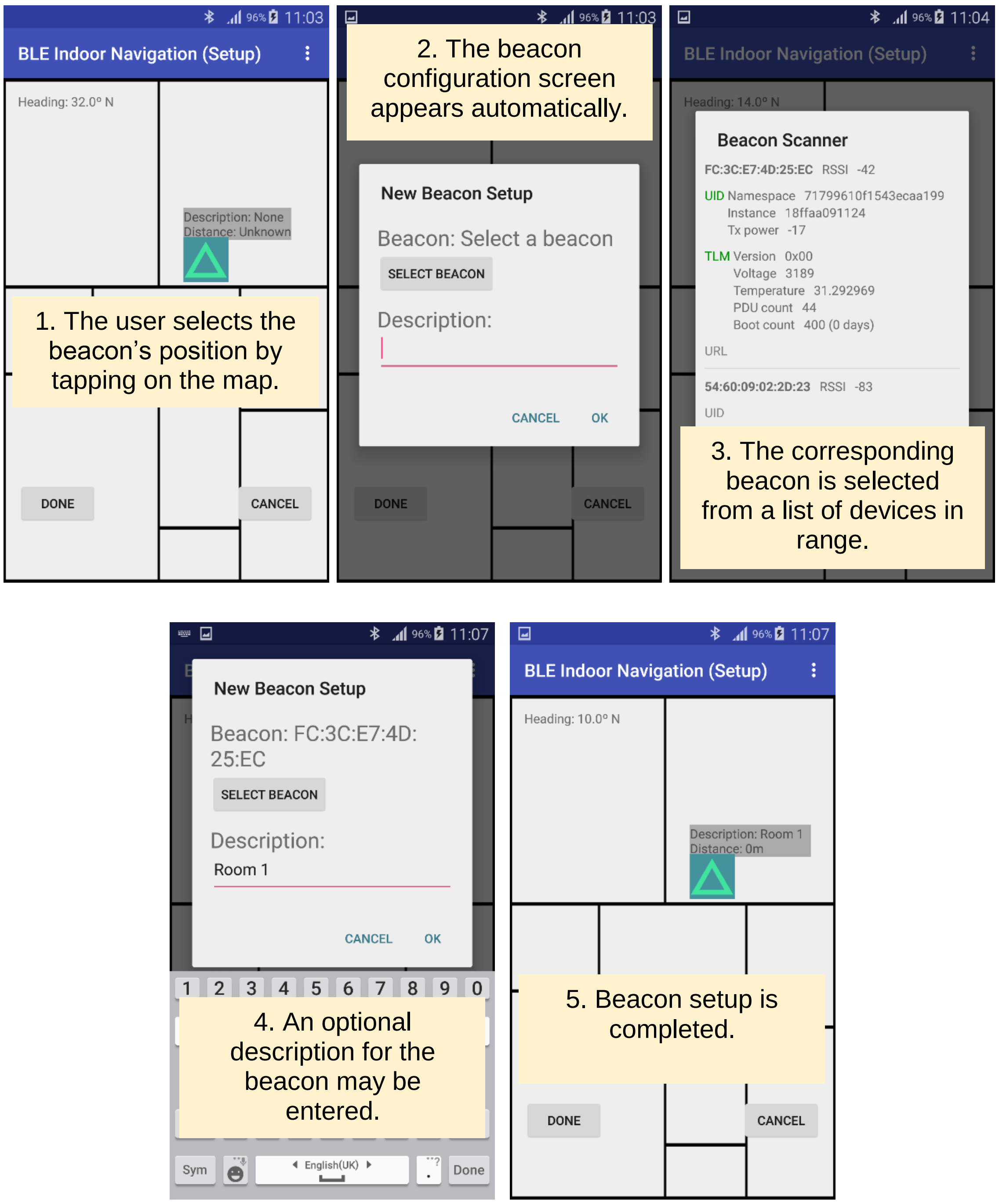
In order to configure the application, the beacons must be positioned at their desired locations throughout the building. Then, the position of every beacon should be added to the application by tapping on the screen and selecting their corresponding place in the map. To avoid possible mistakes, this could be done at the same time as beacons are being placed physically.

In order to better understand the set up process, screenshots of the application are shown next. In Picture 9 we can see how the user enters the setup mode from the menu bar.



Picture 9. Accessing the setup mode

Once in setup mode, the user may start adding the beacons to the map by tapping on the place they are physically placed. By doing this, a new beacon instance will be created and a configuration screen corresponding to it will appear. Once the configuration screen appears, the user will choose the physical beacon to be used by selecting it from a list of beacons populated with the scan results. By selecting the beacon, its address will appear at the configuration screen letting the user know it has been selected. The user may also choose to provide a description for the beacon, this could be the name of a room, its number, etc. All this process can be seen in Picture 10.



