University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

BACHELOR'S THESIS

Study programme/specialisation: Computer Science, Electrical Engineering	Spring semester, 2021 Open / ~~Confidential~~
Authors: Nikolas Cameron Grimstad, Simen Graue Kase, Stephan Frederik Werner Brandasu	
Programme coordinator: Naeem Khademi Supervisor(s): Naeem Khademi	
Title of Bachelor's Thesis: An Indoor Positioning System for IDE using Bluetooth Low-Energy Beacons	
Credits: 20	
Keywords: Indoor positioning, wireless communication, Bluetooth Low Energy	Pages: 37 + attachments Stavanger, 15^{th} of May / 2021 date/year

An Indoor Positioning System for IDE using Bluetooth Low-Energy Beacons

by

Nikolas Cameron Grimstad
Simen Graue Kase
Stephan Frederik Werner Brandasu

Supervisor: Naeem Khademi

A thesis submitted in partial fulfillment for the bachelor's degree of Computer Science and Electrical Engineering

at the

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

May 2021

Abstract

This thesis provides a solution for the challenge of accurate indoor positioning. Using both commercial Bluetooth Low Energy (BLE) beacons and our own Do It Yourself (DIY) beacons, using ESP32's, we were able to compare these solutions in different environments.

A Two-dimensional (2D) Indoor Positioning System (IPS) was setup in the hallway of the Kjølv Egeland building at the University of Stavanger.

Test were run both before and after implementation of the IPS, and even though we were only able to run a limited amount of tests, the results show that an IPS could be a viable solution.

Preface

This bachelor thesis was written at the Department of Electrical Engineering and Computer Science at the University of Stavanger.

We would like to give a special thanks to our supervisor Naeem Khademi, an Associate Professor at UiS, who has given us valuable guidance and contributed with insightful and nuanced perspectives.

We would also like to thank Ståle Freyer for teaching us how to solder and for helping us with our beacon hardware orders.

Thanks to Statsbygg, to Operations Manager Jan Magnus Kringstad, for giving us the floor plans for UiS.

Also thanks to Sorina Sarac for loaning us her Huawei Mate 10 Lite, Saskia Brandasu for loaning us her Samsung Galaxy S8, and to Ingunn Graue Kase for helping with proofreading our thesis.

Finally we would like to thank UiS, for offered learning resources and for buying the beacon hardware for us.

Contents

List of Figures

Acronyms

2D Two-dimensional

AoA Angle of Arrival

AoD Angle of Departure

APs Access Points

BLE Bluetooth Low Energy

dBm Decibel-milliwatt

DIY Do It Yourself

EID Ephemeral Identifier

GPS Global Positioning System

IPS Indoor Positioning System

KE Kjølv Egeland

RSSI Received Signal Strength Indication

RX Receive Level

SoC System on a Chip

TDoA Time Difference of Arrival

TLM Telemetry

ToA Time of Arrival

ToF Time-of-flight

UID Unique Identifier

URL Uniform Resource Locator

UUID Universally Unique Identifier

UWB Ultra-Wideband

1 Introduction

1.1 Problem

The question that will be be explored through this thesis comes in 2 parts; Is it viable to make your own DIY beacon instead of buying a commercial ready made solution, and is the improvement of setting up an IPS worth it?

Map applications used for finding your way around places have become very common, but Global Positioning System (GPS) struggles a lot when you move indoors. This makes navigating using these map applications very difficult in large and complex buildings. The solution to this problem is a technology dedicated for indoor use referred to as IPS. But there are many different methods of implementing IPS. In this thesis we will focus on implementing IPS through the use of BLE beacons. This is a small device that broadcasts a Bluetooth signal at a set power and a set interval. This can be used to help improve location accuracy indoors.

1.2 Objective

To document the process of choosing the type of Bluetooth beacon, whether it be commercial or a DIY solution based on factors such as convenience and price. Document the process of setting up said beacons, and to see how well each solution works in specific scenarios. This information can aid places like Universities, Library's, stadiums or tunnels to set up a beacon network to enable indoor positioning at their venue.

