University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Computer Science	Spring semester, 2016 Open / ~~Confidential~~
Author: Stian Sandve	..... S. Sandve ..... (signature author)
Instructor: Terje Kårstad Supervisor(s): Terje Kårstad	
Title of Master's Thesis: Indoor positioning system using BLE beacons Norwegian title: Innendørs posisjoneringssystem ved bruk av BLE beacons	
ECTS: 30	
Subject headings: Indoor positioning, wireless communication, Bluetooth Low Energy, web development	Pages: 75 + attachments/other: 1 Stavanger, 15th of June / 2016 Date/year

Indoor Positioning System Using BLE Beacons

by

Stian Sandve

Supervisor: Terje Kårstad

A thesis submitted in partial fulfillment for the
master's degree of Computer Science

at the

Faculty of Science and Technology

Department of Electrical Engineering and Computer Science

June 2016

This page is intentionally left blank.

Everything should be made as simple as possible, but not simpler.

ALBERT EINSTEIN

Abstract

This thesis provides a viable solution to the challenge of real-time estimation of people's positions inside a building during emergency situations. Such estimates can be extremely valuable to emergency responders by potentially reducing evacuation times and increasing the chance of saving lives. To provide rescuers with this information, an indoor positioning system using BLE beacons was implemented. Real-time updates is made accessible through a modern web application. By introducing a method to continuously calibrate the path loss exponent, an acceptable accuracy was obtained, where more than 70% of the position estimates had less than 2 meters of error. A manual search operation would put rescuers and victims at unnecessary high risks. By using the proposed system, rescuers can save time, which is critical in any emergency response operation.

Preface

This thesis was written at the Department of Electrical Engineering and Computer Science at the University of Stavanger. I would like to thank my supervisor Terje Kårstad for his valuable advice and support. I would also like to thank my fellow students.

This page is intentionally left blank.

Contents

List of Figures

This page is intentionally left blank.

Code Listings

This page is intentionally left blank.

Acronyms

AFH Adaptive Frequency Hopping

API Application Programming Interface

ARQ Automatic Repeat Request

BLE Bluetooth Low Energy

BS Base Station

CRC Cyclic Redundancy Check

dBm Decibel-milliwatt[1] (sometimes written dB_{mW})

DOM Document Object Model

FEC Forward Error Correction

FH-CDMA Frequency Hopping Code Division Multiple Access

GPS Global Positioning System

ID Identifier

IoT Internet of Things

IP Internet Protocol

ISM Industrial, Scientific and Medical

JSON JavaScript Object Notation

MPA Multi Page Application

MS Mobile Station

MVC Model View Controller

MVVM Model-View-ViewModel

PLE Path Loss Exponent

QoS Quality of Service

RSSI Received Signal Strength Indication

[1]The power ratio in decibels (dB) of the measured power referenced to one milliwatt (mW).

SNR Signal to Noise Ratio

SPA Single Page Application

TLM Telemetry

TX Transmission

UID Unique Identifier

UPS Uninterruptible Power Supply

URL Uniform Resource Locator

UUID Universally Unique Identifier

1 Introduction

1.1 Problem

The problem that is explored in this project is related to safety. If for some reason a building has to be evacuated, it is often challenging to keep track of the state of the evacuation. How many people were inside the building when the alarm went off? How many people have successfully been evacuated? And most importantly, are there anyone left inside the building? If there are, it would be beneficial if we could tell exactly were they are located to reduce the risk of both rescuers and victims. The purpose of this system is not to serve as an indoor navigation system, but rather a system that lets rescuers pin-point were people are located.

Localization of positions and detection of objects is a research area that has existed for a while, but it is still an ongoing research. The well developed Global Positioning System (GPS) works great in outdoor environments, whereas indoor positioning is still a challenging issue [34]. The reason for that is the demanding, dynamic indoor environment, causing severe multipath fading, leading to unpredictable propagation conditions. Much effort has been put into designing accurate indoor positioning systems, including technologies like acoustics, optics, WiFi, Bluetooth, ZigBee, GSM and RFID, just to mention a few. There is still no outstanding technology for indoor positioning systems as the various technologies has advantages and disadvantages regarding accuracy, availability, complexity and costs.

New technology are constantly entering the market, and one of the rapidly growing technologies that is predicted to make a big impact the coming years, is something called beacons. Beacons are low-cost devices that broadcasts a signal that can be registered by other smart devices nearby. They make use of *Bluetooth Low Energy* and can be almost the size of a coin. The aim of beacons are mainly to provide contextual awareness. This can in turn enable businesses to deliver contextually relevant content and information to users at very specific locations. The use cases are only limited by one's imagination. Beacons let smart devices such as smart phones estimate the distance to nearby beacons based on the RSSI (Received Signal Strength Indication). In this project, we will examine if it is possible to use these distance estimates to locate people inside buildings and make the information available to safety personnel in case of an emergency.

1.2 Objective

In case of an event that requires a building to be evacuated, it would be beneficial to get real time information about the peoples location. This information can aid rescuers in getting the remaining people out of the building as quickly as possible and increase the chance of saving lives.

1.3 Approach

The system consists of base stations, mobile stations, a processing server and a web server. The base stations are implemented with custom software installed on WiFi and Bluetooth Low Energy (BLE) enabled Raspberry Pi's. BLE beacons from Estimote are used as mobile stations. The base stations listen for mobile stations and the processing server collects information from the base stations. The web server hosts a custom web client developed in AngularJS that fetches real-time updates from the processing server and presents relevant information to first responders or system administrators. The processing server and web server may run on the same physical machine.

To estimate the position of a mobile station, RSSI based distance estimates are induced into a trilateration algorithm on the processing server. To increase the accuracy of the position estimates, a technique to continuously calibrate the path loss exponent was implemented.

1.4 Thesis Outline

Chapter 2 - Background:
Literature review of key concepts that the rest of the thesis is based on. Examples of related work are listed in the end.

Chapter 3 - Architecture:
Description of architecture and implementation details of the system that was developed.

Chapter 4 - Methodology, Results and Analysis:
Methodology used in the experiments and the results that was obtained. An analysis that discuss the reason and significance of the results is also presented.

Chapter 5 - Conclusions:
Summary of conclusions and evaluation of significance. Ideas for future work are also covered.

Appendix A - Source Code:
The software developed during this project is embedded in the PDF as a 7-Zip-file named `source.7z`.

2 Background

2.1 Internet of Things

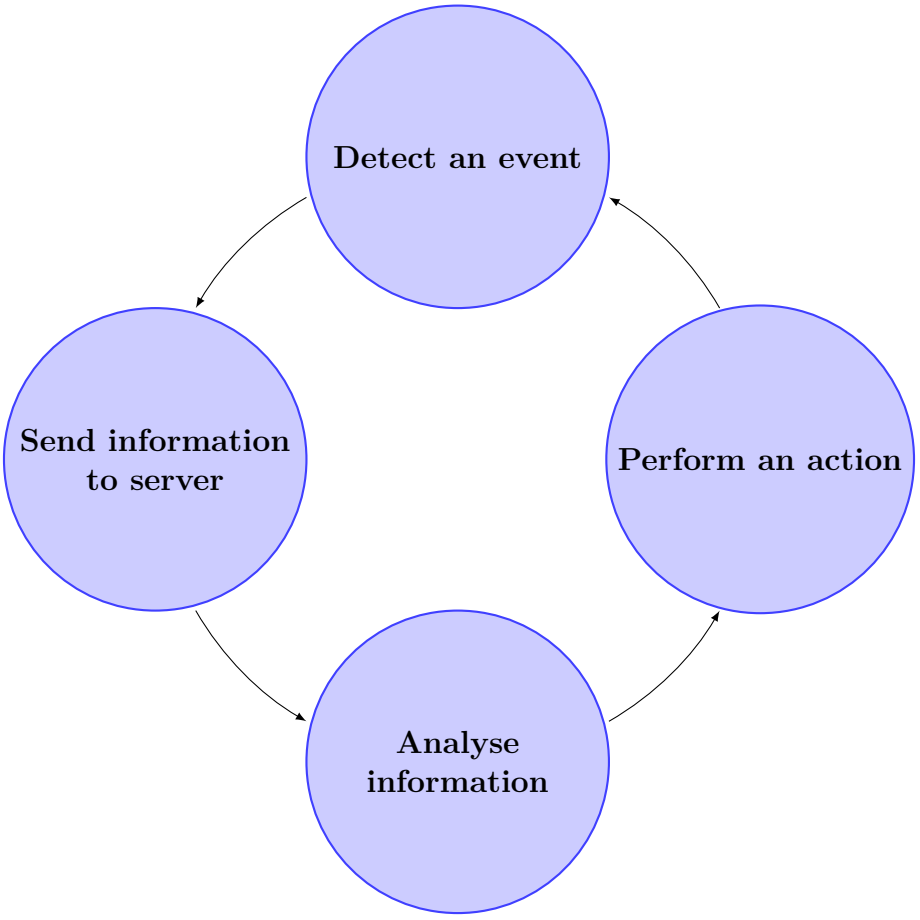


Figure 2.1: Traditional IoT architecture.

This project belongs to the category *internet of things*, or IoT. What does this imply? Inter means "between", so we could say that IoT is a network between things. Moreover, it is a network of uniquely identifiable and interoperable objects connected by services over the internet. It is a technology that connects the physical world with the digital world. An overview of a traditional IoT architecture is outlined in Figure 2.1. A physical device, e.g. a sensor, detects an event and notifies a server

about it. The server then analyse the information and acts accordingly. Here is a real world example: A dairy farmer may have hundreds of cows. In order to breed the cows, the farmer has to observe their behaviour and find out when a cow is going through *estrus*[1] so that he can inseminate her. Even for a small number of cows, this is a difficult task because the signs are often weak and the cows does not necessarily go into estrus at the same time. In 2013, Fujitsu introduced a system [8] where a sensor is placed on each cow and information about its movement is transmitted to a server that analyses the data. Certain fluctuations in movement can indicate that the cow is going into estrus and thus is ready to be inseminated. The farmer will get a notification on his smart phone so that he knows when the time is right. The system helped farmers raise their successful insemination rate.

Why do we need IoT? What can be achieved? IoT can contribute to solve climate issues, famine, public health and shortage of resources. It has received enormous attention and is predicted to be the next revolution or era in the series of major economic cycles. By 2020, we assume that there will be 25-100 billion devices connected to the internet. A report from McKinsey & Company's Global Institute [31] estimates that IoT can be worth between 3.9 and 11.1 trillion dollars by 2025. I.e, it can potentially be worth ~11% of the world's economy.

2.2 The Wireless Medium

This section is based on information found in [17].

Communication across a network is carried over a medium. The medium provides the channel over which the message travels from source to destination. Modern networks primarily use two categories of medium:

- Guided medium (copper cable, optical fibre link etc.)
- Wireless medium (open air, sea etc.)

It is obviously harder to communicate using a wireless medium compared to a closed guided medium. Wireless mediums are open to everyone, so the chance for interference affecting the signal quality is high. Imagine a choir of 50 people where each singer sings a different song and you are trying to listen to one of them. Obstacles in the physical environment such as walls, rain and fog also makes wireless communication a lot harder than its alternative. When developing a wireless sensor network, it is important to be aware of the challenges presented in this section.

2.2.1 Capacity

Capacity is the rate at which information can be transmitted over a communications channel and is a factor of the noise environment, frequency bandwidth and

[1] A period of 16 hours every 21 days where the cow is fertile. This period is colloquially known as *heat*.

modulation scheme. If we make the assumption that the channel is noise-free, then the capacity of the channel is limited to the bandwidth B, which is the range of frequencies present in the signal. The *capacity* of a noise-free channel can be stated as

$$C = 2B \log_2 M \tag{2.1}$$

where C is the capacity in bits per second, B is the frequency bandwidth in Hertz and M is the number of values that each signal can carry. Equation 2.1 is called *Nyquist formulation* [17] and gives the upper capacity limit for the available bandwidth. The equation is idealized because it does not consider environmental influences, such as noise. When the channel is not noise free, noise introduces additional distortion in the signals and therefore reduces the capacity. Shannon and Hartley developed a similar equation for capacity:

$$C = B \log_2 \left(1 + \frac{S}{N}\right) \tag{2.2}$$

where S is the received signal strength and N is the level of noise (elaborated in Section 2.2.3). $\frac{S}{N}$ is known as the the Signal to Noise Ratio (SNR). Equation 2.2 is known as the Shannon-Hartley theorem.

2.2.2 Path Loss

The difference in signal strength at the transmitter P_t and receiver P_r is affected by distance, antenna gains, losses in the electrical circuits and other random factors [17]. This difference can be modeled as

$$\frac{P_t}{P_r} = \left(\frac{4\pi d}{\lambda}\right)^{\gamma} = \left(\frac{4\pi f d}{c}\right)^{\gamma} \tag{2.3}$$

where λ is the wavelength (which is the relationship between the frequency f and the speed of light c), d is the distance between the transmitter and receiver and γ is the Path Loss Exponent (PLE). The PLE is usually ~2 for free space and ~4 in dense environments. In densely obstructed areas it can be higher than 4 and it can also be less than 2 if the environment forms a waveguide or if special receiving techniques are used.

Decibel is a logarithmic unit that indicates ratio. It is convenient to use the logarithmic scale to indicate the power level of a signal because it can describe very large or very small numbers with shorter notation. The ratio in bels is the base 10 logarithm of the ratio of P_1 and P_2:

$$Ratio_B = \log_{10}(P_1/P_2) \tag{2.4}$$

Decibel is one tenth of a bel, hence

$$1B = 10dB \tag{2.5}$$

therefore

$$Ratio_{dB} = 10\log_{10}(P_1/P_0) \tag{2.6}$$

Based on the ratio between the transmitted and received signal power levels, path loss can be expressed in decibels as:

$$L_{dB} = 10\log\left(\frac{P_t}{P_r}\right) = 10\gamma\log\left(\frac{4\pi fd}{c}\right) \tag{2.7}$$

For free space where the path loss exponent γ is four, Equation 2.7 can be further simplified to

$$\begin{aligned} L_{dB} &= 4 \times 10\log\left(\frac{4\pi fd}{c}\right) \\ &= 4 \times 10(\log(4\pi fd) - \log(c)) \\ &= 4 \times 10(\log(4\pi) + \log(f) + \log(d) - \log(c)) \\ &= 40\log(f) + 40\log(d) + 40\log(4\pi) - 40\log(3 \times 10^8) \\ &= 40\log(f) + 40\log(d) - 295.12 \text{ dB} \end{aligned} \tag{2.8}$$

As shown in Equation 2.8, the loss in signal strength depends on distance and frequency. Signals with higher frequencies attenuates quicker and is more easily absorbed by water and water vapor than lower frequencies. A 5 GHz signal is more sensitive to rain, hail and fog compared to a 1 GHz signal. This means that higher frequency signals needs higher transmission power to cover the same range as lower frequency signals. It also implies that the signals fades away shortly after the intended receiver.

2.2.3 Transmission Impairments

In addition to attenuation, there are other factors that affects the signal quality in a negative manner. These impairments can be divided into three categories; *noise*, *distortions* and *Doppler fading*.

Noise

Undesired signals interfere with the desired signals, and these undesired signals are called noise. There are various types of noise:

- **White noise:** This is caused by thermal agitation of electrons and can not be eliminated (otherwise there would be no upper limit on the capacity according to Equation 2.2). The reason it is called white noise is that it is independent of frequency. Another name of this phenomena is *thermal noise* because of its dependency on temperature. Thermal noise can be expressed as

$$N = kTW \tag{2.9}$$

  where N is the thermal noise per W Hz bandwidth, k is the Boltzman's constant (1.3803×10^{-23}J/°K) and T is the temperature in Kelvin.

- **Intermodulation noise:** When signals with different frequencies are combined, new signals with other frequencies can be produced. E.g. two signals with frequency f_1 and f_2 can produce a signal with frequency $f_1 + f_2$, $f_1 - f_2$ or $n * f_i$. This effect can result in interference with the carrier and is therefore categorized as noise.

- **Crosstalk:** Unwanted coupling of transmission channels that causes multiple signals to interfere with each other because they are received by the same antenna.

- **Impulse noise:** Irregular pulses and noise produced by electromagnetic interference or faults in the communication system. This type of noise can be recognized by high amplitudes and short duration. Impulse noise is the main reason for bit errors in digital communication.

Physical Environment

Trees, buildings, poles, etc. can cause degradation of the signal quality due to behavior of electromagnetic waves. *Reflection*, *diffraction*, *scattering*, *refraction* and *absorption* (see Figure 2.2) are effects related to the physical environment and will be described below:

- **Reflection:** If an electromagnetic wave hits a soft surface, some of the energy will be absorbed and some will be reflected. The effect is stronger if the surface is large relative to the signals wavelength.