1. The user selects the beacon's position by tapping on the map.

2. The beacon configuration screen appears automatically.

Beacon: Select a beacon

Description:

3. The corresponding beacon is selected from a list of devices in range.

```
FC:3C:E7:4D:25:EC  RSSI  -42

UID Namespace  71799610f1543ecaa199
    Instance  18ffaa091124
    Tx power  -17

TLM Version  0x00
    Voltage  3189
    Temperature  31.292969
    PDU count  44
    Boot count  400 (0 days)

URL

54:60:09:02:2D:23  RSSI  -83

UID
```

4. An optional description for the beacon may be entered.

Beacon: FC:3C:E7:4D:25:EC

Description: Room 1

5. Beacon setup is completed.

Picture 10. New beacon setup process

Once the beacons have been configured, the user may press the “Done” button at the bottom of the screen to exit the setup mode. All configured beacons will be saved in order to avoid repeating the setup process every time.

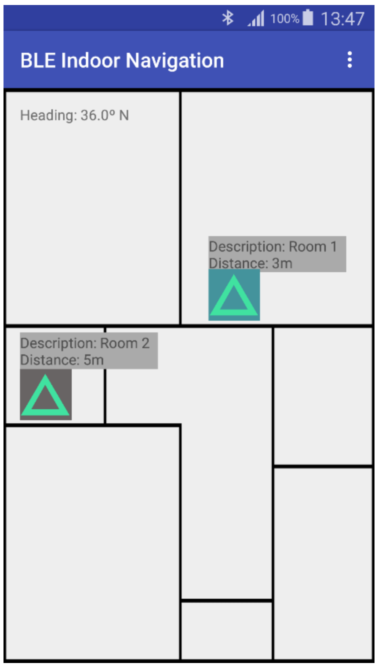
### 7.3.3 Navigation

Immediately after exiting set up mode, the application will begin calculating the distances to each of the configured beacons. These calculations will be later used to show the user's position within the map.

#### 7.3.3.1 Distance calculation

Distance calculations are performed at 1 sec intervals using formula (4), explained in the previous chapter. As mentioned before, the beacon advertising interval was set at 100 ms, which means, an average of ten RSSI readings is collected every second. For each one of these readings, a distance calculation is performed and the results used to obtain an average distance. The reason for doing this is to decrease the difference between calculations caused by RSSI fluctuation.

Every beacon displays its distance to the user independently, as shown in Picture 11. We can also see that the distances are rounded to the nearest meter.



Picture 11. Each beacon's distance to the user

For this project, the distance calculations where performed as follows. First, RSSI readings measured at different distances from one of the beacons were collected and an average value was calculated. This information is shown in Table 2.

Table 2. Average RSSI measurements at predefined distances

| **Distance** | **Average RSSI** |
|---|---|
| 1m | -71dBm |
| 2m | -75dBm |
| 3m | -77dBm |
| 4m | -81dBm |
| 5m | -87dBm |

In the next step, the attenuation exponent was calculated using formula (3), mentioned in the previous chapter. In an ideal situation this exponent would be calculated for every beacon but, in the case of this experiment, an approximate value calibrated around the navigation area proved to be enough to obtain fairly accurate results.

One necessary component for the calculation of both the attenuation exponent and the overall distance is the beacon's RSSI value calibrated at 1 m. This measurement is collected in an obstacle-free environment and, in the case of this experiment, equals to -68 dBm. Using the readings and distances presented in Table 2, along with this value, the attenuation exponent was calculated as shown in (5). Notice that this is calculated for every RSSI reading and its corresponding distance. All the results are then averaged in order to obtain a single value to be used for all distance calculations.

$$n = -\left(\frac{(-75)-(-68)}{10log_{10}(2)}\right) \tag{5}$$

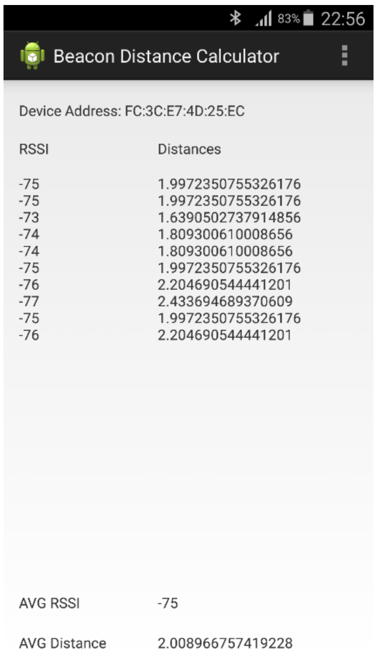
Now that all the necessary components for calculating distances have been collected, these can be used in formula (4) as shown in (6).

$$RSSI = (-68) - 10(2.2)\log(d) \tag{6}$$

Rearranging (6) for *d* we obtain (7) which is the formula used in the application.

$$d = 10^{\frac{(-68)-RSSI}{10*2.2}} \tag{7}$$

In order to track how these calculations are performed, a small additional application was created. All RSSI values collected and their corresponding distance results are shown on the screen, as seen in Picture 12.