1.3 Approach

The system consists of a beacon, a mobile phone, and a IPS service. The beacons will be our DIY beacons and some commercial beacons. These beacons will be installed and then an IPS will be set up to take advantage of them. The mobile phones will look for beacons, and with the help of a specific app will know what to do with them.

1.4 Thesis Outline

Chapter 2 - Literature Review

Literature review of key concept, beacons and related works.

Chapter 3 - Methodology

Implementations and details of our system.

Chapter 4 - Results & Discussion

The results and analysis of significance of the results.

Chapter 5 - Conclusions - & Recommendations

Summary of our project, our general thoughts and recommendations for future works.

Appendix A - Source Code

The software developed during this thesis.

2 Literature Review

2.1 Methods of detecting position in 2D space

There are many different methods of trying to detect the position of a device in 2D space, in this sub chapter we will explore some of these.

2.1.1 Trilateration

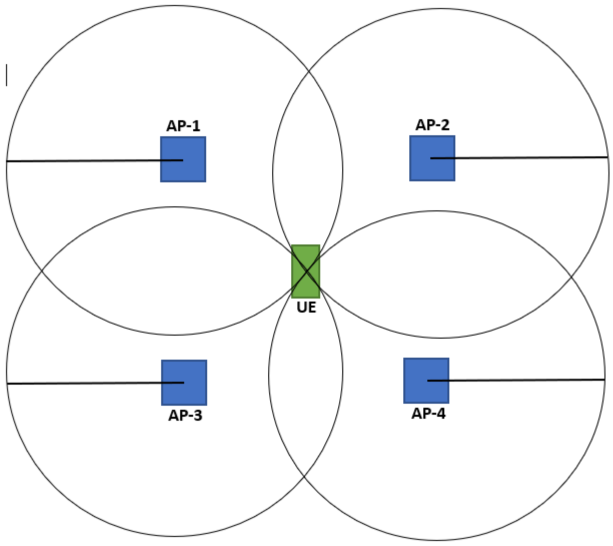


Figure 2.1: Trilateration method

Trilateration is a common method for position calculations. Trilateration uses three or more fixed points in 2D space to calculate the position of an object, by finding the intersection of a series of circles, as shown in figure 2.1. The distance between the Access Points (APs) can be found by using either RSSI, Time of Arrival (ToA) or Time Difference of Arrival (TDoA). The distance estimated by signal strength is presented as a circle with a radius around the APs. The intersection of three AP radius's provide a point or an area of the user equipment [31].

2.1.2 ToA & TDoA

ToA and TDoA measure distance to a stationary by measuring the signal loop time. This means that a stationary device sends a signal to a mobile device. The mobile device returns the signal to the stationary device, and the stationary device will calculate the full round travel time to determine how far away the mobile device is. The formula for ToA calculation is:

$$d = c * (t_{arrival} - t_{sent})$$

Where c refers to the speed of light. The formula for TDoA calculation is:

$$\Delta d = c * (\Delta t)$$

[29]

To find the location of the mobile device in 2-dimensional space, multiple stationary devices have to be used to triangulate the location of the mobile device. The difference between ToA and TDoA is that in ToA all devices must have a synchronised clock, while for TDoA only the stationary devices need to have a synchronised clock.

2.1.3 RSSI & RX Power level

Receive Level (RX) and RSSI measure distance using radio signal strength received by the antenna. RX is measured in milliWatts (mW) or Decibel-milliwatt (dBm) which is an absolute measurement. While RSSI is measured through a relative measurement set by a given device manufacturer. This means that RSSI is not measured on the same scale between different devices. Some manufacturer might choose to have a "normal" scale going from 0-100, while another one could choose to do 0-60 [35]. Additionally, this type of measurement could be problematic because different chip-set manufacturers for the mobile phone could give different readings [15].

Power level based measurements also struggle a lot with signal noise. Signal noise could be created by reflective surfaces in the environment the beacon is installed in, or by having to fight over the same frequency as a lot of other nearby Bluetooth devices. Objects such as pillars can also physically block the signal making readings unreliable.

The issue of signal noise can be partially remedied using a Kalman filter [8]. This filter is designed to estimate the value of a measurement based on a joint probability distribution. This will significantly reduce the amount of noise in the RSSI readings of a beacon. [8]

To find your position in 2D space power level based measurements use trilateration in the same way that ToA does.