- **Diffraction:** If an electromagnetic wave hits a sharp edge, the direction of the wave will be bent towards the edge.

- **Scattering:** If an electromagnetic wave hits objects like poles, trees, etc., the wave can be divided into multiple copies that is scattered around the area.

- **Refraction:** If an electromagnetic wave travels from one transmission medium to another, the direction of the wave may change.

- **Absorption:** When an electromagnetic wave passes through a medium, the energy of the wave can attenuate.

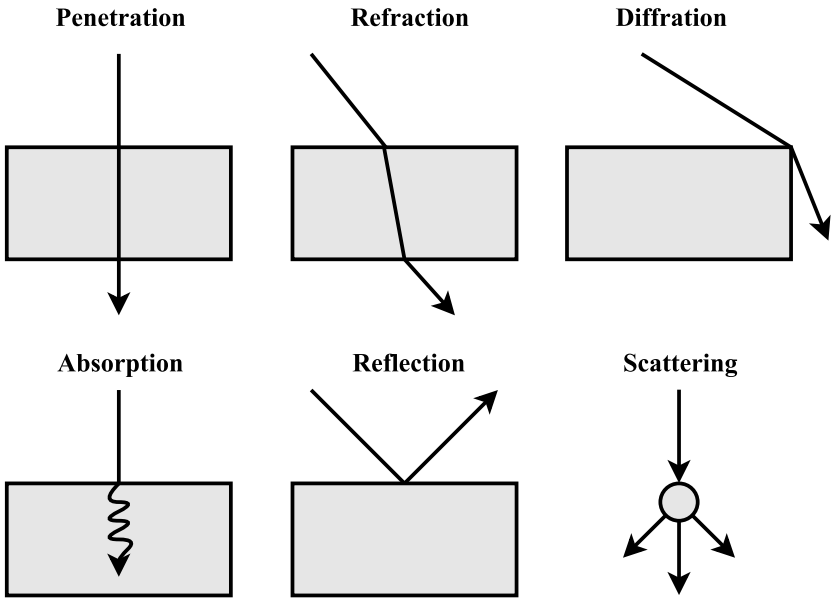


Figure 2.2: Effects caused by the physical environment.

These phenomena underlie another important transmission impairment called *multipath fading*. In addition to the original signal, there may be reflected, scattered and/or diffracted copies that reaches the intended receiver. These copies travel further than the original signal and the interference caused by these copies result in reduced signal quality at the receiving end.

Doppler Fading

If the transmitter and receiver is moving relative to each other while communicating, the frequency of the transmitted signals will change because of the change in distance. If they are moving towards each other, the frequency will increase, and if they are moving away from each other, the frequency will decrease. This is called the *Doppler fading*. The velocity with which the distance is changing determines how much the frequency changes. Doppler fading may cause the receiver to sample the signals at wrong frequencies, which means that the receiver may not be able to recover the information that was sent.

The impairments described in the previous sections can cause the SNR to be lower than the required level so that the receiver is unable to interpret the information. A well known attack called *jamming* is based on this theory. Jamming is obtained by transmitting intentional noise in the frequency range one wants to attack.

2.3 Bluetooth

Bluetooth is a wireless technology invented in 1994, and was later standardized as IEEE 802.15.1 [2]. While WiFi (IEEE 802.11) was intended to form Wireless Local

Area Network (WLAN), Bluetooth was made to form Wireless Personal Area Networks (WPAN). WiFi is typically set up with a centered access point that creates an infrastructure, while Bluetooth is mainly used for peer-to-peer/ad-hoc communication. This is not always true. For example, many of the modern cameras can be configured to transfer a live video feed to another device over WiFi. Bluetooth is in many ways similar to WiFi, e.g., it transmits data wirelessly by utilizing radio signals in the 2.4 GHz Industrial, Scientific and Medical (ISM) unlicensed band. Some key differences are highlighted in the next paragraphs.

Bluetooth is designed to operate at lower speed, short range and most importantly, less power. Speed varies from below 1 MBit/s to 100 MBit/s and and range is typically between approximately 10 - 100 meters. These variables depends on the Bluetooth version as well as the transmission power of the Bluetooth module. The typical applications of Bluetooth are communication between personal devices such as mobile phones, head sets, printers, meat thermometers, and so on. By using Bluetooth, it is possible to establish a link and transfer data with minimal configuration compared to WiFi. Because of the short range, Bluetooth can be more secure and less vulnerable to interference than WiFi. This is because the signals may fade away shortly after the intended receiver.

The Bluetooth Core Specification Version 4.x is a collection of Bluetooth technologies. It includes the *Classic Bluetooth*, *Bluetooth High Speed* and *Bluetooth Low Energy* protocols. In this problem we will focus on the latter protocol, which is also known as Bluetooth Smart. BLE differs from regular Bluetooth in three ways:

- **Power consumption:** BLE has very low energy requirements and can last for years on a small battery.
- **Lower cost:** BLE is 60-80% cheaper than regular Bluetooth [20].
- **Applications:** BLE is ideal for simple applications requiring small periodic transfers of state. Classic Bluetooth is preferred for more complex applications requiring consistent communication and more data throughput.

2.3.1 Reliability and Robustness

The 2.4 GHz ISM band has become crowded, whereby frequency planning is needed to reduce interference and improve robustness. A technique used in the Bluetooth protocols is called Frequency Hopping Code Division Multiple Access (FH-CDMA). In Classic Bluetooth, the 2.4 GHz radio band is divided into 79 channels of 1 MHz each where a new channel is chosen every 625 µS. Bluetooth low energy uses frequency hopping on 37, 2 MHz channels, at the same hopping rate (see Figure 2.3). Each communicating device pair has its own frequency hopping schema which is determined during the initial connection and chosen in order to avoid connection conflicts. Frequency hopping minimizes potential interference issues within a Bluetooth system, with other radio-based systems and from other interference sources such as microwave ovens. The narrow bandwidth and the frequency hopping feature also provide a good multipath performance in an obstacle intense environment. It

should be noted that frequency hopping is only used in data transfer mode and not when the device is advertising its presence. The channels used for advertisement is limited to 2402, 2426 and 2480 MHz. These can be used to broadcast data to anybody scanning.

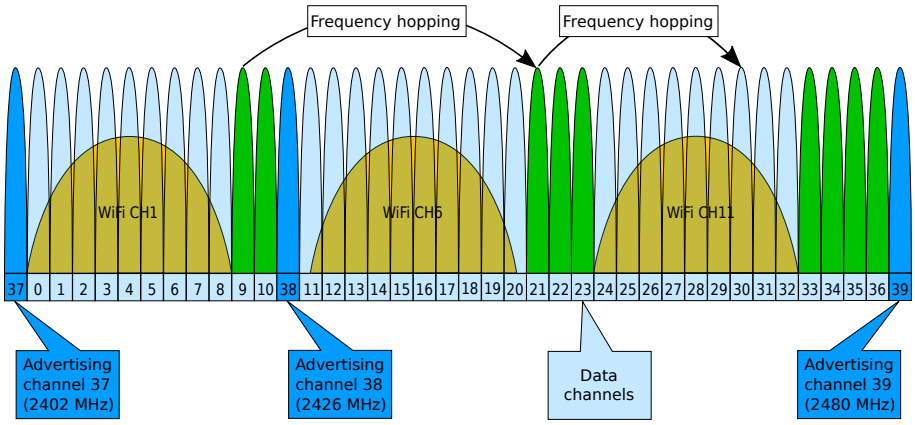


Figure 2.3: BLE and Wifi frequency channels.

Both Classic Bluetooth and Bluetooth low energy apply the Adaptive Frequency Hopping (AFH) feature which detects potential channel interference. For instance, this interference could come from a WiFi device transmitting in close proximity. If such interference is found, the channel is automatically blacklisted and will not be used until the interference has ceased. AFH prevents Bluetooth from interfering with other nearby wireless technologies such as WiFi.

Bluetooth supports both Forward Error Correction (FEC) and packet retransmission. The FEC uses a Hamming code that can correct one-bit errors and detect two-bit errors. Then an Automatic Repeat Request (ARQ) packet retransmission scheme is applied. Each packet payload contains a Cyclic Redundancy Check (CRC) checksum to check for errors. Each transmitted packet contains an Acknowledged/Not Acknowledged (ACK/NAK) bit to indicate the status of previously received packets. Retransmission is done if packets are lost or NAKed. This procedure allows for safe data transmission.

2.4 Beacons

Bluetooth beacons [1] operate on the BLE protocol. The beacons can be very small (Figure 2.4) and can normally run off a coin cell battery for more than a year depending on their configuration. They broadcast a signal that can be registered by smart devices nearby and it is up to the receiving part to decide how to act on the information received. A common way of interacting with beacons is to let your smart phone listen for beacons. When the phone receives a packet from a beacon, it queries a database with the Identifier (ID) of the beacon to see what app the beacon is associated to and then alerts the app (if it is installed). In other words,

the receiver has to do the heavy lifting, the beacons just provide real world context to your apps.

Figure 2.4: The size of an Estimote beacon compared to a hand. This is actually one of the bigger beacons on the market.

2.4.1 Broadcasting Power and Range

Broadcasting power is the power with which the beacon broadcasts it's signal and is given in dBm. The value typically ranges between -30 dBm and +4 dBm. Broadcasting power directly impacts signal range (see Equation 2.3). The more power, the longer range. Increasing the power can also make the signal more stable, but it is important to keep in mind that it will have a negative impact on battery life. The beacon's range is technically up to 70 meters. In real-world conditions, however, one should expect up to 40-50 meters.

2.4.2 Advertising Interval

Beacons do not broadcast constantly. They "blink" instead. Advertising interval describes the time between each blink. The value ranges between 100 ms and 2000 ms. The shorter the interval, the more stable the signal. Just as with broadcasting power, advertising interval heavily affects the battery life.

2.4.3 Received Signal Strength Indication

Received Signal Strength Indication (RSSI) is the strength of the beacon's signal as seen on the receiving device. The signal strength depends on distance and broadcasting power. At maximum broadcasting power (+4 dBm) the RSSI ranges from

-26 (a few centimeters) to -100 (40-50m distance) [21]. RSSI is used to approximate distance between the device and the beacon using a measured value defined by the protocols (2.4.4). Due to external factors influencing radio waves—such as absorption, interference, or diffraction—RSSI tends to fluctuate. The further away the device is from the beacon, the more unstable the RSSI becomes.

2.4.4 Protocols

Today there exist two protocols/firmwares for BLE Beacons, namely iBeacon [6] and Eddystone [4,5]. iBeacon is created by Apple and is closed source, while Eddystone is created by Google and is open source. Table 2.1 highlights some of the differences [3,7] between the two protocols. The main difference is how the packet broadcasted by the beacons look like. iBeacon only supports one packet type, while Eddystone specifies three types (UID, URL and TLM).

iBeacon packets

Beacons broadcast tiny packets of data, containing their iBeacon ID and information about signal strength so that the receiver can understand which beacon it hears and how far away it is. Figure 2.5 describes the structure of an iBeacon data frame.

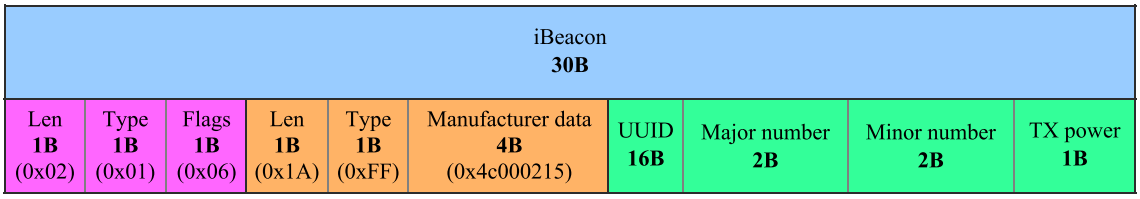


Figure 2.5: A detailed overview of how an iBeacon data frame is structured.

Every iBeacon ID is 20 bytes long and is divided into three sections:

- **UUID** (16 bytes): A Universally Unique Identifier (UUID) that identifies the beacon. This is typically unique to a vendor.
- **Major number** (2 bytes): Intended to identify a subset of beacons within a large group.
- **Minor number** (2 bytes): Intended to identify a specific beacon.

There is also an additional TX (Transmission) power field of 1 byte. This is a factory-calibrated, read-only constant which indicates the expected RSSI at a distance of 1 meter from the beacon. Knowing the RSSI at 1 meter, and the current RSSI (measured by the receiver), it is possible to estimate the distance.

The iBeacon prefix contains the hex data 0x0201061AFF004C0215. This breaks down as follows:

- **0x020106** defines the advertising packet as BLE General Discoverable and BR/EDR high-speed incompatible. This means that it only supports broadcasting, not connecting. A more detailed description is found in [35], part A, § 1.3.
- **0x1AFF** says that the following data is 26 bytes long and is Manufacturer Specific Data.
- **0x004C** is Apple's Bluetooth Sig ID and is the part of this specification that makes it Apple-dependent.
- **0x02** is a secondary ID that denotes a proximity beacon, which is used by all beacons.
- **0x15** defines the remaining length to be 21 bytes $(16 + 2 + 2 + 1)$.

Eddystone packets

As mentioned earlier, the Eddystone protocol specifies three frame types. Figure 2.6 describes the structure of these packets. Details about the different frame types will be reviewed in the following sections.

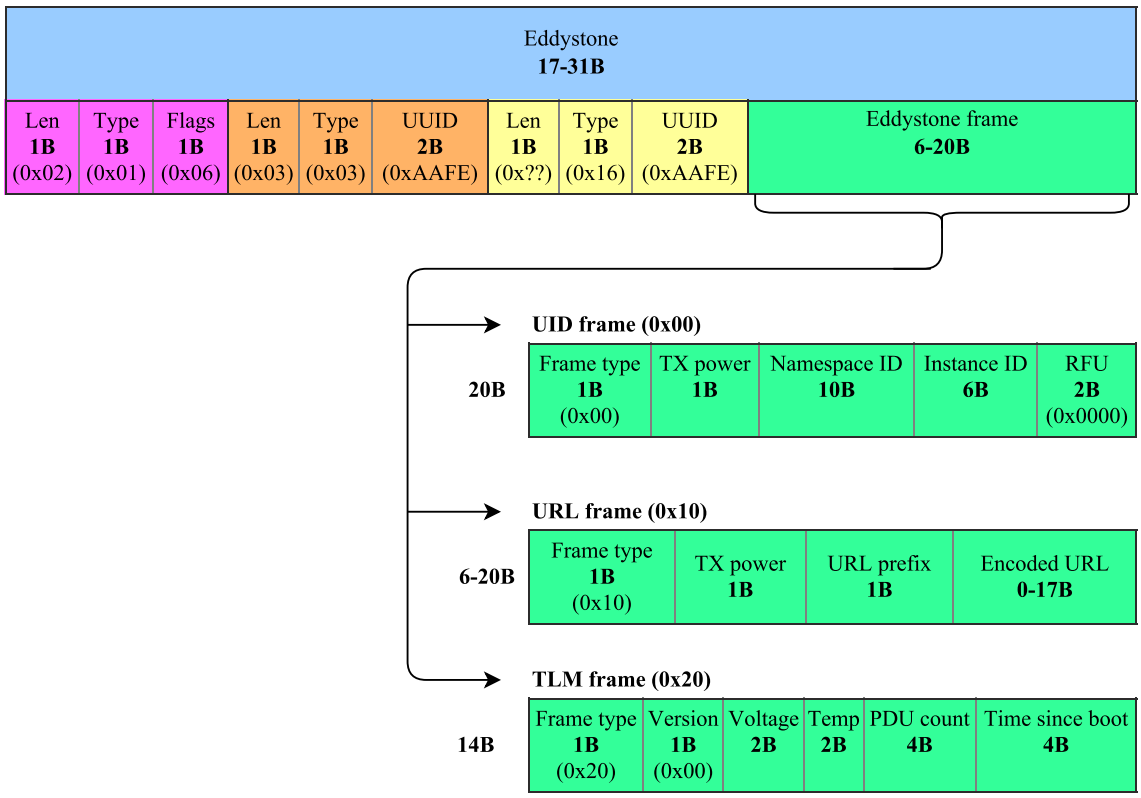


Figure 2.6: A detailed overview of how Eddystone frames are structured.