Device Address: FC:3C:E7:4D:25:EC

| RSSI | Distances |
|---|---|
| -75 | 1.9972350755326176 |
| -75 | 1.9972350755326176 |
| -73 | 1.6390502737914856 |
| -74 | 1.809300610008656 |
| -74 | 1.809300610008656 |
| -75 | 1.9972350755326176 |
| -76 | 2.204690544441201 |
| -77 | 2.433694689370609 |
| -75 | 1.9972350755326176 |
| -76 | 2.204690544441201 |

| | |
|---|---|
| AVG RSSI | -75 |
| AVG Distance | 2.008966757419228 |

Picture 12. RSSI collection and distance calculation

To have an idea of how accurate distance calculations are, results were collected at predefined distances for periods of 30 seconds. This means that approximately 300 values were gathered and averaged. These values were then compared to the actual distances, shown in Table 3.

Table 3. Comparison between calculated and actual distances

| **Actual distance** | **Average calculated distance** |
|---|---|
| 1m | 1.8538m |
| 2m | 2.4632m |
| 3m | 4.4712m |
| 4m | 6.2581m |
| 5m | 7.4569m |

Results shown in Table 3 correspond to only one of the beacons. As mentioned before, it is preferable to calculate an attenuation factor for each device. Even though beacons sharing the same configuration may perform similarly, obstacles and interferences present in the environment should not be neglected. However, in the case of this project, the testing area was clear of major obstacles and was small enough for the same attenuation factor to be used for both beacons.

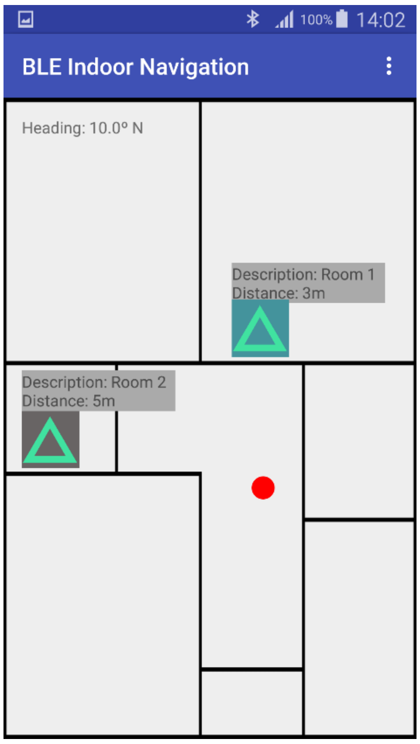
Based on the outcomes of this experiment, we can see that the accuracy of the system seems to be quite good at distances closer than 3 m. It is at longer distances that the calculations tend to differ quite significantly from the actual measurements, most likely due to signal attenuation.

Other factors that appear to affect the RSSI are the speed at which the user moves and the way the mobile device is handled, especially when there is not clear line of sight between this and the beacon. How the beacons are positioned seems to be an issue as well since RSSI readings seem to improve when the beacon is placed at the same height as the mobile device. This is not a common practice in this kind of systems, though. In the case of this project, beacons were positioned at the top a door, at about 2 meters from the ground.

#### 7.3.3.2 User’s movement

Now that the application knows how far the user is from each of the beacons, it can also estimate its position and display it in the map. This is done by displaying a red dot, referred to as “pointer”, representing the user and moving it through the screen as the user walks around the navigation area.

In order to calculate the user’s initial location, distance calculations to the closest beacons are selected and used to position the pointer as accurately as possible within the map. This might not work all the time, however, which is why the user can set the pointer at a known start position by tapping on the screen outside the setup mode. How the pointer is located relatively to the configured beacons is shown in Picture 13.



Picture 13. User’s position relative to the beacons

As the user moves around, distances will change and the pointer will move accordingly. It is here where both the compass and gyroscope sensors, integrated into the phone, come in handy. These are used to know in which direction, or more specifically, which cardinal point the user is heading for. In order for this information to be useful, the map must be oriented towards a certain cardinal point. As seen in Figure 12, the map used for this project has been orientated west, since that is the direction the user will be walking towards after entering the apartment. This means that the pointer will move forward if the user walks west, back if the user walks east, and so on.