The Eddystone prefix contains the hex data 0x0201060303AAFE??16AAFE. This breaks down as follows:

- **0x020106** is the same as for the first 3 bytes of the iBeacon packet.
- **0x0303AAFE**
  - **0x03** says that the next message segment is 3 bytes long.
  - **0x03** refers to the complete list of 16-bit Service UUIDs.
  - **0xAAFE** is the Eddystone Service UUID.
- **0x??** denotes the length of the remaining packet. This can vary from 6 bytes to 20 bytes.
- **0x16** defines the type of the data packet. Specifically it tells us that the first 2 octets contain the 16 bit Service UUID followed by additional service data
- **0xAAFE**: This is the Eddystone service UUID.

Eddystone-UID

Eddystone-UID (Unique Identifier) contains an identifier of a beacon. An app installed on the phone can use the identifier to trigger desired action, just like with iBeacon. Whereas the iBeacon identifier is composed of three parts: UUID, major number and minor number, and is 20 bytes long. Eddystone-UID is 16 bytes long and divided into two parts:

- **Namespace** (10 bytes): Similar in purpose to iBeacon's UUID. In iBeacon you would usually assign a unique UUID to all of your beacons to easily filter them out from other people's beacons. In Eddyston-UID, you can do the same with the namespace.
- **Instance** (6 bytes): Similar in purpose to iBeacon's major and minor numbers, i.e., to differentiate between your individual beacons. *Instance* is represented as a string up to 12 characters long.

Eddystone-URL

Eddystone-URL packet contains a single field: URL. The size of the field depends on the length of the URL.

The promise and purpose of the Eddystone-URL packet ties directly into the concept of Physical Web [12]. Whereas with iBeacon or Eddystone-UID there's a need for an app to take the beacon's identifier and translate it into certain actions, with Eddystone-URL the data is encoded directly in the beacon's advertising packet. This means that the user can access content, usually in form of a website, without the developer needing to build a native experience. So instead of having many different apps fetching contextual data, we might have just one, and it can simply be the web browser.

The URL could be a regular web page providing relevant information, e.g., a beacon next to a movie poster could broadcast a link to a YouTube trailer. It also could be a dynamic web application, with contextual parameters embedded in the URL, e.g., http://my-airliner.com/check-in?flight-number=DK6523.

Eddystone-TLM

Eddystone-TLM packet is designed to be broadcasted by the beacon alongside the data packets (i.e., UID and/or URL) for the purposes of fleet management. Nearby Bluetooth-enabled devices can read these packets and relay them to a fleet management service. This service can then notify the owner of the beacon that, e.g., the battery is running out. The TLM packet is broadcasted less frequently than the data packet. As can been seen from Figure 2.6, both the UUID frame and URL frame exposes the TX power (calbrated at 0 m) while the TLM frame does not.

The telemetry packet consists of:

- Battery voltage, which can be used to estimate the battery level of a beacon.
- Beacon temperature.
- Number of packets sent since the beacon was last powered-up or rebooted.
- Beacon uptime, i.e., time since last power-up or reboot.

A brief comparison of the two protocols is given in Table 2.1.

	Apple's iBeacon	**Google's Eddystone**
Compatibility	It is Android and iOS compatible, but native only for iOS.	Compatible with any platform that supports BLE, hence it is both Android and iOS compatible.
Profile	It is proprietary software, thus the specification is controlled by Apple.	It is open source and the specification is publicly available on GitHub so that anybody can contribute to it.
Usage	UUIDs are tied in to the developer's server. Therefore, when it is sent to a smartphone, the device would need a specific app to do a particular task with the information received. A mobile app is necessary to receive messages via iBeacon.	Eddystone, on the other hand, sends out URL in place of UUID, which can simply open in a web browser vis-à-vis specific apps. For iOS devices, it is supported by Chrome with the 'Today' notifications enabled, whereas for Android devices, it is supported on the 'Physical Web' browser.
Security and privacy	There is no specific feature such as Ephemeral Identifiers (EIDs) in iBeacon. The signal transmitted by a beacon is a public signal and can be detected by any iOS device and certain Android devices with proper specifications.	Eddystone has a built-in feature called EIDs that constantly change and allow beacons to broadcast a signal that can only be identified by 'authorized clients'.
API	iBeacon provides two API methods for apps to detect iBeacons devices: ranging, which works only when the app is active, and provides proximity estimations; and monitoring, which works even if the app is not running, and provides a binary "in range" and "out of range" information. Apple has no specific API made available for iBeacon fleet management.	Eddystone has an advantage here since Google has launched two APIs (Nearby API and Proximity Beacon API) that makes Eddystone beacons more powerful. These APIs also make beacon fleet management much easier.

Table 2.1: Comparison of beacon protocols.

Beacons that use BLE radios are produced by many companies and one of the major vendors is called Estimote (which is the type of beacons used in this project). Estimote beacons are built around the nRF51822 chip which is produced by the Norwegian company called Nordic Semiconductor and supports both the major protocols (iBeacon and Eddystone).

2.5 Related Work

Positioning systems intended for indoor use has already been suggested, and as mentioned in the introduction, a variety of technologies has been carefully studied. The authors of [16] has designed and tested an acoustic positioning system which achieved sub-centimeter accuracy. However, the implemented system had a limited coverage area of 3 m^3 due to directed transducers of low range.

A real time positioning system using RSSI values from an unmodified WLAN is suggested in [34]. A major advantage of this system is that it can be used in existing WiFi networks, but the disadvantage is that most of the position estimates are off by several meters.

In [28] they used UHF RFID[2] in their positioning system. Similar to the acoustic system in [16], it achieved very high accuracy, but only for a limited range and in environments with negligible multipath fading effects.

P. Lin *et al.* [27] developed an interesting system for safety risk assessment of workers in tunnels. They used fingerprinting [25] techniques to estimate the position of the workers. Android smart phones was used as mobile stations and WiFi access points as base stations. They concluded that WiFi was reliable and accurate enough for some cases, but applications that require less than 1 meter of error should use other technologies.

There has been a number of studies that focus on how people in emergency situations react. In [32], different forms of building evacuations (normal, controlled and panic) are experimented with. From the results it can be concluded that it is hard to predict how people in panic are moving.

The related work presented above uses technologies other than BLE, which is what we will focus on in this thesis. However, there has also been suggested an interesting solution [24] where they use BLE beacons to track the distance of moving people. It presents the fundamental idea that will be used in this thesis. The authors of [40] did a comparison of BLE and WiFi in indoor positioning applications. BLE was unexpectedly found to be a more accurate location technology than WiFi. BLE also has other advantages over WiFi, such as low power consumption and low cost, which makes it interesting to see if it can be a dominant technology for indoor positioning systems in the future.

[2]Ultra High Frequency Radio-Frequency Identification

This page is intentionally left blank.

3 Architecture

The main objective of this system is to locate people inside buildings using Bluetooth Low Energy beacons. The following sections discuss the technologies used, why they were chosen and how the system was designed and implemented.

3.1 Base Stations

Beacons are usually deployed in a fixed location and will not move. Actually, the Estimote beacons are shipped with double sided tape so that you can place them on a wall, for instance. They are also waterproof so that you can safely deploy them in a wet environment. When designing a positioning system there are typically two components required; base stations and mobile stations. In GPS (Global Positioning System), the satellites are base stations while people, cars, boats etc. are mobile stations. The same applies for cellular networks. When using beacons for positioning, there are at least two design options:

1. Place beacons at fixed locations and use them as base stations. This requires people to wear a smart phone to be located (or a custom made device).
2. Tag people (or other objects) with beacons and use Bluetooth enabled hardware as base stations, e.g. a micro controller or a small computer.

Number 1 is obviously the easiest to implement and deploy as most people own a smart phone already (they would just need to install an app), but requiring people to wear a smart phone could be a drawback. There are many companies that prohibits the use of cell phones at work due to various reasons. Off shore workers for instance, normally have to leave their phone in a locker at the heliport for safety reasons. The operators justify the rules by pointing out the hazards related to the batteries and possible interference in helicopter electronics.

Number 2 is more cumbersome to implement as it requires custom hardware for the base stations, but on the other hand it is more versatile. As it is not dependent on smart phones as mobile stations, it can be used to track any object, not just human beings. Animals can be tracked inside farm buildings, explosive material can be tracked inside factories or warehouses, babies can be tracked in hospitals and so on. It is also possible to implement both alternatives in the same system, but that will not be examined in this project. For this specific implementation,

alternative 2 was chosen due to its versatility. Alternative number 2 also sets apart from existing systems that uses the WiFi radio on smart phones in conjunction with existing access points to estimate the location of nearby people.

The requirements of a Base Station (BS) in alternative 2 are as following:

- It has to be Bluetooth 4.0 enabled.
- It should have a small form factor.
- It has to be connected to either the other base stations or a central server node.
  - This connection should preferably be wireless.
- It should be able to run off a battery in case of a power loss.

A prototype that meets these requirements was built using a Raspberry Pi 2 model B (see Figure 3.1) with USB compatible WiFi and Bluetooth dongles. A 2800 mAh power bank was added to fulfil the last criteria. Raspberry Pi is a series of credit card sized single-board computers that runs on various Linux distributions (Raspberry Pi 2 can also run Windows 10). The Raspberry Pi 2 Model B ships with:

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

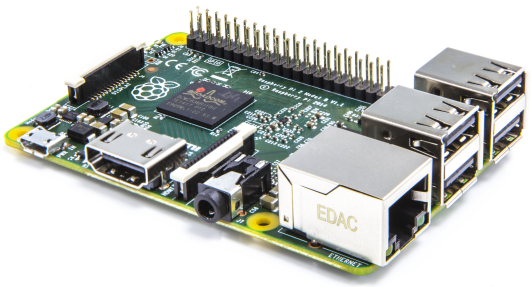


Figure 3.1: Illustration of the Raspberry Pi 2 Model B.

If this system would go into production, the Raspberry Pi could be replaced by a micro controller (or a Raspberry Pi Zero [13]) to further reduce the footprint, power requirements and prize. The Raspberry Pi was chosen because it runs a complete operating system and therefore has wide support for popular development frameworks. In Figure 3.2, there is a photo of a fully assembled base station. The Raspberry Pi is embedded in the black case and connected to the mains electricity via a small Uninterruptible Power Supply (UPS). The Bluetooth and WiFi dongles is located on the right hand side.

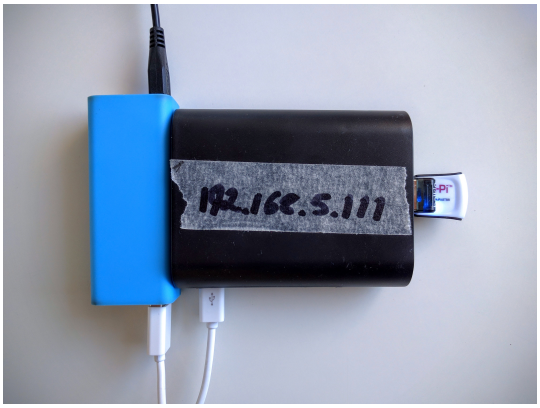


Figure 3.2: This is how the base stations look like.

A simple overview of the system is outlined in Figure 3.3. Each base station listens for mobile stations (beacons) and uses the RSSI to estimate how far away the beacon is. All of the base stations are connected to a hub node (server) to which they transmit the estimated distances of in range beacons. The responsibility of the hub node is to process the information and make it available to clients, which in this case are rescuers.

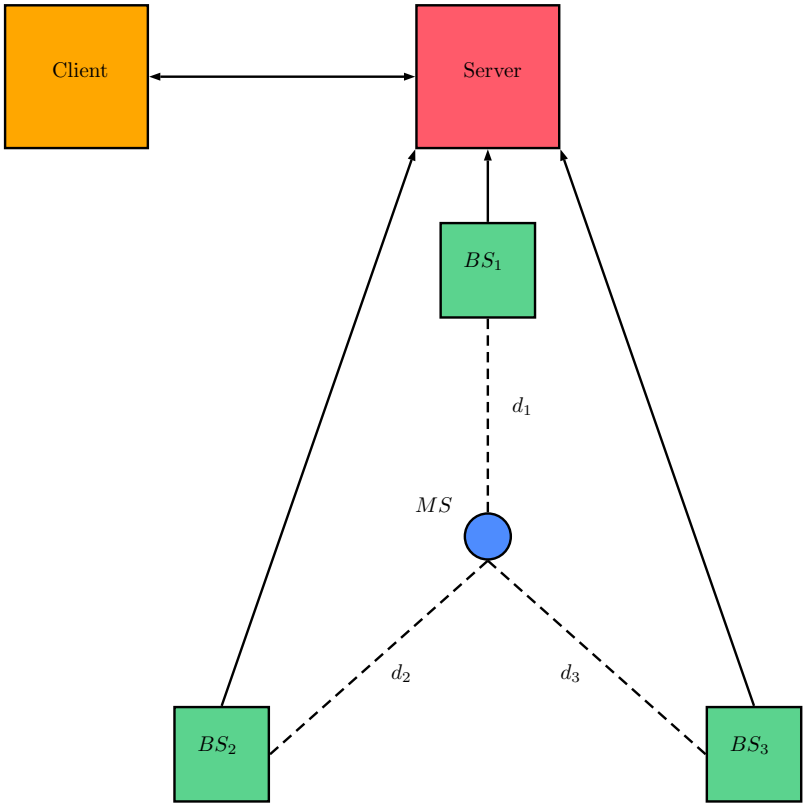


Figure 3.3: Overview of the system components and how they interact. BS stands for *base station* and MS stands for *mobile station*. The distance between a base station and a mobile station is denoted by the letter *d*.

3.2 Position Determination Algorithm

There exists a large variation of positioning techniques, however, there is only a limited number of algorithms to infer position information from measurements. Position inference is usually based on measurable variables such as timing, angle of arrival, angle of emission, signal strength, acceleration, rotation, etc. These are physical measurements that contain a lot of noise. Therefore, it should be emphasized that we are rarely able to *ascertain* an objects position. Position estimates are typically based on probabilistic methods.

For distance based positioning algorithms, there are generally two types. One type based on signal timing, and one based on signal strength (e.g. RSSI). The most common methods are described in [39]. Timing based algorithms typically rely on *Time of Arrival (ToA)*, *Time Difference of Arrival (TDoA)* or *Roundtrip Time of Flight (RToF)*. By using standardized protocols and off-the-shelf devices, we only have access to high level APIs. The only information relevant to positioning that is provided by the APIs, is the RSSI. Therefore, we are limited to choose an RSSI-based algorithm. RSSI can also be used in distance independent methods such as *fingerprinting* [25]. The idea behind this technique is to record environmental values

(electrical, physical, optical, etc.) from a number of known locations and store them in a database. This is called the offline phase. In the online tracking phase the current environmental values from an unknown location is compared to those stored in the fingerprint and the closest match is returned as the estimated position. The drawback with fingerprinting (and other scene based methods) is the extra overhead in the offline training phase, and that it is very sensitive to small changes in the environment [39]. For this reason, a distance based method called *lateration*, or *trilateration* [29], was chosen.

This section will describe how we can obtain the position of a Mobile Station (MS) based on RSSI values. The RSSI values obtained by a device depend on a large number of unpredictable factors that may lead to reflection, diffraction, scattering or absorption (described in Section 2.2). One of these factors is the distance between the transmitter and the receiver. Increasing distance result in increasing attenuation, and vice versa. We will now use Equation 2.8 to derive a model for estimating the distance between to endpoints based on RSSI and signal frequency.

$$L_{dB} = \gamma 10 \log\left(\frac{4\pi f d}{c}\right) = 10\gamma\left(\log(4\pi) + \log(f) + \log(d) - \log(c)\right) \tag{3.1}$$

We are interested in the distance d, thus we can move the expression containing d to the left side

$$10\gamma \log(d) = L_{dB} + 10\gamma\left(\log(c) - \log(4\pi) - \log(f)\right) \tag{3.2}$$

$$\log(d) = \frac{L_{dB}}{10\gamma} + \log(c) - \log(4\pi) - \log(f) \implies d = 10^{\left(\frac{L_{dB}}{10\gamma} + \log(c) - \log(4\pi) - \log(f)\right)}, \tag{3.3}$$

where c is the speed of light, which is $3 \times 10^8\,\mathrm{m\,s^{-1}}$ and f is the frequency of the signal, which is $2.4 \times 10^9\,\mathrm{Hz}$. By inserting these known values into 3.5, we can further simplify the model:

$$d = 10^{\left(\frac{L_{dB}}{10\gamma} + \log(3\times10^8) - \log(4\pi) - \log(2.4\times10^9)\right)} = 10^{\left(\frac{L_{dB}}{10\gamma} - 2\right)} \tag{3.4}$$

L_{dB} is equal to the the $RSSI_t$ seen at the transmitting end minus the $RSSI_r$ seen at the receiving end of the link, thus:

$$d = 10^{\left(\frac{RSSI_t - RSSI_r}{10\gamma} - 2\right)}, \tag{3.5}$$

where γ is the path loss exponent (see Section 2.2.2). In Figure 3.4, Equation 3.5 has been plotted with two different path loss exponents and a transmission power of $+4$ dBm and -8 dBm. It becomes obvious that the value of the path loss exponent

has a big impact on the estimated distance, therefore it has to be chosen wisely. This will be discussed in more detail in Section 3.2.1.

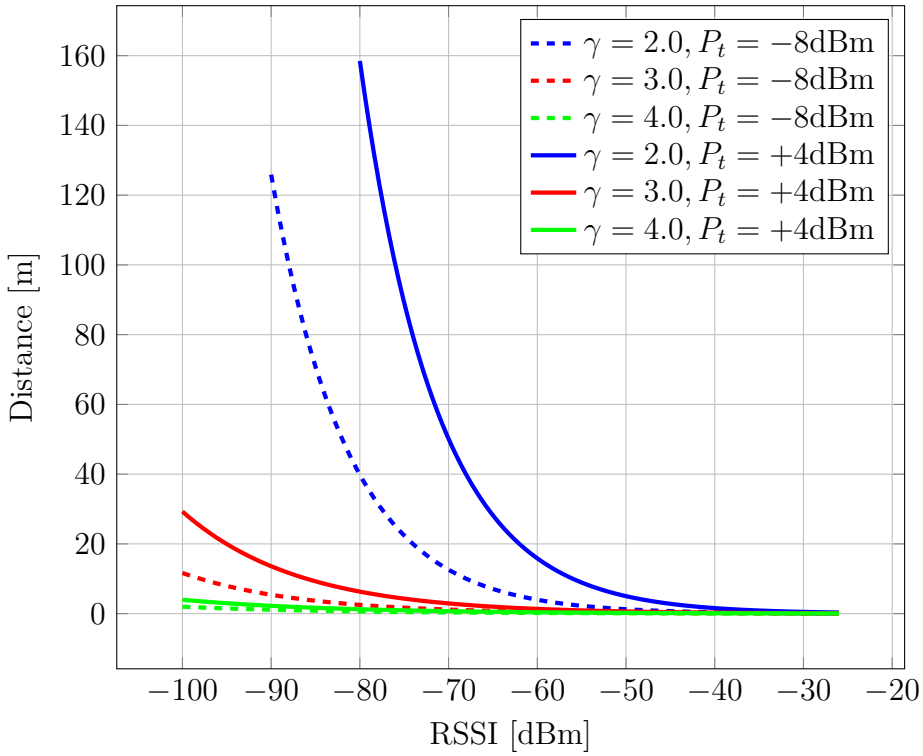


Figure 3.4: These plots shows the relationship between the RSSI, TX power and the estimated distance. It also shows the effect of the path loss exponent.

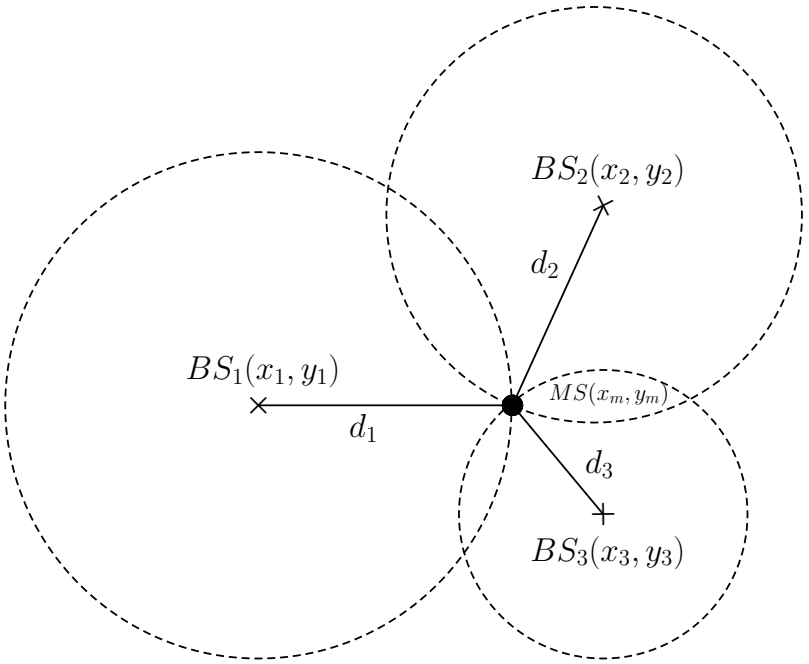


Figure 3.5: Totally compatible distance estimates.

Before implementing the trilateration algorithm, a naive proximity based method was tested. The idea is to simply chose the base stations that is closest to the beacon and state that the beacon has to be located within a certain range of that station. E.g., if we have one base station in each room, we can assume that if a mobile station is in the same room, this station will appear as the closest one. This works well with a high density of base stations, but to increase scalability, the more sophisticated method of trilateration was implemented.

Figure 3.5 shows a scenario of three base stations BS_1, BS_2 and BS_3 together with a mobile station MS that needs to be located. Each base station calculates the distance to the mobile station based on the RSSI and the distances are represented by d_1, d_2 and d_3. Given that the measurements are not erroneous, we known that the mobile station has to be located on the intersection of the circles defined by (x_i, y_i) and d_i where $i \in \{1, ..., 3\}$. The circles will intersect in exactly one point (x, y), given that the distance estimates are totally compatible. By using the equations that defines the circles, we obtain the following non-linear equation system:

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 &= d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 &= d_2^2 \\ (x - x_3)^2 + (y - y_3)^2 &= d_3^2 \end{aligned} \tag{3.6}$$

By solving Equation 3.6 for x and y we find the intersection point between all three circles which is also the position of the mobile station.

This problem can be further generalised in the case of n base stations. For every base station BS_i and distance measurement d_i, $i \in \{1, ..., n\}$, there exists a circle with center coordinates (x_i, y_i) and a radius d_i. The equation system is formed by:

$$\begin{aligned} (x - x_1)^2 + (y - y_1)^2 &= d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 &= d_2^2 \\ &\vdots \\ (x - x_n)^2 + (y - y_n)^2 &= d_n^2 \end{aligned} \tag{3.7}$$

In a real world application, there will always be measurement errors, hence we cannot assume that the equation system in 3.7 will have one single solution. The problem then becomes a non-linear optimization problem for x and y, which is not trivial to solve. However, the problem can be linearized by subtracting one equation from all others, so to linearize equation system 3.7, we subtract equation n from all other equations $1, ..., n - 1$.

First, we expand the equations for circle $1, ..., n$:

$$\begin{aligned} d_1^2 &= x^2 - 2xx_1 + x_1^2 + y^2 - 2yy_1 + y_1^2 \\ d_2^2 &= x^2 - 2xx_2 + x_2^2 + y^2 - 2yy_2 + y_2^2 \\ &\vdots \\ d_n^2 &= x^2 - 2xx_n + x_n^2 + y^2 - 2yy_n + y_n^2 \end{aligned} \tag{3.8}$$

If we subtract the n'th equation from all the others, we get the following expression:

$$d_i^2 - d_n^2 = x_i^2 - x_n^2 - 2x(x_i - x_n) + y_i^2 - y_n^2 - 2y(y_i - y_n), \quad i = \{1, ..., n-1\} \tag{3.9}$$

By moving all the unknown expressions to the left side and keeping the known variables at the right side we get:

$$2x(x_i - x_n) + 2y(y_i - y_n) = x_i^2 - x_n^2 + y_i^2 - y_n^2 - d_i^2 + d_n^2, \quad i = \{1, ..., n-1\} \tag{3.10}$$

Let

$$\begin{aligned} a_{i1} &= 2(x_i - x_n) \\ a_{i2} &= 2(y_i - y_n) \\ b_i &= x_i^2 - x_n^2 + y_i^2 - y_n^2 - d_i^2 + d_n^2 \end{aligned}$$

where $i = \{1, ..., n-1\}$. We can now write the equations in 3.10 on the form:

$$AX = b \tag{3.11}$$

where $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ \vdots & \vdots \\ a_{n-11} & a_{n-12} \end{bmatrix}$, $X = \begin{bmatrix} x \\ y \end{bmatrix}$ and $b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix}$.

In case all the circles intersect at a single point, we will have one unique solution, but generally, Equation 3.11 is an overdetermined system, so the goal is to find the coefficients in X which fit the equations "best". By using a technique called *linear least squares* [23], we can find the solution $\tilde{X}$ that minimizes the measurement errors of all the base stations. The least squares solution of 3.11 becomes:

$$\tilde{X} = \left(A^T \cdot A\right)^{-1} A^T \cdot b \tag{3.12}$$

To get a better understanding of the equation system, we can solve the problem graphically by using a method based on the radical axes of the circles. The radical axis (or geometric power line) of two circles is the locus of points with equal geometric

power from both circles [18]. The geometric power of a point with respect to a circle is a real number that reflects the relative distance of the point from the circle. Thereby, the geometric power of a point with regard to a circle is zero if the point belongs to the circle, it is negative if the point lies inside the circle or it is positive if the point is outside the circle. Radical axes for three different cases are shown in Figure 3.6. N circles would cut each other in a single point if, and only if, a point exists with zero geometric power with regard to all the circles. Thus, N circles cut each other in a single point, if the $N(N-1)/2$ radical axes obtained with the different pairs of circles cut each other in a single point and this point belongs to all the N circles (which is the case in Figure 3.5). Figure 3.7 demonstrates a case where all the radical axes are positive with regard to all the circles, hence the radical center i positive. In Figure 3.8 the radical center is negative with regard to all circles. Note that the radical axis between two circles with the anchor points C_1 and C_2 is perpendicular to the line connecting C_1 and C_2. If the radical axis is negative (the circles are overlapping), the axis runs through both of the intersection points.

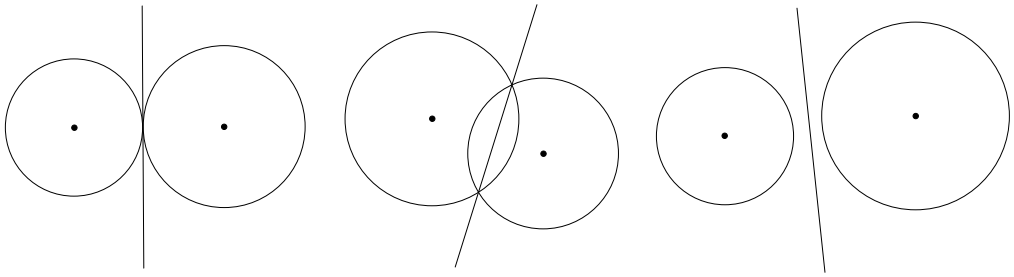


Figure 3.6: Radical axes in three different cases.

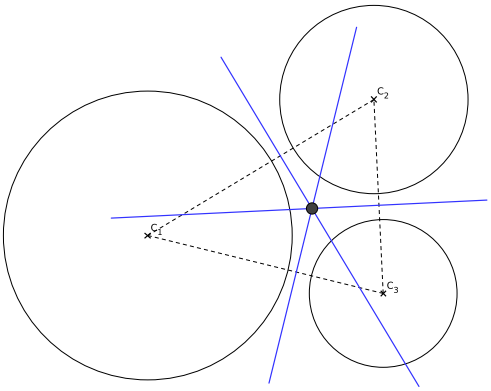


Figure 3.7: Positive radical axes.

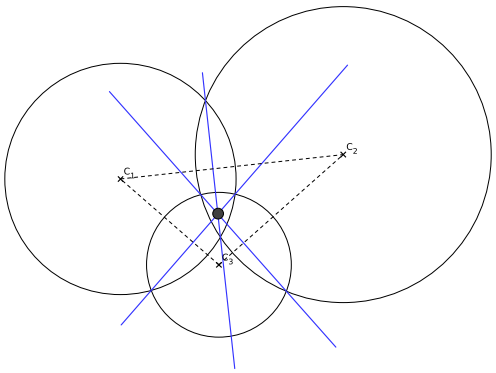


Figure 3.8: Negative radical axes.

3.2.1 Choosing the Path Loss Exponent

As we have seen in Figure 3.4, the path loss exponent γ is an important factor when calculating the distances between the mobile stations and the base stations. In the

first implementation, the path loss exponent was implemented as a fixed parameter. This can be chosen to fit the propagation model for a specific environment at a given instant. We can assume that the propagation model will remain the same for a short period of time, however, it is unlikely that the environment will not change in the future. Small changes in position and direction combined with moving objects, differently shaped and furnished rooms, etc. may result in drastic changes to the signal propagation. Therefore, we propose a solution that continuously calculates the path loss exponent for the present environment to account for these changes. This is achieved by introducing reference beacons placed at known positions so that we can calculate the path loss exponent for the propagation channel between the reference node and each of the base stations. These exponents are later used when estimating the distance between a mobile stations and nearby base stations to obtain a more accurate result. Testing results of the adaptive solution is presented in Section 4.2.

3.3 Client

The main purpose of the client (see Figure 3.3) is to provide safety personnel with information about missing people during an evacuation. It also gives administrators the ability to manage their base stations and test the system. The client was implemented as a web application to make it easy to use and to eliminate the need to install anything. Technologies and implementation details will be discussed in the following sections.

3.3.1 Technologies

AngularJS

AngularJS is a powerful JavaScript framework used to develop Single Page Application (SPA) (see section 3.3.1) with dynamic content. It lets you extend the HTML vocabulary by creating new elements and tags. It is open source and maintained mainly by Google.

AngularJS implements the Model View Controller (MVC) [9] (or Model-View-ViewModel (MVVM) [10]) pattern and has 2-way data binding between the view and controller so that developers do not have to worry about Document Object Model (DOM) manipulation. AngularJS detects changes to the model and modifies HTML expressions in the view via a controller. Likewise, any alterations to the view are reflected in the model. Dependency injection is also one of the core features of AngularJS which makes it easier to write testable components. The different concepts of AngularJS are described in the list below:

- **Templates** are HTML documents with additional markup that can be interpreted by the AngularJS compiler.

- **Directives** is what makes developers able to extend HTML with custom attributes and elements. Directives are very handy for making reusable components.
- **Views** are what the users sees (the DOM). AngularJS generates the view by compiling the templates and instantiating directives and expressions.
- **Model** is the data shown to the user and with which the user interacts.
- **Controllers** are responsible for defining methods and properties that the view can bind to and interact with. Controllers should be lightweight and only focus on the view they are controlling.
- **Scope** is the context where the model is stored so that controllers, directives and expressions can access it.
- **Data binding** is essentially the "glue" between the view and controller. If a controller changes an object on the scope, the change is instantly reflected in the view, and if a user alters an object from the view, the change is also reflected back to the controller. This is known as 2-way data binding.
- **Services** are suitable for sharing view-independent functionality across an application. For example, a service that takes care of communication with the backend can be injected to multiple controllers. This also enables isolated testing of the controllers without a backend by injecting a mock of the data access service.
- **Routes** defines different states that the users can navigate to within the application. A state can be configured to use a specific template and controller.
- **Modules** serve as containers to assist in organizing code within the application. Modules can contain sub-modules.

Listing 3.1, 3.2 and 3.3 demonstrates a minimal AngularJS application that utilizes the `ng-repeat` directive.

```
1 <body ng-app="mountainApp">
2     <table ng-controller="MountainController">
3         <tr ng-repeat="mountain in mountains | orderBy:'
              -height'">
4             <td>{{mountain.name}}</td>
5             <td>{{mountain.height}} meters</td>
6         </tr>
7     </table>
8 </body>
```

Listing 3.1: HTML template.

The `ng-repeat` directive provides an attribute that can be used on any HTML element. The example in Listing 3.1 iterates over the `mountains` array on the scope and builds a table with one row per mountain and one column for each mountain property.

```
angular.module('mountainApp', [])
  .controller('MountainController', ['$scope',
    function($scope) {
      $scope.mountains = [
        {
            name: 'Kilimanjaro',
            height: 5895
        },
        {
            name: 'Mount Everest',
            height: 8848
        },
        {
            name: 'K2',
            height: 8611
        },
      ];
    }
  ]
);
```

Listing 3.2: AngularJS sample app.

It may seem overkill to do this for a table with only three rows, but the underlying array of mountains in Listing 3.2 could contain thousands of objects without the need to change the template. If the array of objects changes during runtime, AngularJS will dynamically rebuild the DOM to reflect the changes.

```
<body ng-app="mountainApp">
    <table>
        <tr>
            <td>Mount Everest</td>
            <td>8848 meters</td>
        </tr>
        <tr>
            <td>K2</td>
            <td>8611 meters</td>
        </tr>
        <tr>
            <td>Kilimanjaro</td>
            <td>5895 meters</td>
        </tr>
    </table>
</body>
```

Listing 3.3: HTML view.