The way the application decides how far the pointer should move is relative to how much the user walks around the navigation area. For example, if the user is standing at 1 m from one of the beacons and walks 2 m away from it, the application will calculate the difference between distances and move the pointer towards the corresponding direction. If two or more beacons are in range, the application will select the closest one, also taking into consideration the user's direction. Inside the screen, 1 m is represented by 100 dp (Density-independent Pixels). This value is multiplied by the number of meters covered by the user while walking, making the pointer move proportionally inside the screen. It is worth pointing out that the measurement of screen units was calculated for the device used in this project and may have to be readjusted when using other devices with different screen densities.

It is also important to have in mind that the pointer is only used to roughly show the user's position and might not be always 100% precise. Taking into account the accuracy of the distance calculations, the pointer may not behave as expected and might appear closer or further from the actual position of the beacon. This inaccuracy may be tolerated as long as the position shown is not too far from reality and the user can have an idea of where he/she is.

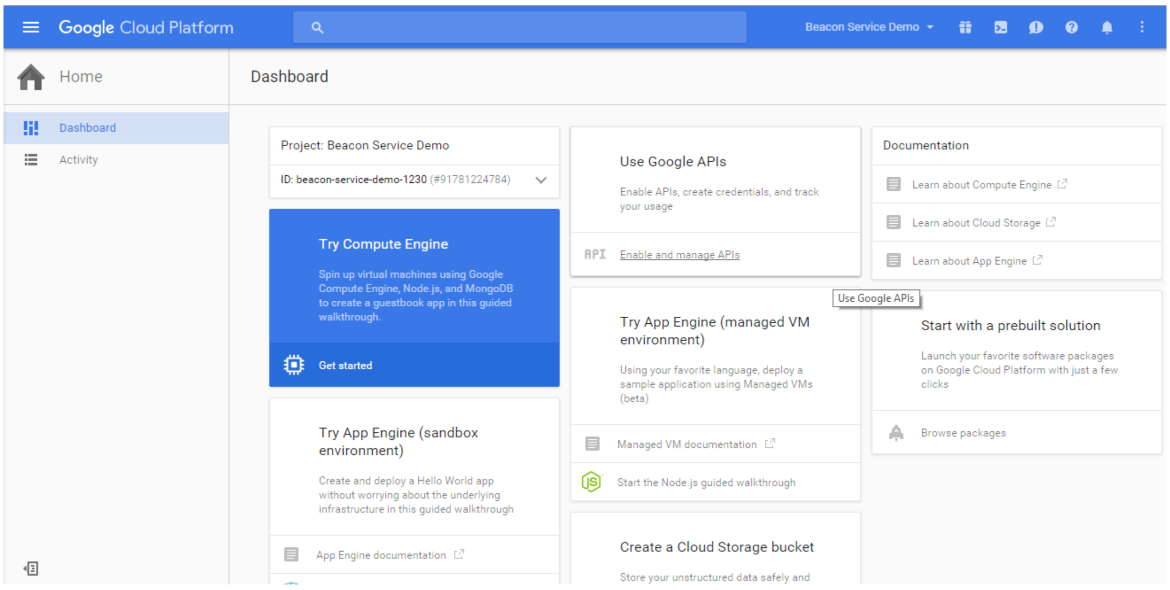
### 7.3.4 Obtaining additional information with beacons

The navigation system can be further improved by enabling it to provide additional information to the user. For this project, the same beacons used as reference are also used as sources of information.

As already mentioned in Chapter 4, beacons are limited to broadcasting only 31 bytes of data, which renders them useless for other purposes other than broadcasting their RSSI and identifying information. It is because of this that protocols, such as Eddystone, are used to improve the beacon's usability.

One of the features offered by Eddystone is the option to add attachments containing additional information to the beacon. These attachments are stored at Google's cloud and can be accessed at any time by the application using the Proximity Beacon API [22]. An API is a software tool that enables the application to perform certain tasks it would not be able to perform without it. In the case of the Proximity Beacon API, these tasks involve accessing Google's cloud, registering beacons as well as adding and retrieving data attachments. All this is done over the internet, which is why the user must be online in order to access all these features.

In order to access the Proximity Beacon API, one must first create a Google account and request an API key. This key authenticates the application and allows requests made by it to be accepted by Google's servers. API key requests are made through the Google Cloud Platform, an online platform that provides cloud-based services to Android developers. A screenshot of the platform is shown in Picture 14.