In Listing 3.3, we can see how the compiled view would look like to the end user. Note that the mountains are now sorted by height. This was obtained by providing a *filter* to the `ng-repeat` directive (see Listing 3.1, line 3). The minus sign in front of the filter (`orderBy:'-height'`) tells AngularJS to sort the values in descending

order. This filter is an AngularJS built in filter, but custom filters can easily be created.

Bootstrap

Bootstrap, or Twitter Bootstrap, is a powerful front-end framework developed at Twitter. It provides a collection of tools for creating responsive web pages. For a web page to be responsive, it has to look good on all types of devices (desktops, tablets and phones), that means that it has to adjust its layout according to the window size of the browser. This is achieved by using HTML and CSS to resize, shrink, hide, enlarge, or move content to optimize user experience. Figure 3.9 and 3.10 demonstrates the front page of the evacuation application that was developed in this project. Figure 3.9 shows how the web page would look like on a desktop, while Figure 3.10 demonstrates the layout on a mobile client. On desktops, the content will align horizontally, while on mobiles it will be stacked vertically. Also note that the navigation menu has been collapsed into a drop down menu on the mobile page.

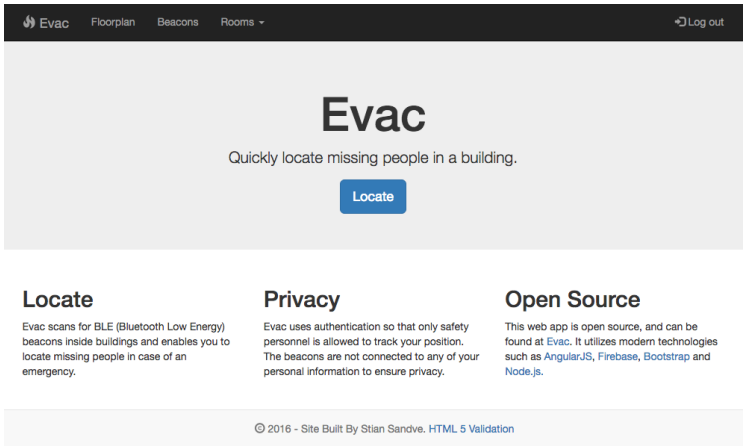


Figure 3.9: Desktop.

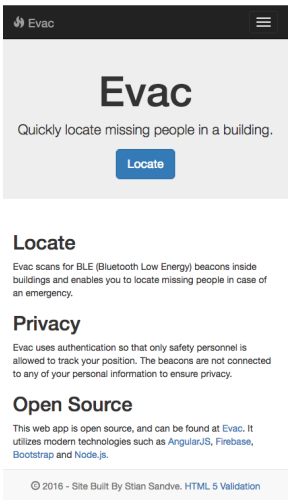


Figure 3.10: Mobile.

Firebase

Firebase is a cloud services provider that was acquired by Google in 2014. It provides a real time NoSQL database. Data is stored as JavaScript Object Notation (JSON) and synced to all connected clients in realtime. The data is also available if the app goes offline and will be synced to the server once it regains connection. It provides the following authentication services:

- **Email and password:** This lets the users register and authenticate by email and password.

- **Anonymous:** Anonymous authentication generates a unique identifier for each user that lasts as long as their session. It does not require users to share any personal information.
- **Custom:** If you already have an authentication system, you can generate custom login tokens that can be integrated with Firebase.
- **Social providers:** Firebase has also made it easy to authenticate users with Facebook, Twitter, Google or Github.

In Section 3.3.1, the idea of two-way data binding was introduced. The synchronization features of Firebase, combined with the data binding in AngularJS, makes a system that can handle three-way data binding. When data changes at one client, those updates are immediately persisted to Firebase and rendered across all connected clients. From Figure 3.11, we can see that if data is updated in either one of the view, model or database, the changes will propagate in real-time to the other two.

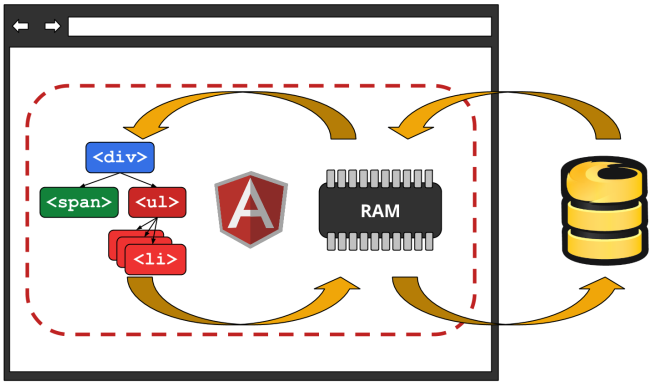


Figure 3.11: Three way data binding (source: http://codepen.io).

Single Page Application

The client is a Single Page Application. The advantages of an SPA over a Multi Page Application (MPA), is primarily related to responsiveness. In SPAs, the required files are fetched once when user opens the page, then all the content is provided through routing and http/https requests to a server Application Programming Interface (API). In MPAs, a round-trip to the server is necessary each time a user navigates to a new page or performs an action that requires the current page to refresh. The HTML is then built on the server side before it is transmitted back to the client. This often requires a full reload of the page, whereas in SPA, the underlying content is simply replaced without reloading the page. The life cycles of the two architectures are highlighted in Figure 3.12 and 3.13.

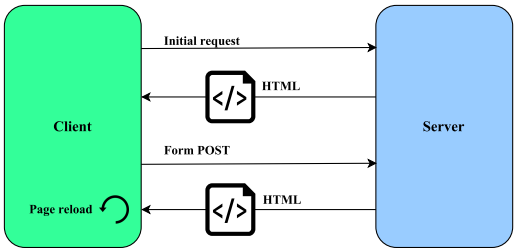


Figure 3.12: MPA life cycle.

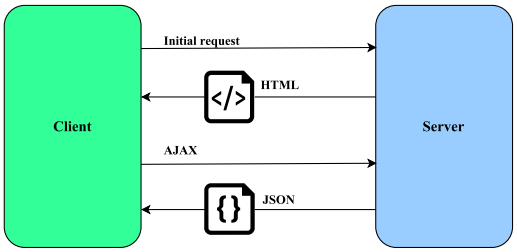


Figure 3.13: SPA life cycle.

SPA eases the load on the server, especially for applications with heavy business logic, which makes it more scalable. The processing of business rules can be moved to the client side so that the server is only responsible for providing data access and authentication. SPA also requires less content to be downloaded. In traditional MPAs, the entire HTML markup has to be downloaded each time a user requests a new state, while in SPA, only the changes needed to go from one state to another is downloaded. In some cases, this will increase efficiency drastically. Different pages within an application often share a lot of the same markup (navigation bar, footer, side bar, etc.) that only needs to be downloaded and rendered once. Consider a web page containing a written article and a photo gallery. A traditional MPA would download every single part of the page and reload the browser when the user wants to see the next photo. If this was implemented by an SPA framework, navigating to the next photo in the gallery only requires the photo to be downloaded. In order to increase the perceived performance, a technique called predictive loading could be implemented. If a user is looking through a gallery of photos, it is likely that he wants to see the next photo after viewing the current photo. By making this assumption, we can preload the next photo under the hood so that is instantly ready to be viewed when the user requests the next photo.

3.3.2 Implementation

The key features of the client application are:

- User authentication
- Floor plan with information about the number of people in each room
- A test page with a graphical representation of the estimated positions of beacons with respect to a base station.
- An administration page where users can list, edit and add base stations.
- A dashboard which lets users quickly determine whether a building is cleared or not.

Separation of concerns is obtained by structuring the application into the following modules

```
/
├── auth
├── beacons
├── core
├── floorplan
├── landing
├── layout
├── rooms
└── app.module.js
```

where `app.module.js` is the module that ties it all together. Each module can have controllers, directives, services, templates and configurations and may look like this:

```
beacons
├── directives
│   ├── beacon.html
│   └── beacon.directive.js
├── beacons.html
├── beacons.controller.js
├── beacons.module.js
├── beacons.service.js
└── config.route.js
```

One of the core modules is the authentication module. It restricts access to authorized users, which in this case would be system administrators and safety personnel. The system is designed with the intention that each building (or group of buildings) will have one user. When a user logs in, he will have access to all information that belongs to a specific building. The client receives beacon data from the server over a WebSocket [15] connection. The connection is implemented by a library called `socket.io`, which enables real-time bidirectional event driven communication. Socket.io is in principle a wrapper around the WebSocket API, but it also has fallbacks to other protocols to make it work on browsers that does not support WebSockets. Every time a beacon is updated on the server, a state object is transmitted to the client so that the updates can be reflected in the views.

One of the views is the beacon view. To benefit from the features provided by AngularJS, a directive was written to enable us to represent a beacon by an HTML element. An example is provided in Listing 3.4.

```
1 <beacon ng-repeat="beacon in beacons" x="beacon.x" y="
    beacon.y" name="beacon.id"></beacon>
```

Listing 3.4: Sample usage of the beacon directive.

This produces one beacon for each object in the list of beacons provided. The window size of the client is mapped to a specified domain, e.g., the directive can be configured to handle a maximum distance of 30 meters on the x-axis and 20 meters

on the y-axis. That means that if the window size is 800x600 pixels, and the beacon has the position (15, 19), it will be mapped to the position (600, 570) on the screen. Whenever the underlying beacon object gets a new position from the server, the beacon will animate from the current position to the new position to enhance user experience. The beacons are styled with CSS that adds animating expanding rings (ripples) to imitate the radio signals broadcasted by the beacons in order to enrich the user experience. An example of how beacons will look like in the browser is demonstrated in Figure 3.14.

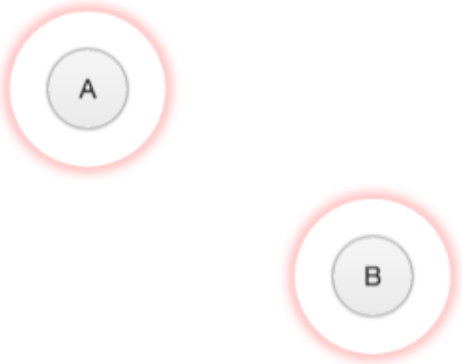


Figure 3.14: Graphical representation of beacons in the web application.

The part of the directive that watches for and acts upon position changes is presented in Listing 3.5. The scope variables `windowWidth` and `windowHeight` are updated whenever the window size of the browser changes so that the linear scales used to calculate the beacon position will update correspondingly. When the `tween` function is called, it will use a `TweenLite` [14] object to find the element that has the class `beacon` attached to it. It will then trigger an animation effect from whatever the current position is to the new position given by `x` and `y` over the course of 2 seconds.

```
1  scope.$watch('x', function (value) {
2    var xVal = d3.scale.linear()
3      // The domain of the x-axis is 0 to 30 meters.
4      .domain([0, 30])
5      .range([0, scope.windowWidth])(value);
6      tween(xVal, scope.y);
7  });
8
9  scope.$watch('y', function (value) {
10   var yVal = d3.scale.linear()
11     // The domain of the y-axis is 0 to 20 meters.
12     .domain([0, 20])
13     .range([0, scope.windowHeight])(value);
14     tween(scope.x, yVal);
15 });
16
17 function tween(x, y) {
```

```
18    var tl = new TimelineLite();
19    tl.add(TweenLite.to(element.find('.beacon'), 2, {
20      x: x,
21      y: y,
22      ease: 'easeOutExpo'
23    }));
24    tl.play();
25  }
```

Listing 3.5: This code serves the purpose of watching for updates on the beacon position and trigger an animation to visualize the movement.

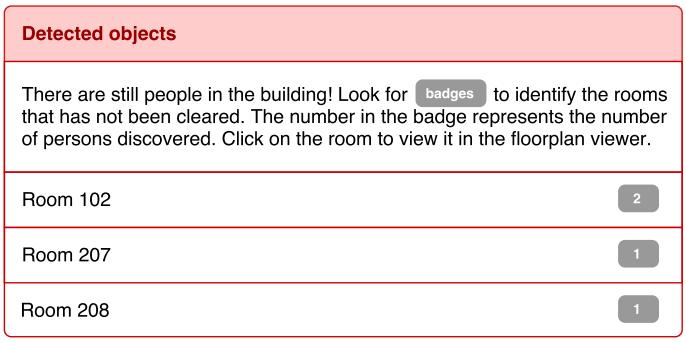


Figure 3.15: Dashboard view that lets users easily determine whether the building is empty or not.

The dashboard page of the application is just a simple list of the rooms in the building, and if the system has detected beacons in a room, a badge will appear next to the room name to indicate the number of people currently in the room (see Figure 3.15). The state is updated in real time using `socket.io`. The beacon data that is received from the server is structured in a key-value object where the IP address of the base station is the key, and the value is an array of beacons that is in range of the base station. The client fetches the list of base stations from the database and then uses the IP addresses to look up present beacons. These beacons are then attached to their corresponding room objects that are rendered in the list. The key parts of the room list implementation is listed in Listing 3.6 and 3.7.

```
1  <ul class="list-group">
2    <li class="list-group-item" ng-repeat="room in vm.rooms"
       >
3      {{room.name}}<span ng-show="room.beacons.length > 0"
         class="badge">{{room.beacons.length}}</span>
4    </li>
5  </ul>
```

Listing 3.6: The key part of the room list template.

```
1  // Fetch the list of rooms from the database and attach
     the
2  // array to the scope (or in this case ViewModel).
3  vm.rooms = roomService.getRoomsByUser(user.uid);
```

```
4
5  socket.on('beacon', function (presentBeacons) {
6    for (var i = 0; i < vm.rooms.length; i++) {
7      if (vm.rooms[i].address in presentBeacons) {
8        vm.rooms[i].beacons = presentBeacons[vm.rooms[i].
             address];
9      }
10    }
11 });
```

Listing 3.7: Part of the controller that receives updates from the server and updates the array of rooms that is rendered in the room list described in Listing 3.6.

The floor plan module provides a simple interface to present floors and rooms by using JSON. The properties of a room is then converted into CSS properties to give it a position within the building and also a length and width. The number of people detected in a room is displayed in a badge, just like in the room list view. The CSS is applied to the element by the help of a custom 'room' directive which can be used as a traditional HTML element like in Listing 3.8. The array `vm.rooms` is a description of the rooms in JSON format that is retrieved from the database and applied to the scope.

```
1 <div class="floorplan">
2   <evac-room ng-repeat="room in vm.rooms" room="room"></
       evac-room>
3 </div>
```

Listing 3.8: Usage of the `evac-room` directive.

3.4 Server

This system has several nodes that acts as servers. The base stations collects information about nearby beacons and communicate this information to the main server that handles the processing and client requests. The stack of tools used to build the servers are presented the list below:

- **Node.js** is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient. Node.js' package ecosystem, *npm*, is the largest ecosystem of open source libraries in the world.
- **Express.js** is a minimal and flexible Node.js web application framework that provides a robust set of features that enables rapid development of Node based web applications.

The components that form the server system consists of n base stations and a central processing server. The processing server can also act as a web server that hosts the web page. Alternatively, the web page can be hosted on a dedicated server.

Implementation details regarding the base stations and the processing server will be presented in the following sections.

3.4.1 Base Station

The base stations main responsibility is to listen for nearby beacons and convey information about these beacons to the processing server via a WebSocket link. The ability to listen for Bluetooth beacons is provided by a third party Node.js module [11]. This module lets the application listen for three types of events; *found*, *updated* and *lost*. The "updated" event is the most influential one because this will trigger every time a beacon broadcasts a data frame so that we can obtain information such as ID, TX power and RSSI.

```
1  EddystoneBeaconScanner.on('updated', function (beacon) {
2    if (beacon.id === calibrationNodeId) {
3      socket.emit('gamma-updated', {
4        ip: ip,
5        gamma: calculateGamma(beacon)
6      });
7    } else {
8      socket.emit('bs-updated', {
9        beacon: beacon,
10       ip: ip,
11       position: position
12     });
13   }
14 });
```

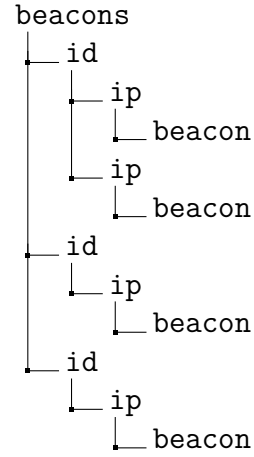
Listing 3.9: Example code of how a base stations can listen for beacons and notify the server when a beacon is updated. The `ip` and `position` are global variables that is transmitted along with the beacon object to let the server identify the base station.

In Listing 3.9, one can see how the updated event might be implemented. A beacon object which is a JSON representation of the Eddystone packet described in Figure 2.6 is given as a parameter to the callback function. In this example there is also a calibration node for the purpose of continuously calibrating the path loss exponent. This is located at a known distance from the base station so that all the factors required to calculate the path loss exponent are available. The formula to calculate the path loss exponent γ is derived from Equation 2.8 and is shown in Equation 3.13.

$$\gamma = \frac{L_{dB}}{10\log(d) + 20} \tag{3.13}$$

3.4.2 Processing Server

The processing server is the hub node that collects data from all the base stations and converts it into meaningful information before it is made available to the clients. The server uses `socket.io` (WebSocket) to communicate with both the base stations and the clients. Updates from the base stations are received from each node individually, while updates that are pushed to the clients are broadcasted to every connected client at simultaneously. When the server receives a beacon update from a base station, it stores the information in the following format



which is essentially a nested key-value store. It is of interest to find out which base stations can hear a specific beacon, and how good it can hear it. This data structure lets us look up beacons based on the beacon ID and the IP address of the base station that has it in range. Each time a base station is updated, this data object is injected into a processing algorithm that estimates the distances from the beacons to the base stations that are in range. The distance estimates is then filtered with a running average filter defined in Equation 4.1 before they are used to estimate the position of the beacon. The position is found by using the trilateration algorithm defined by Equation 3.12. Alternatively, the naive approach described in Section 3.2 is used, which results in a less accurate position estimate. The position estimates are broadcasted to all connected clients each time a beacon is updated. A flowchart that describes the interaction between the tiers is shown in Figure 3.16.

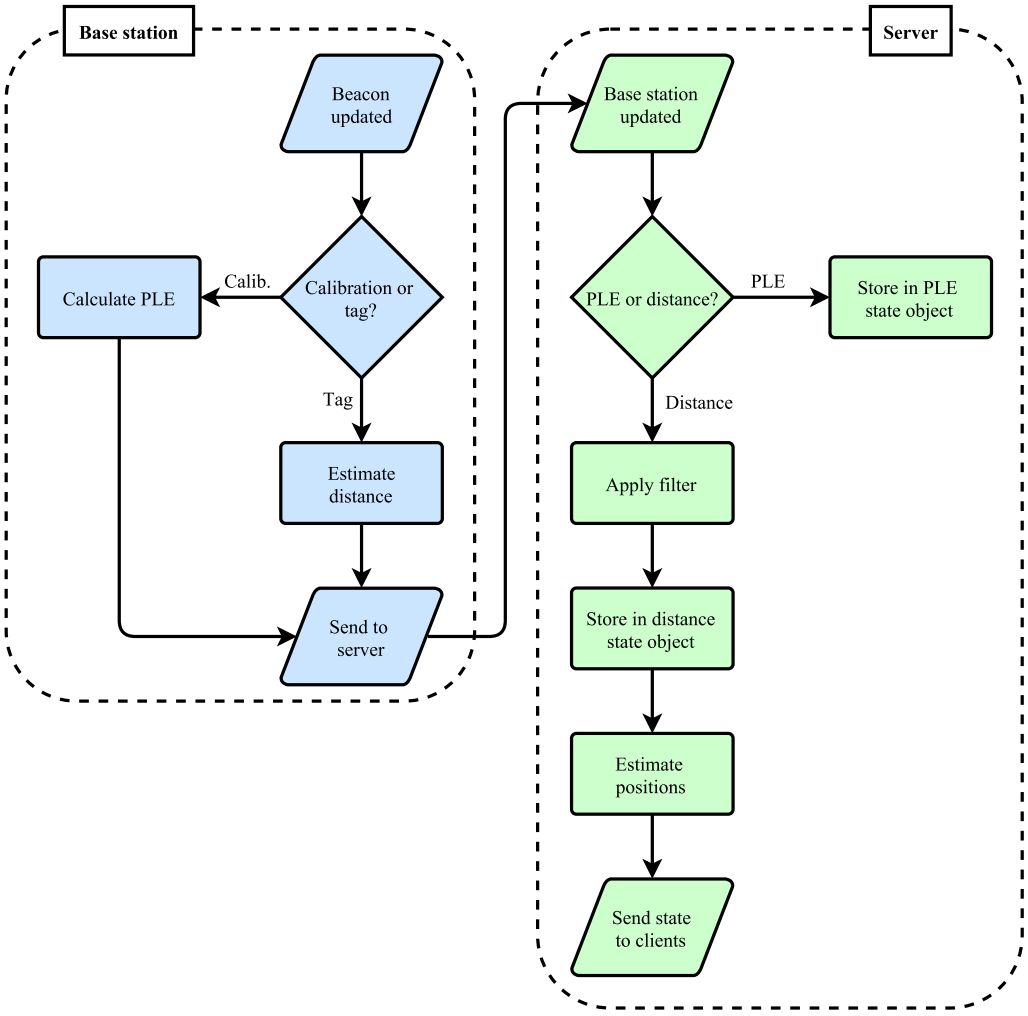


Figure 3.16: Simplified flowchart to describe the principal interaction between the system components.

4 Methodology, Results and Analysis

4.1 Methodology

The objective of the tests was to reveal how the beacons behave and more specifically how the signals are affected by attenuation, interference and objects. The testing was carried out in an indoor environment with walls and furniture. A sketch of the first floor in the testing environment is shown in Figure 4.1.

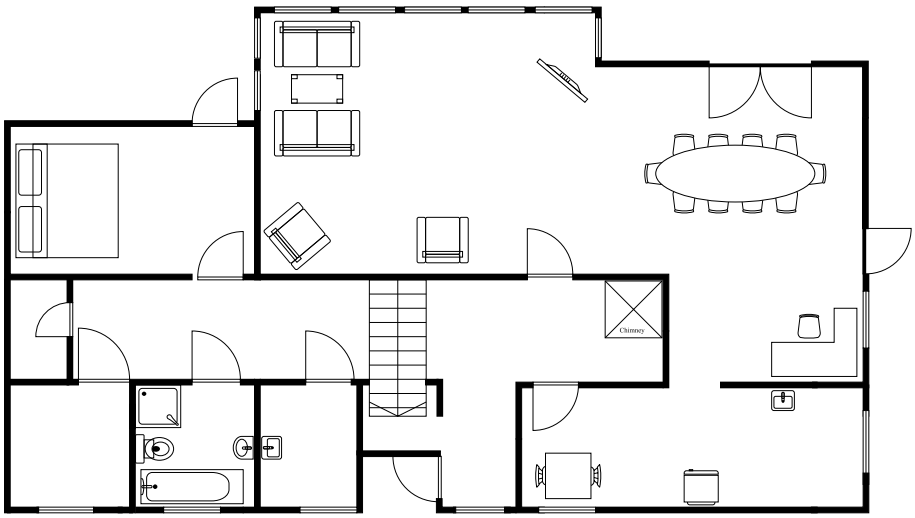


Figure 4.1: Floorplan of the first floor in the testing environment.

Raw data was collected and stored on the base stations and was processed by a Python script in post to extract the relevant information. The Estimote beacons was configured with maximum TX power (+4dBm) during all the tests (unless anything else is specified) and a broadcast frequency of 2 Hz. Most of the plotted data has been filtered with a moving average filter [36] due to severe fluctuations in the raw signals. If the signals are not filtered, it will be explicitly specified. The filter that was used, was implemented by convolution[1] according to the following

[1]Convolution is a mathematical operation where each value in the output is expressed as the sum of values in the input multiplied by a set of weighing coefficients.

equation:

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j] \tag{4.1}$$

Where x is the input signal, y is the output signal and M is the number of points to average over.

4.2 Results

To demonstrate the effect of filtering, raw data has been plotted together with the smoothed version of the same signal in Figure 4.2.

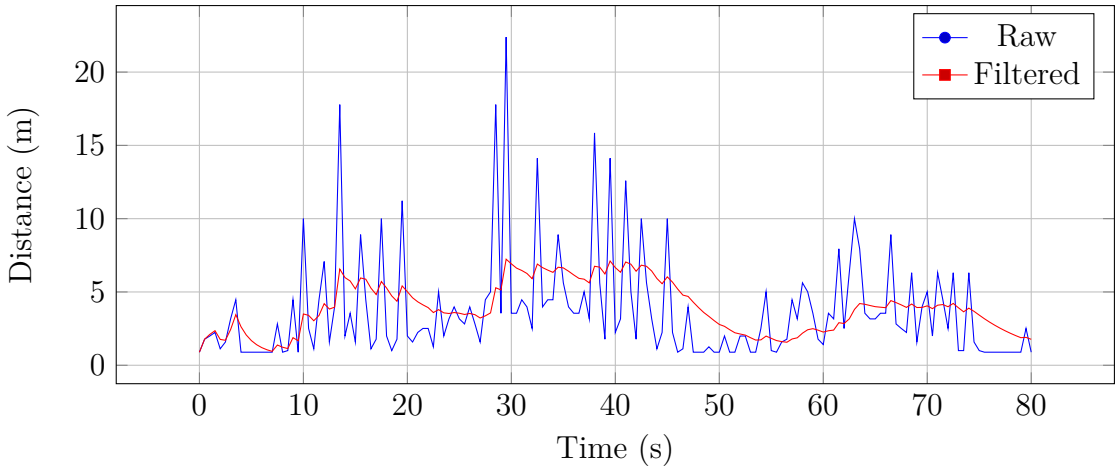


Figure 4.2: These plots demonstrates the effect of filtering raw distance values.

The next test was performed to see if the beacons behave consistently within the same brand. Three beacons was placed 180 cm from a base station for approximately 1 minute and 30 seconds. The RSSI values from each individual beacon is plotted in Figure 4.3. Based on these observations, it can be seen that the signals does follow a similar pattern and is not considered to be inconsistent. There is also a clear peak in the signals at the beginning and end of the plots. This phenomena was seen in most of the tests, and could indicate that the base stations need some time to interpret the signals correctly when a device is found or lost.

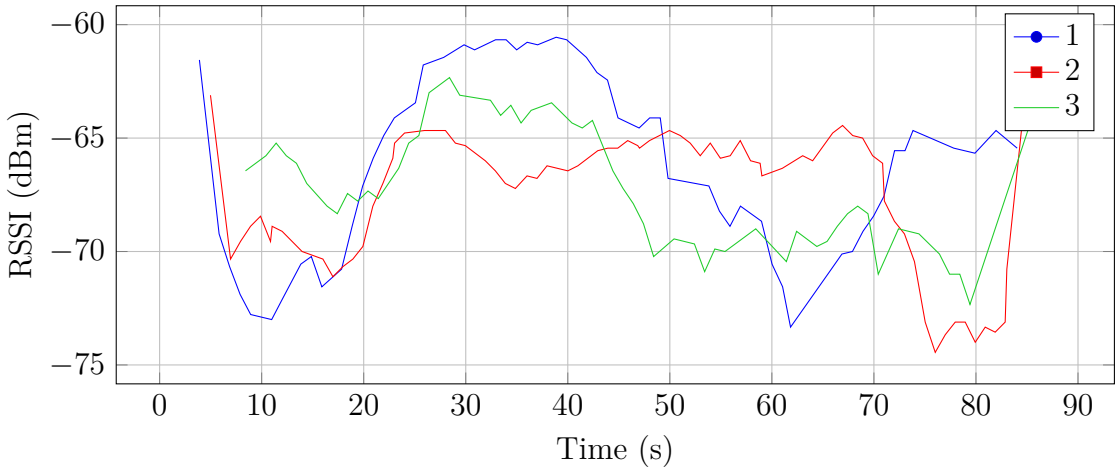


Figure 4.3: RSSI from 3 beacons at 180 cm. TX power is maximum (4dBm).

It was also interesting to see if there were variations among beacons with different hardware. This test was carried out with an Estimote beacon and a Huawei Nexus 6P both at 5 meters from the base station. In Figure 4.4 we can see that there is in fact a remarkable difference in the estimated distance between the two devices. The signal from the Nexus 6P is also smoother than from the Estimote. This will be discussed in more detail in Section 4.3.

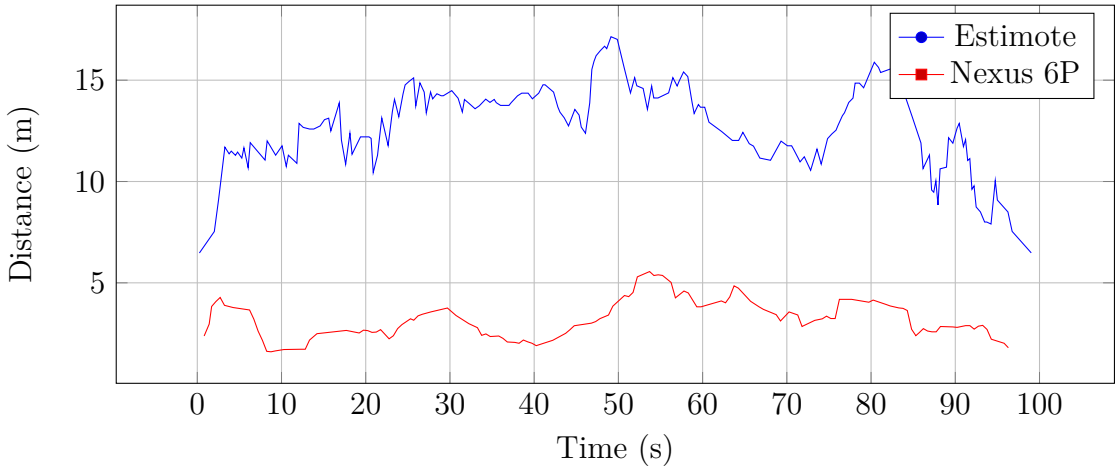


Figure 4.4: Estimote versus Nexus 6P at 5 m from the base station. Nexus TX power was -16 dBm and Estimote was -19 dBm.

Another important thing to test, was how well the signals travel through the human body. The results are plotted in Figure 4.5. It is evident that the human body absorb the 2.4 GHz signals quite well. Note the difference in number of measurements between the two plots. The reason why the red line has a lot fewer points than the blue line is because the signal was frequently lost, despite of the short distance (3 meters).

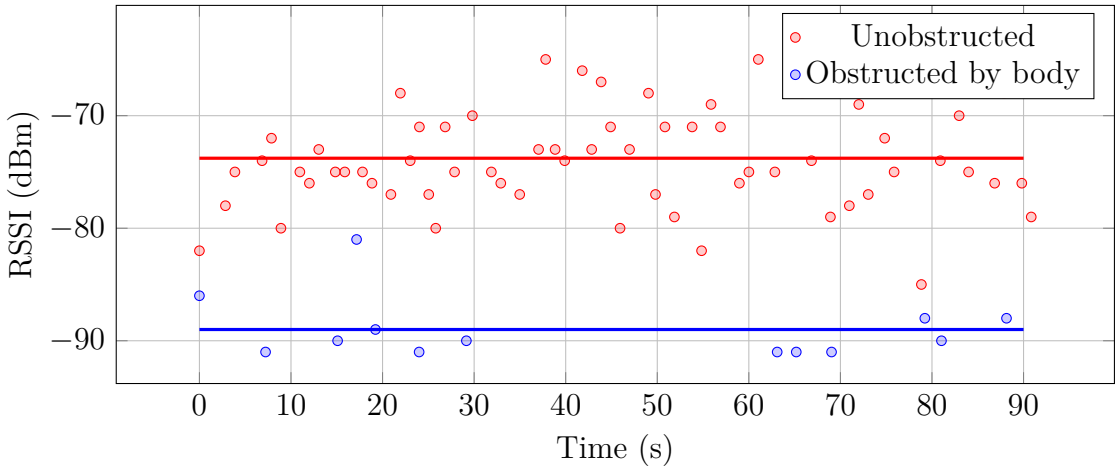


Figure 4.5: RSSI at 300 cm. TX power was maximum (4dBm).

The next test was performed to find out how much the signal is degraded when going through a door or a light wall. The experiment was carried out with a mobile station in one room, and a base station in the neighboring room. They were positioned in such a way that when the door was open, they were within each others line-of-sight. In Figure 4.6, the estimated distance between the beacon and the base station has been plotted. The first half of the measurements was taken with the door open, and the last half was taken with the door closed. As expected, the obstruction caused the signal strength to drop slightly and the estimated distances increased. The increase is so minor that it is considered to be negligible.