Project: Beacon Service Demo
ID: beacon-service-demo-1230 (#91781224784)

Try Compute Engine
Spin up virtual machines using Google Compute Engine, Node.js, and MongoDB to create a guestbook app in this guided walkthrough.

Try App Engine (sandbox environment)
Create and deploy a Hello World app without worrying about the underlying infrastructure in this guided walkthrough
App Engine documentation

Use Google APIs
Enable APIs, create credentials, and track your usage
API Enable and manage APIs

Try App Engine (managed VM environment)
Using your favorite language, deploy a sample application using Managed VMs (beta)
Managed VM documentation
Start the Node.js guided walkthrough

Create a Cloud Storage bucket
Store your unstructured data safely and

Documentation
Learn about Compute Engine
Learn about Cloud Storage
Learn about App Engine

Start with a prebuilt solution
Launch your favorite software packages on Google Cloud Platform with just a few clicks

Picture 14. Google Cloud Platform

API keys have to be associated with a certain project and user, which is why the developer must create a Google Cloud Platform project that identifies the application being developed.

Instructions on how to implement the Proximity Beacon API into an Android application, as well as code samples used for this project, are provided by Google at their Developers website [22].

#### 7.3.4.1 Registering beacons with Google Eddystone

Once the API has been enabled and the necessary permissions granted to the application, data attachments can be added to the beacons.

In order to add data attachments to the beacons, these must be first registered with Google [34] using their own ID. Beacon registration requests, as well as attachments added to the beacon later on, are sent to the Google servers using HTML libraries compatible with Android.

The beacon ID is broadcasted by the beacon hardware and is composed of the beacon type (Eddystone, iBeacon, etc.) and a unique identifier. This last one corresponds to the beacons UID made from the Namespace ID and Instance ID fields contained in the UID Eddystone frame (see Chapter 4).

Each beacon has its own UID which can be configured by the user using the guidelines provided by the Eddystone documentation [19]. This is normally done because most brand new beacons have a preconfigured UID already registered by the manufacturer. Once a beacon using a certain UID is registered, that value cannot be used to register a different beacon. Figure 13 shows an example of how the beacon UID is constructed.

| Namespace ID (10 Bytes) | Instance ID (6 Bytes) |
|---|---|
| 71799610F1543ECAA199 | 18FFAA091124 |
| Eddystone UID : 71799610F1543ECAA19918FFAA091124 | |

Figure 13. Eddystone UID configuration example

As previously seen in Figure 13, UIDs are numerical values formatted in hexadecimal notation. Developers may create their own UID choosing hex values, from 0 to F, and use them following the same format shown previously.

Another information required to register a beacon is its status. The beacon status can be either ACTIVE, which means the beacon is operating normally; INACTIVE, meaning the beacon is out of service and invisible to scanners; and DECOMMISSIONED, meaning the beacon has been deactivated and cannot be used for any purpose. The status of the beacon can be changed once it is registered. If the beacon must be temporary out of service the INACTIVE status must be chosen. The DECOMMISSIONED value must only be used to permanently disable the beacon registered to a specific UID and is irreversible.

When the UID and the status of the beacon have been set, this information is sent to Google servers for the device to be registered. The data is contained into a registration request formatted as shown in Figure 14 and sent to the following URL: *https://proximitybeacon.googleapis.com/v1beta1/beacons:register*

```
{
  "advertisedId": {
    "type": "EDDYSTONE",
    "id": "Fr4Z98nSoW0hgAAAAAAAAg=="
  },
  "status": "ACTIVE"
}
```

Figure 14. Beacon registration request

In the previous figure (Figure 14) we can see how the beacon ID, originally in hexadecimal format, has been encoded in Base64 before being sent. This is also done for the contents of data attachments, discussed later on this chapter.

The format in which the registration request is formatted is called JSON (JavaScript Object Notation). JSON is a way of arranging data in a logical way, easy for humans to understand and machines to process. Other request and response messages, discussed next, are also formatted in this way.

Optionally, additional information can be added to the beacon when registering it. This can be the latitude and longitude of its position, indoor floor level, as well as information used by additional APIs such as Google Places [35]. Such additional fields may be added to the original HTTP request as shown in Figure 15.

```
{
  "advertisedId": {
    "type": "EDDYSTONE",
    "id": "Fr4Z98nSoW0hgAAAAAAAAg=="
  },
  "status": "ACTIVE",
  "placeId": "ChIJTxax6NoSkFQRWPvFXI1LypQ",
  "latLng": {
    "latitude": "47.6693771",
    "longitude": "-122.1966037"
  },
  "indoorLevel": {
    "name": "1"
  },
  "expectedStability": "STABLE",
  "description": "An example beacon."
}
```

Figure 15. Additional beacon register information

Once the registration information is sent to the server, this will return a code according to the outcome of the request. The possible codes are: ***200 (OK)***, which means the beacons has been successfully registered; ***403 (Forbidden)***, meaning the beacon has already been registered to another application and its information is not available; and ***404 (Not Found)*** which means the beacon has not been registered either because the request was unsuccessful or any other problem with the server.