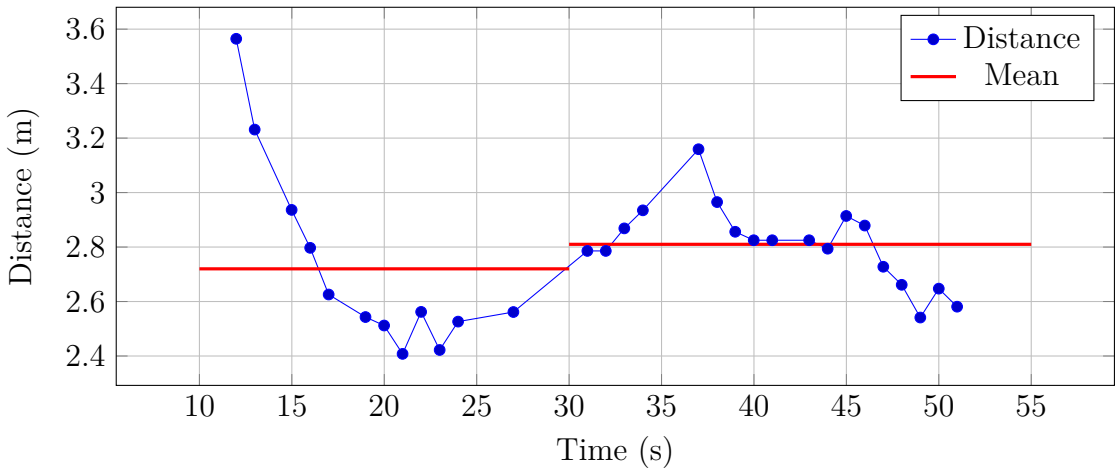


Figure 4.6: Estimated distance with open/closed door at approximately 3 meters.

The objective of the next test was to find out if the beacons interfere with each other. This was tested with a single beacon for one minute before introducing a second one, and after another minute, a third beacon was introduced. The results are visualized in Figure 4.7. Beacon one and two does not seem to interfere with each other at any time, but the signal from the third beacon appears stronger to the receiver.

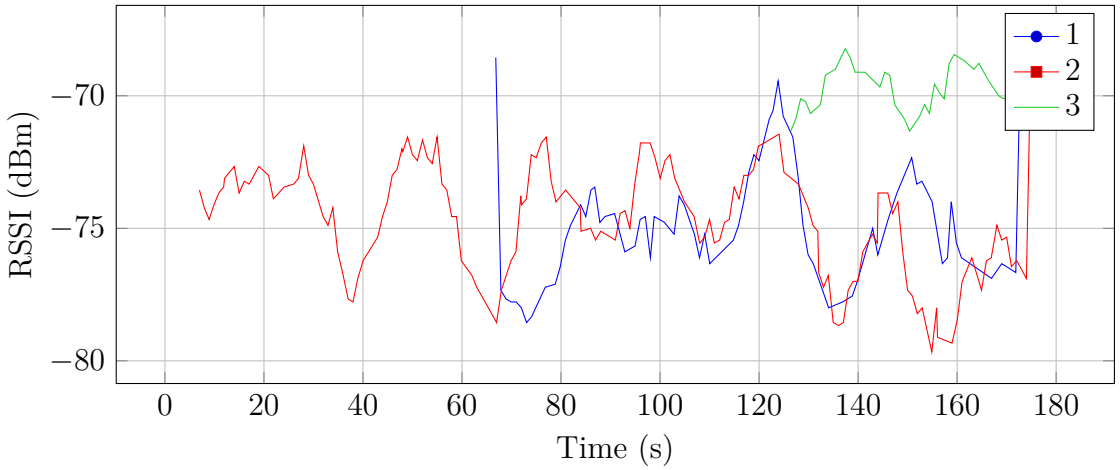


Figure 4.7: RSSI interference. TX power was maximum (4dBm).

It was also interesting to see if other sources of interference could affect the signal quality. Microwave ovens often operate at the same frequencies as WiFi and Bluetooth and could potentially degrade signal quality due to interference. Figure 4.8 shows the RSSI from a beacon measured at a distance of approximately 30 cm. The first 30 seconds was logged without any running microwave ovens nearby. After 30 seconds, an oven was turned on. The red lines illustrates the average RSSI in the first and second half of the testing period respectively. As we can see from the figure, there is a slight increase in the RSSI in the last part of the test session were the microwave oven was running. It could be random, but it is also plausible that noise from the oven could have an impact on what the base station hears.

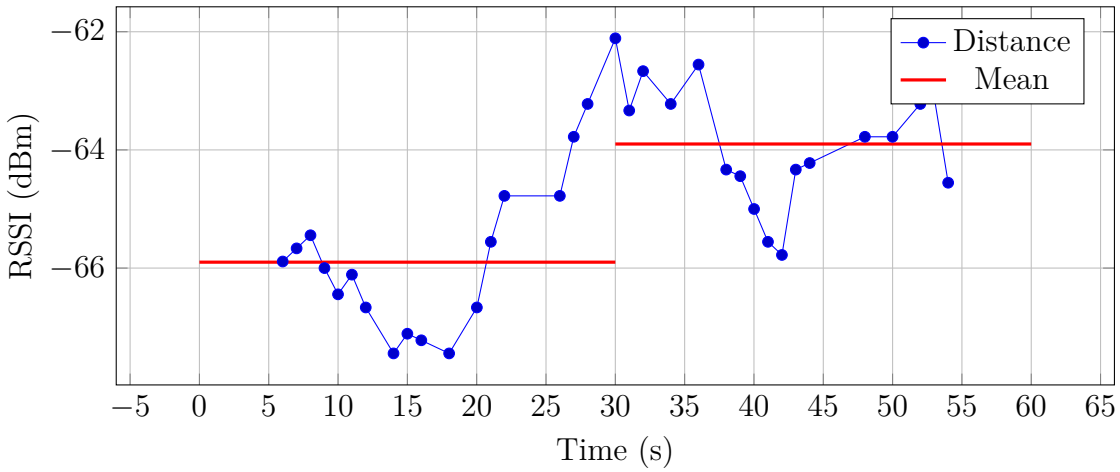


Figure 4.8: Microwave oven interference.

The next test was performed to see how well the distance estimates matches the actual distance at various ranges. In Figure 4.9, the estimated distance is plotted next to the actual distance. The session started with a beacon right next to a base station before it was moved back five meters with one meter steps staying one minute at each step. Note that when the beacon is in immediate proximity to the

base station, the estimated distance is constant, whereas when the beacon starts to move, the signal begins to fluctuate. Throughout the rest of the session, the estimates does follow the actual distance to a certain extent.

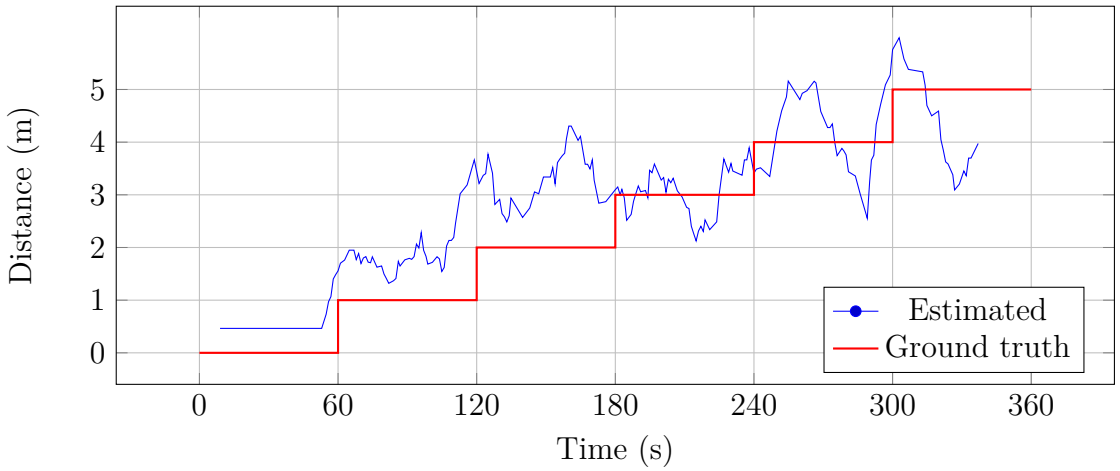
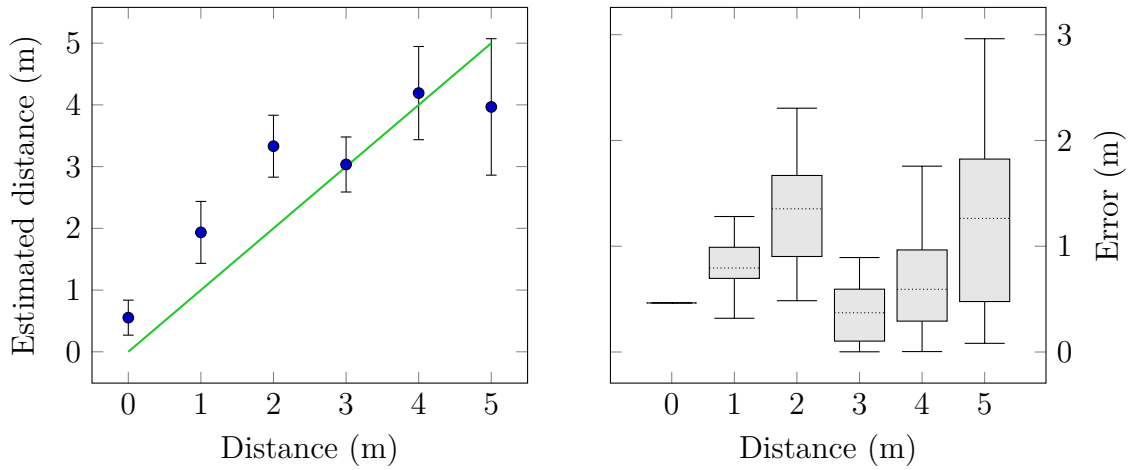


Figure 4.9: The beacon is moving from 0 to 5 meters, staying one minute at each step. The path loss exponent γ was set to 2.4 during the distance estimation.

In Figure 4.10a the mean and standard errors at the different steps has been plotted. The mean of the distance estimations at each step is represented with a dot, while the standard deviation is represented with the bars. The distribution of errors is represented in Figure 4.10b where the box defines the lower and upper quartile (Q_1 and Q_3) along with the median (dotted line inside the boxes). The upper whisker is defined by the maximum value that is $\leq Q_3 + 1.5 \times \text{IQR}$, where IRQ is the inter quartile range ($Q_3 - Q_1$). The lower whisker is defined as the minimum value that is $\geq Q_1 + 1.5 \times \text{IRQ}$. Note that outliers are not shown in this plot. As we saw earlier (Figure 4.9), the zeroth meter is estimated a little too high but very stable. The estimates continues to stay above the ground truth for the next two meters before they align almost perfectly with the the distance at 3 and 4 meters. The estimates at 5 meters starts to drop below the ground truth for the first time during the test. From the error distribution plot (Figure 4.10b) we can see that the errors in distance estimation is highest at 5 meters, which was expected. However, it was not expected to see such high error values at 1 and 2 meters, but the reason could be that these tests were carried out with a constant PLE. The median (dashed line) estimation error is below 1.5 meters for all the tested distances.



(a) Mean and standard deviations.

(b) Distance estimation error.

Figure 4.10: The plots above describes the variability of errors in distance estimation with regards to the actual distance.

Figure 4.11 demonstrates the accuracy of the position estimates performed by the system. The plot has been created by logging position estimates calculated from the distance to three base stations and comparing them to the actual position to get the error. The Cumulative Distribution Function (CDF) of errors in trilateration with adaptive path loss exponent (discussed in Section 3.2.1) is plotted in orange. For comparison, a test with constant path loss exponent of 2.2 (found by trial and error) was also carried out. The results of this test is plotted in green. An improvement of approximately 1 meter in accuracy was seen after the adaptive path loss exponent was introduced. We can see that for the adaptive PLE, more than 70% of the estimates has less than 2 meters of error, while for the constant PLE the error can be up to ~3 meters for the same 70% of estimations.

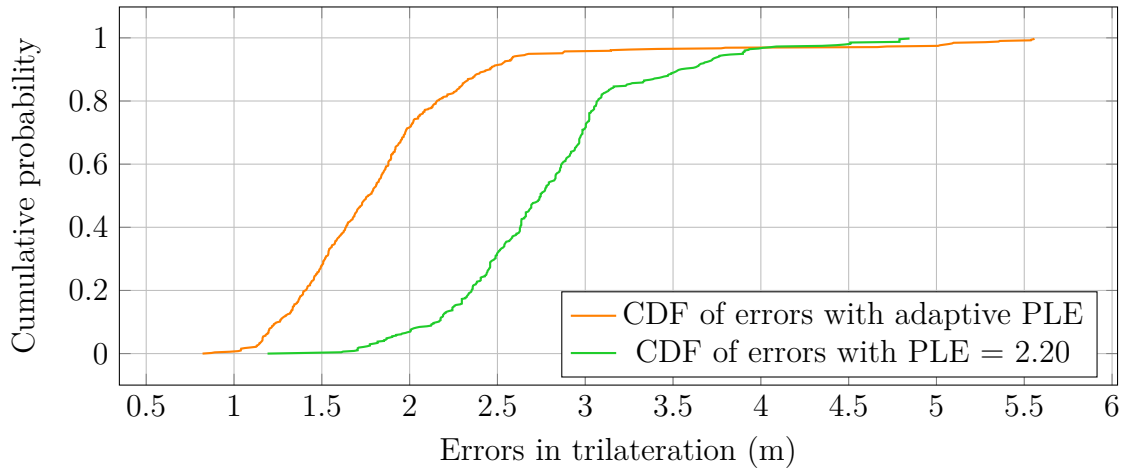


Figure 4.11: Empirical cumulative distribution function of errors in trilateration with adaptive and constant path loss exponents (PLE).

4.3 Analysis

In the following sections we will discuss the results from the experiments. We will try to explain the reason for the outcomes and discuss their significance for the overall system performance.

4.3.1 RSSI Fluctuations

The most apparent phenomena that was seen in the tests, is that the signal strength fluctuates heavily. One of the reasons for this can be that the beacons advertises on three channels with significantly different center frequencies (see Section 2.3.1). It should be emphasized that the advertising channels does not benefit from the frequency hopping scheme that is specified for the data channels. External noise and multipath fading can influence how the signals appear to the receiver. From Figure 3.4 we see that only a few decibels of noise would impact the distance estimates largely. It is also likely that signal characteristics is hardware dependent an varies between different chipsets. The RSSI is a value that is reported by the chipset at the receiving end of the link, therefore, the radio circuits and antennas used in the Bluetooth module on the base stations will affect the RSSI. Since this is a system that will be deployed in existing buildings, we cannot require any drastic changes in the environment, hence we cannot assume an optimal propagation model for the radio signals. The only noise affecting factor that we are able to control is the chipset used on the transmitters (beacons) and receivers (base stations), and also the sampling algorithms used to smooth the RSSI values.

4.3.2 Distance Estimation Error

Although the RSSI is fluctuating a lot, it is possible to see a realistic trend between the RSSI and distance, especially after smoothing the samples. In Figure 4.9, 4.10a and 4.10b, we can see that the estimated distances follows the actual distance to a certain extent. It starts off by being a little higher in the first period, and towards the end, it is a bit below the actual distance. Even if it is not correct, the step pattern is definitely visible. By adjusting the parameters in the distance calculation, it could be possible to obtain a closer fit, but keep in mind that these adjustments could lead to more errors in other scenarios (known as *overtraining*).

4.3.3 Hardware Differences

From Figure 4.3 we can see that there is a consistent performance among the three beacons from Estimote, but if we introduce a beacon from another vendor (see Figure 4.4), the variations becomes apparent. The beacon used in the latter experiment is not a true beacon, but a virtual beacon enabled by the Estimote application for Android. The hardware is a Google Nexus 6P smartphone that supports BLE.

It was not possible to see or adjust the TX power of the virtual beacon, but the received frames stated that the TX power was -16 dBm. The closest available TX power on the Estimote beacons was -19 dBm. The TX power is taken into account when estimating the distances, so the difference should not influence the result in any way. The curves generated by samples from the Nexus 6P and the Estimote beacons both follow the same pattern, but there is a noticeable difference in noise between the two. The Nexus 6P produces a lot cleaner signal than the Estimote beacon does. There is also a gap of ± 5 meters between the two lines, which was not expected. It is possible that there is a bug in the Estimote app that incorrectly sets the TX power property of the frame to -16 dBm when it is in fact something else. Another factor that can explain the gap, is that the chipset on the Nexus 6P produces a cleaner signal which in turn increases the SNR so that the reiceived signal appears to be stronger.