In addition to the response code, a message containing the registered beacon information will be returned. Contained in this message, among others, is a property called ***beaconName***, which is used to identify the beacon when adding attachments to it. This can be seen in Figure 16.

```
{
"beaconName": "beacons/3!71799610F1543ECAA19918FFAA091124",
 "advertisedId": {
  "type": "EDDYSTONE",
  "id": "Fr4Z98nSoW0hgAAAAAAAAg=="
  },
  "status": "ACTIVE"
}
```

Figure 16. Beacon registration response

### 7.3.4.2 Adding and retrieving beacon data attachments

Once the beacons have been successfully registered, attachments can be added to them [36]. These attachments can be 1KB long and may contain data in the form of a text string, a reference to an external resource or structured data such as JSON.

Data attachments are sent to the server using the URL: *https://proximitybeacon.googleapis.com/v1beta1/beacons/beaconName/attachments* and are formatted as shown in Figure 17. Notice the *beaconName* part of the URL, which is where the ***beaconName*** property, obtained from the register response, is placed.

```
{
  "namespacedType":"project-name/attachment-type",
  "data":" RXhhbXBsZSBhdHRhY2htZW50"
}
```

Figure 17. Beacon data attachment

The *data* part of the attachment shown in Figure 17 corresponds to the string “Example attachment” encoded in Base64. The ***namespacedType*** part corresponds to the Google Cloud Platform project, created in order to request the API key, and the type of the attachment. This last one can be anything that describes the data contained in the attachment.

Once the attachments have been sent to Google’s servers, these will be given a name that represents a path that can be used to retrieve them. Then, a server response (200 OK) and the successfully created attachment, formatted as shown in Figure 18, will be returned to the application.

```
 {
"attachmentName":
"beacons/3!442560da34bd204a9ed7000782503039/attachments/5a58d6d1-d9b5-
4de6-a01d-dc73dff18779",
"namespacedType":"project-name/my-attachment-type",
"data":"aGVsbG8gd29ybGQh"
}
```

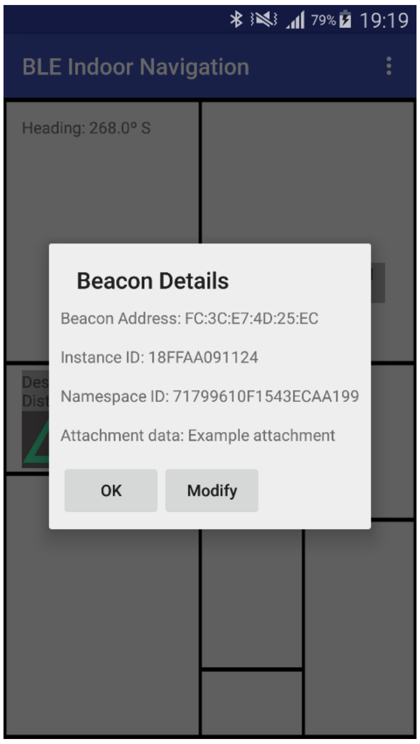
Figure 18. Server response containing successfully created attachment

All beacon attachments may be retrieved [37] sending a request to the following URL *https://proximitybeacon.googleapis.com/v1beta1/{**beaconName=beacons/***}/attachments* to which the server will respond with a message formatted as shown in Figure 19. The ***beaconName*** part of the URL is required and used to retrieve all the attachments related to that beacon. An optional field corresponding to the ***namespacedType*** may also be used to retrieve specific attachments.

```
 {
"attachments": [
{
object(BeaconAttachment)
}
],
}
```

Figure 19. Server response containing a requested beacon attachment

Data contained in the attachments can be used and displayed by the application however the developer wishes to do so. In the case of the application developed for this thesis, the user may click on the beacon outside setup mode and have access to the attached information. This is shown in Picture 15.



Beacon Details

Beacon Address: FC:3C:E7:4D:25:EC

Instance ID: 18FFAA091124

Namespace ID: 71799610F1543ECAA199

Attachment data: Example attachment

Picture 15. Beacon details including attachment data.

It is evident how adding attachments to beacons may be a long task. It may also seem illogical to go to such trouble for only a small amount of additional data. Nevertheless, having access to tools such as the Proximity Beacon API, among other APIs provided by Google, is an advantage. These tools contain features that would be hard and time-consuming to develop otherwise. Furthermore, attachments are essential to separate the hardware advertisements from the information related to the role the beacon plays inside the application. The data attached may seem small, but it can be used to access external resources such as databases. Other valuable features provided by Eddystone include beacon management and monitoring, as well as attachment retrieval using other APIs such as Nearby Messages API [38].

# 8 CONCLUSION

As opposed to the well-established concept of outdoor navigation, indoor navigation must still overcome many challenges before becoming an everyday use commodity. One example is the limited amount of technologies that can be used for this purpose, as well as the accuracy offered by these. Furthermore, how to make indoor navigation systems available to the public is still a great challenge.

The main purpose of this thesis was to introduce indoor navigation and show how, by using common devices, it is possible to create an indoor navigation system. Taking advantage of the both the beacons and the sensors integrated into a smartphone, a simple but functional application was developed. Regardless of its simplicity, this prototype comprises all the indoor navigation concepts discussed throughout the document.

The developed application will continue to be improved and will eventually be published in the Android marketplace. Future work will focus on finding new ways to improve the application functionality regarding both the positioning algorithm and the hardware configuration. Making the system adaptable to different environments will be a top priority since the results of the distance calculations previously shown can only relate to a very specific testing environment.

The information presented in this document discusses only the most essential concepts that need to be understood when working on a project such as this. Most of the research carried out was related to RF signals and the possible ways to implement them into this kind of system. One of the greatest challenges faced during the writing of this thesis was understanding and presenting this information as clearly as possible.

This work represents only a small part of what indoor navigation really is. As technology advances, more sophisticated and interesting solutions related to this topic will continue emerging. At the same time, more people will have access to these and they will, eventually, become part of the everyday life.

# 9 REFERENCES

[1] Ojeda, L. & Borestein, J. 2007. 'Personal Dead-reckoning System for GPS-denied Environments'. IEEE International Workshop on Safety, Security, and Rescue Robotics, pp. 1 – 6. Available from: IEEE Xplore Digital Library.

[2] Storms, W., Shockley, J. & Raquet, J. 2010. 'Magnetic Field Navigation in an Indoor Environment'. Ubiquitous Positioning Indoor Navigation and Location Based Service, pp. 1 – 10. Available from: IEEE Xplore Digital Library.

[3] Grigorik, I. 2013. 'High Performance Browser Networking: What every web developer should know about networking and web performance', Chapter 6. California: O'Reilly Media.

[4] Downey, C. 2013. Electronic Design, 03 April 2013. 'Understanding Wireless Range Calculations'. Available from: <http://electronicdesign.com/communications/understanding-wireless-range-calculations>. [20 December 2015]

[5] FCC Rules for Unlicensed Wireless Equipment operating in the ISM bands. Available from: <http://www.afar.net/tutorials/fcc-rules>. [20 December 2015]

[6] Bluetooth radio interface, modulation, & channels. Available from: <http://www.radio-electronics.com/info/wireless/bluetooth/radio-interface-modulation.php>. [29 January 2016]

[7] Wang, Y., Yang, X., Zhao, Y., Liu, Y. & Cuthbert, L. 2013. 'Bluetooth Positioning using RSSI and Triangulation Methods'. Consumer Communications and Networking Conference, pp. 837 – 842. Available from: IEEE Xplore Digital Library.

[8] Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M. 2014. 'Internet of Things for Smart Cities'. IEEE Internet of Things Journal, vol. 1, no. 1, pp. 22 – 32. Available from: IEEE Xplore Digital Library.

[9] Lindh, J. 2015. 'Bluetooth® Low Energy Beacons'. Application Report, Texas Instruments. Available at: <http://www.ti.com/lit/an/swra475/swra475.pdf>. [30 November 2015]

[10] Galeev, M. 2011. Z-Focus Consulting, 14 July 2011. 'Bluetooth 4.0: An introduction to Bluetooth Low Energy—Part I'. Available from: <http://www.eetimes.com/document.asp?doc_id=1278927>. [02 February 2016]

[11] Nilsson, R. & Saltzstein, B. Connect Blue, Bluetooth Low Energy Technology and Healthcare. Available from: <http://www.connectblue.com/press/articles/bluetooth-low-energy-technology-and-healthcare/>. [02 February 2016]

[12] Faragher, R. & Harle, R. 2015. 'Location Fingerprinting with Bluetooth Low Energy Beacons'. IEEE Journal on Selected Areas in Communications, vol. 33, no. 11, pp. 2418 – 2428. Available from: IEEE Xplore Digital Library.