4.3.4 Absorptive Losses

Based on the results in Figure 4.5, it is clear that a substantial amount of the signal strength is absorbed by the human body. Not only is the RSSI significantly lower than for free space, but a significant amount of packet losses was seen in the test. The receiver was not able to recover more than ~20% of the packets that was transmitted through the body. The experiment also demonstrates a ~15 dB reduction in RSSI which will have significant effect on the distance estimation. This was expected, and there is not much we can do about it, without changing to another protocol with a lower carrier frequency. This could potentially be a drawback in crowded buildings or if the beacon is placed on the body in such a way that the signal has to travel through the body to reach a base stations. This has to be thought about when deploying the system. To lower the effect, the beacon should be placed in the upper part of the body, the base stations on the middle (or higher) of the wall and there should be a sufficient amount of base stations to cover a person even if one or more is obstructed by the body. Testing has also proven that concrete degrades the signal quality substantially compared to thinner wood paneled walls. The testing environment has a concrete block chimney that lead to largely reduced range when the sender and receiver was positioned in such a way that the chimney was in between the two. It is difficult to tell whether or not the received signal actually traveled through the concrete, or of it was captured from the multipath components. In Figure 4.6 we can see that the difference in RSSI with line-of-sight and through a door, is minimal.

4.3.5 Interference

Interference is bad when it comes to communication, but it is almost inevitable. There can exist multiple sources of interference in the 2.4 GHz band in a typical indoor environment, e.g. WiFi, microwave ovens, wireless phones etc. Two of the tests in this experiment was specifically targeted against interference. In the test

were the objective was to find out if the beacons interfere with each other (see Figure 4.7), it was discovered that there is no clear sign of interference. The signal characteristics does not change significantly when introducing more beacons to the network, and there is no packet drops. The microwave oven experiment (see Figure 4.8) demonstrates that the RSSI can in fact be affected by interference. The test results indicates that the signals generated by the microwave oven can fool the receiver to interpret a higher signal strength. A theory is that the noise generated by the microwave oven and the signals from the beacon both contribute to the RSSI. The authors of [33] has done a research to understand the effect of microwave ovens on Bluetooth networks, and they experienced that the interference were only significant at short distances (less than 1 m). When moving further away (around 10 m) from the oven, the noise did not impair operation or usability of the system. In [26] the influence of microwave ovens on IEEE 802.11 (WiFi) was discussed. The general conclusion was that microwave ovens only interfere with devices in close proximity. These studies focused on data transfer and how the interference affected throughput, but the effect is limited due to frequency hopping. Recall from Section 2.3.1 that frequency hopping is not used for advertisements, hence the microwave oven could interfere with the beacons depending on the frequency of the noise it generates. This issue could potentially threaten the Quality of Service (QoS) [22] , and it should be noted that other similar sources of interference has not been studied explicitly.

4.3.6 Adaptive PLE

Recall from Section 3.2.1 that we introduced a mechanism for adaptively choosing the PLE to address the issues related to a constant PLE. As we saw in Figure 4.11, the effect of this mechanism was positive, but we will discuss some thoughts about it in this section. To get the best effect, the mobile station should be close to a reference beacon to have propagation channels similar to those of the reference beacon. This is even more important in densely obstructed areas where a little change in position means a totally different propagation model. When trying to measure the distance between two points which are not in line-of-sight, one estimates the length of the propagation path, which can be quite different from the actual distance between the two points.

Another thing worth mentioning is that the reference beacons introduces extra overhead in the deployment phase, however, it increases the versatility of the system even more compared to other methods. As described in Section 3.2, one of the drawbacks of fingerprinting is that any changes to the environment may corrupt the fingerprint which requires an update of the database. This means that the adaptive PLE makes the system more versatile because it automatically adapts to environmental changes. The low cost of BLE transmitters allows for many calibration beacons to be deployed, hence the accuracy of the two methods should be comparable.

4.3.7 Summary of Test Results

A brief summary of the testing results are listed below:

- The signals fluctuates a lot and has to be filtered to stabilize the results.
- Although the signals are very unstable, it is possible to see a realistic pattern between RSSI and distance.
- The beacons perform consistently within the same brand, but signal characteristics can vary a lot for different types of beacons.
- The signals are easily absorbed by the human body and concrete walls, but lighter doors and walls have minor impact.
- The beacons does not interfere with each other, but the signal quality can degrade due to external interference such as micro wave ovens.
- The proposed method to continuously calibrate the path loss exponent increased the accuracy of the system.

4.3.8 Deployment Options

There is a number of factors that needs to be reflected upon before deploying the proposed system. Accuracy is important in every positioning system, but the higher accuracy, the more sacrifices has to me made when it comes to simplicity in implementation, deployment and maintenance. Examples of configurable parameters are base station density, calibration beacon density, beacon broadcast interval, tracking algorithm complexity, etc.

Higher density of base stations and calibration beacons could give more accurate position estimates but result in more overhead in the deployment and maintenance phases. It would also increase the total cost of the system. A lower broadcast interval would allow for a higher window size in the smoothing filter in order to reduce the fluctuation effect, but it would decrease battery life of the beacons, thus increasing maintenance overhead. Other tracking algorithms that are based on an offline mapping phase [25] could give more accurate position estimates, but the tight coupling to the environment of which it is installed can be negative. Another option is to use the naive proximity based approach mentioned in Section 3.2 where we (based on the base station density) could define zones. The size of a zone could be e.g. one room or span over multiple rooms. If such an uncertainty in the estimates is acceptable, the system could be simplified and very easy to implement and install. A system designer should balance these parameters in order to deliver a satisfactory accuracy.

This page is intentionally left blank.

5 Conclusions

5.1 Summary

This thesis provides a viable and practical solution to the challenge of tracking people inside a building during an emergency situation. The proposed system can be a helpful tool to increase the chance of saving lives in search-and-rescue operations. The approach uses off-the-shelf hardware to reduce the cost and increase availability. It is designed in such a way that it can be used in environments where mobile phones are not allowed, e.g., offshore oil platforms. Custom tracking software was tested on a real system and proved to give an acceptable accuracy where more than 70% of the position estimates had less than 2 meters of error. This was obtained by implementing a technique to continuously calibrate the path loss exponent for the given propagation environment. The real time web client worked well as a front end to the positioning system and was able to quickly provide first responders with important information about the state of evacuation. The 2.4 GHz signals poor ability to penetrate the human body makes the placement of beacons and base stations critical. Radio characteristics is not consistent among various hardware and other chips may perform better or worse than the tested nRF51822. Accuracy depends on a number of factors that should be evaluated and balanced by a system designer before deploying the solution.

5.2 Significance

A manual search operation would put rescuers and victims at unnecessary high risks. By using the proposed system, rescuers can save time, which is critical in any emergency response operation. The faster a building is evacuated, the higher the chance of avoiding danger and saving lives.

5.3 Future Work

The following sections discuss directions for future work.

5.3.1 Experiment with Other Methods

While this thesis has researched and implemented the basic framework for an indoor positioning system using BLE beacons, more positioning methods needs to be tested in the future. The best alternative candidate is probably the fingerprinting method described in Section 3.2.

The proposed system involves simple filtering of RSSI values to smooth the heavy fluctuations in the raw signal strengths. However, more advanced tracking algorithms, such as the Kalman filter [30, 37], could be used to predict the movement of people inside the building to increase accuracy. It can be argued that tracking techniques is not as relevant in this system compared to traditional navigation systems because we make the assumption that people who are left inside the building is probably not moving. Even if the people were moving, we would have to develop a mathematical model to describe the movement in order to make accurate predictions. Various models are described and tested in [38] and we can see that the behaviors are very different depending on the conditions. It is especially hard to predict the movement of people that are in panic. A comparison with and without a sophisticated tracking algorithm could be performed in future work to learn about the effects.

The trilateration algorithm used in this system only considers two dimensions. Going from 2D to 3D is not a lot of work now that we have established the set of equations (see Equation 3.12) for trilateration. For the system to be ready for deployment, a third dimension should be implemented so that we are able to distinguish different floors. The third dimension has already been implemented in code, but it has yet to be tested. At least 4 base stations are required for 3D positioning, so in order to test it, another base stations has to be acquired. Another way to achieve this would be to reduce the TX power of the beacons to ensure that they will never reach a base station on a different floor than they are currently located.

5.3.2 Examine Scalability

To keep the prototype simple and to meet the budget of the project, only three base stations and three mobile stations were used. The scalability of the system needs to be more thoroughly investigated in the future. This testing should involve more beacons (or a higher density of beacons) and a larger building with more base stations. Due to the findings in the test where the signals was obstructed by the human body (demonstrated in Figure 4.5), it would be interesting to see the performance in a crowded building.

5.3.3 Practical Comparisons of Alternative Systems

Theoretical comparisons of existing positioning systems using WiFi may not be accurate, therefore, a practical comparison between the proposed system and an

equivalent system using WiFi can be performed to understand the differences in performance. It would also be interesting to develop a system that combines the two technologies (BLE and WiFi) in an attempt to increase robustness and accuracy. The two technologies could be used simultaneously or they could be divided into a primary and secondary system were the secondary system is used as a fallback in case of a failure in the primary system.

5.3.4 Experiment with Other Hardware

The discoveries made in the test plotted in Figure 4.4 tells us that differences related to transmission and reception circuitry as well as the antennas used, needs to be more extensively researched. Although we have tested the system with two kinds of beacons, the same receiver module was used in all the tests. More professional equipment with "rubber ducky" antennas in both ends of the link is expected to increase the signal strength with approximately 20 dBm according to [19]. That implies a better range and a lower density of base stations would be required.

5.3.5 Implement the Inverted Design Option

Finally, design option 1 that was described in Section 3.1 has not been implemented at this stage. This could completely replace alternative 2, but it is also possible for the two systems to coexist. The cost of alternative 1 would presumably be lower than of alternative 2 considering that most people are equipped with a mobile phone, but remember that alternative 2 has the advantage that it is applicable in situations were mobile phones are not allowed. A major advangtage of alternative 1 however, is that beacons are a lot easier to deploy than the base stations proposed in Section 3.1. Bluetooth beacons are inexpensive, so dense deployments are feasible.

This page is intentionally left blank.

Bibliography

[1] Beacons. https://developers.google.com/beacons/.

[2] Bluetooth. https://www.bluetooth.com/.

[3] Differences Between iBeacon and Eddystone. https://bkon.com/beacons/differences/.

[4] Eddystone. http://developer.estimote.com/eddystone/.

[5] Eddystone protocol specification. https://github.com/google/eddystone.

[6] iBeacon. http://developer.estimote.com/ibeacon/.

[7] iBeacon vs Eddystone: Which One Works Better for Your Pilot Project? http://blog.beaconstac.com/2016/01/ibeacon-vs-eddystone/.

[8] Internet of Cows. http://data-magnum.com/can-a-cow-be-an-iot-platform/.

[9] Model-View-Controller (MVC). http://www.w3schools.com/aspnet/mvc_intro.asp.

[10] Model-View-ViewModel (MVVM). https://msdn.microsoft.com/en-us/library/hh848246.aspx.

[11] Node.js Eddystone beacon scanner. https://github.com/sandeepmistry/node-eddystone-beacon-scanner.

[12] The Physical Web. https://google.github.io/physical-web/.

[13] Raspberry Pi Zero. https://www.raspberrypi.org/blog/raspberry-pi-zero/.

[14] TweenLite. https://greensock.com/tweenlite.

[15] WebSockets. http://www.websocket.org/index.html.

[16] Jose Carlos Segura Carlos Medina and Angel De la Torre. Ultrasound Indoor Positioning System Based on a Low-Power Wireless Sensor Network Providing Sub-Centimeter Accuracy. In *Sensors*, pages 3501–3526. Basel, Switzerland : MDPI, c2000-, 2013.

[17] Erdal Cayirci and Chunming Rong. *Security in Wireless Ad Hoc and Sensor Networks*. John Wiley & Sons Ltd., 2009.

[18] H.S.M. Coxeter. *Introduction to Geometry*. Wiley classics library. Wiley, 1969.

[19] Charles Gervasi. Bluetooth Low Energy. https://www.element14.com/community/groups/wireless/blog/2013/08/23/bluetooth-low-energy, August 2013.

[20] Robin Heydon and Nick Hunn. Bluetooth Low Energy.

[21] Joseph Hoy. *Forensic Radio Survey Techniques for Cell Site Analysis*. John Wiley & Sons, December 2014.

[22] ITU-T. Definitions of Terms Related to Quality of Service. E.800, September 2008.

[23] Richard L. Branham Jr. *Scientific Data Analysis: An Introduction to Overdetermined Systems*. Springer-Verlag New York Inc., 1st edition, 1990. p. 86.

[24] Erik K. Beacon tracking with Node.js and Raspberry Pi. https://medium.com/truth-labs/beacon-tracking-with-node-js-and-raspberry-pi-794afa880318#.fimhtzngq, December 2014.

[25] Kamol Kaemarungsi and Prashant Krishnamurthy. Modeling of Indoor Positioning Systems Based on Location Fingerprinting. Technical report, School of Information Science, University of Pittsburgh, 2004.

[26] Ad Kamerman and Nedim Erkoçevic. Microwave Oven Interference on Wireless LANs Operating in the 2.4 GHz ISM Band. In *The 8th IEEE International Symposium*, volume 3. October 1997.

[27] Peng Lin, Qingbin Li, Qixiang Fan, Xiangyou Gao, Senying Hu, Peng Lin, Qingbin Li, Qixiang Fan, Xiangyou Gao, and Senying Hu. A Real-Time Location-Based Services System Using WiFi Fingerprinting Algorithm for Safety Risk Assessment of Workers in Tunnels. *Mathematical Problems in Engineering, Mathematical Problems in Engineering*, 2014, 2014:e371456, April 2014.

[28] Andreas Loeffler and Heinz Gerhaeuser. Localizing with Passive UHF RFID Tags Using Wideband Signals. In M. I. B. Reaz, editor, *Radio Frequency Identification from System to Applications*. InTech, June 2013.

[29] W.S. Murphy and W.A. Hereman. *Determination of a Position Using Approximation Distances and Trilateration*. 1992.

[30] David Obdrzalek and Achim Gottscheber. *Research and Education in Robotics - EUROBOT 2011: International Conference, Prague, Czech Republic, June 15-17, 2011. Proceedings*. Springer Science & Business Media, June 2011.

[31] James Manyika Richard Dobbs and Jonathan Woetzel. The Internet of Things: Mapping the Value Beyond the Hype, June 2015.

[32] Andrzej Banaszuk Robert Tomastik, Satish Narayanan and Sean Meyn. Model-Based Real-Time Estimation of Building Occupancy During Emergency Egress. In *Pedestrian and Evacuation Dynamics 2008*, pages

215–224. Springer-Verlag Berlin Heidelberg 2010, 1 edition, 2008.

[33] T. W. Rondeau, M. F. D'Souza, and D. G. Sweeney. Residential Microwave Oven Interference on Bluetooth Data Performance. *IEEE Transactions on Consumer Electronics*, 50(3):856–863, August 2004.

[34] Ruben M. Lorenzo Patricia Fernandez Francisco A. Lago Eduardo Garcia Juan Blas Santiago Mazuelas, Alfonso Bahillo and Evaristo J. Abril. Robust Indoor Positioning Provided by Real-Time RSSI Values in Unmodified WLAN Networks. In *IEEE Journal of Selected Topics in Signal Processing*, volume 3, pages 821,831. 2009.

[35] Bluetooth SIG. Supplement to the Bluetooth Core Specification, 2014.

[36] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing San Diego, California, 2 edition.

[37] Morten Tengesdal. *Kalman-filteret: Prosessinnsyn i Praksis*. 2007.

[38] Jaroslaw Was. Experiments on Evacuation Dynamics for Different Classes of Situations. In *Pedestrian and Evacuation Dynamics 2008*, pages 225–232. Springer-Verlag Berlin Heidelberg 2010, 1 edition, 2008.

[39] Martin Werner. *Indoor Location-Based Services: Prerequisites and Foundations*. Springer International Publishing Switzerland, 1 edition, 2014.

[40] Andrew Markham Niki Trigoni Yong Ren Xiaojie Zhao, Zhuoling Xiao. Does BTLE Measure Up Against WiFi? a Comparison of Indoor Location Performance. In *European Wireless 2014*, pages 263,268. VDE VERLAG GMBH, Berlin, Offenbach, Germany, 2014.

This page is intentionally left blank.

A Source Code

 The source code and log files are embedded in the PDF document as a compressed file named `evac.7z`.