[13] Chowdhury, T. I., Rahman, M. M., Parvez, S., Alam, A. K. M. M., Basher, A., Alam, A. & Rizwan, S. 2015. ‘A multi-step approach for RSSi-based distance estimation using smartphones’. 2015 International Conference on Networking Systems and Security, pp. 1 – 5. Available from: IEEE Xplore Digital Library.

[14] Bluetooth Core Specification. Available from: <https://www.bluetooth.com/specifications/bluetooth-core-specification>. [31 January 2016]

[15] Getting Started with iBeacon. Available from: <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>. [05 February 2016]

[16] Google Beacons – AnDevCon, 2015 (video file), Available from: <https://www.youtube.com/watch?v=TZf4WquRGJU>. [29 October 2015]

[17] MFi Program. Available from: <https://developer.apple.com/programs/mfi/>. [05 February 2016]

[18] Eddystone Protocol Specification. Available from: <https://github.com/google/eddystone/blob/master/protocol-specification.md>. [10 January 2016]

[19] Eddystone-UID. Available from: <https://github.com/google/eddystone/tree/master/eddystone-uid>. [10 January 2016]

[20] Eddystone-URL. Available from: <https://github.com/google/eddystone/tree/master/eddystone-url>. [10 January 2016]

[21] Eddystone-TLM. Available from: <https://github.com/google/eddystone/tree/master/eddystone-tlm>. [10 January 2016]

[22] Proximity Beacon API. Available from: <https://developers.google.com/beacons/proximity/guides>. [10 January 2016]

[23] Sun, G.; Chen, J.; Guo, W.; Liu, K.J.R. 2005. ‘Signal processing techniques in network-aided positioning: a survey of state-of-the-art positioning designs’. IEEE Signal Processing Magazine, vol. 22, no. 4, pp.12 – 23. Available from: IEEE Xplore Digital Library.

[24] Sauter, M. 2010. ‘From GSM to LTE: An Introduction to Mobile Networks and Mobile Broadband’, 1.8.1 Call Reselection and Location Area Update, pp. 45. Sussex: John Wiley & Sons.

[25] Farshad, A., Li, J. & Marina, M.K. 2013. ‘A Microscopic Look at WiFi Fingerprinting for Indoor Mobile Phone Localization in Diverse Environments’. 2013 International Conference on Indoor Positioning and Indoor Navigation. pp. 1 – 10.

[26] Antenna Gain - Vol. 9 No. 33, 2009. Available from: <http://www.l-com.com/content/Article.aspx?Type=N&ID=9475>. [15 February 2016]

[27]Gu, Y. & Ren, F. 2015. 'Energy-Efficient Indoor Localization of Smart Hand-Held Devices Using Bluetooth'. IEEE Access, pp. 1450 – 1461. Available from: IEEE Xplore Digital Library.

[28] Lau, E. & Chung, W. 2007. 'Enhanced RSSI-Based Real-Time User Location Tracking System for Indoor and Outdoor Environments'. International Conference on Convergence Information Technology, 2007, pp. 1213 – 1218. Available from: IEEE Xplore Digital Library.

[29] Liu, S.; Jiang, Y.; Striegel, A. 2014. 'Face-to-Face Proximity Estimation Using Bluetooth On Smartphones'. IEEE Transactions on Mobile Computing, vol. 13, no. 4, pp. 811 – 823. Available from: IEEE Xplore Digital Library.

[30] Anagnostopoulos, G. & Deriaz, D. 2014. 'Accuracy Enhancements in Indoor Localization with the Weighted Average Technique'. SENSOR-COMM 2014, pp. 112 – 116.

[31] Zhou, S. & Pollard, J.K. 2006. 'Position Measurement using Bluetooth'. IEEE Transactions on Consumer Electronics, vol. 52, no. 2, pp. 555 – 558. Available from: IEEE Xplore Digital Library.

[32] iBKS105 – Datasheet. Available from: <http://accent-systems.com/wp-content/uploads/2015/11/iBKS-105-Datasheet.pdf>. [20 December 2015]

[33] Eddystone Validator. Available from: <https://github.com/google/eddystone/tree/master/tools/eddystone-validator>. [27 November 2015]

[34] Register Beacons. Available from: <https://developers.google.com/beacons/proximity/register>. [15 February 2016]

[35] Google Places API. Available from: <https://developers.google.com/places/>. [20 February 2016]

[36] Add Attachments to Beacons. Available from: <https://developers.google.com/beacons/proximity/attachments>. [15 February 2016]

[37] Retrieve Attachments. Available from: <https://developers.google.com/beacons/proximity/proximity-retrieve>. [15 February 2016]

[38] Nearby Messages API. Available from: <https://developers.google.com/nearby/messages/overview>. [20 February 2016]