2.1.4 AoA & AoD

Angle of Arrival (AoA) and Angle of Departure (AoD) are different from the other listed methods, because they can find your location in 2D space using only a single beacon. In AoA a transmitter sends a signal to a receiver which has an array of multiple antennas. The receiving device can determine the direction using the phase difference of the incoming signal. The calculation to determine the angle of the incoming signal is:

$$\theta = arccos((\psi\lambda)/(2\pi d))$$

where ψ is the phase difference, λ is the wave length and d is the distance between each adjacent antenna.[36]

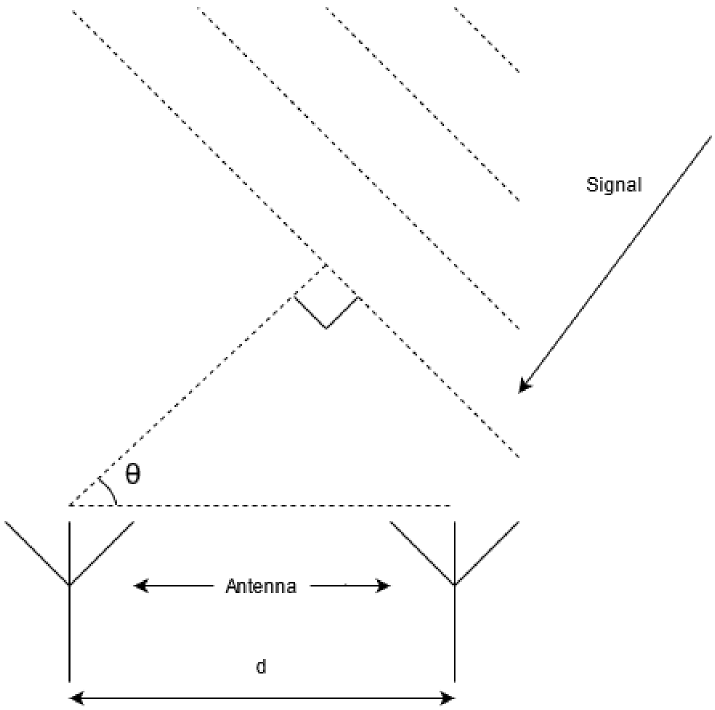


Figure 2.2: Using the phase difference to get the Angle of Arrival

AoA works the opposite way from AoD, this means that in this case the transmitter has multiple antennae and the receiver only has 1 [18].

In theory, this technology would be the best one for our use due to needing less beacons and being very accurate. The issue is that there are very few mobile phones at the time of writing with Bluetooth 5.1 support, this could be very relevant in a few years though.

2.1.5 Fingerprinting

Fingerprinting can be used to improve the location accuracy of any of the technologies listed above. This is done by essentially calibrating the equipment to teach it that a given measurement is in a specific manually measured location. The issue with this though is that for example with RSSI, if a piece of furniture being added or removed could make the measurements inaccurate again. This is due to the furniture blocking the signal.

2.2 Positioning methods and technologies

IPS is an up and coming technology which keep attracting more attention due to its many use cases. Some of the use cases include: Human detection/tracking, sending messages to devices, navigation and many more. There are several different methods and technologies that can be used for an IPS. In this section we will compare a few different technologies, BLE, Wi-Fi and Ultra-Wideband (UWB).

2.2.1 GPS positioning

While GPS works great outside and in building through windows, it often struggle with its accuracy when it comes to indoor positioning. This is because the GPS satellites works best with a direct line of sight to devices. When indoors, the signals weakens or distorts as it travels through walls, and the result is inaccurate.

2.2.2 BLE positioning system

Bluetooth Low Energy (BLE) [24] is a radio-frequency technology which can be used for device tracking, indoor navigation, and many more indoor positioning cases. Its low energy, fairly accurate and supports both iOS and Android.

An IPS using BLE relies on Bluetooth beacons being mounted on objects, walls or ceilings, from where they emit radio signals at predetermined intervals. Bluetooth beacons operate on the BLE protocols. They are low-power Bluetooth transmitters that can be detected by devices such as smartphones and BLE-enabled sensors. The beacons do not transmit signals constantly. They "blink".

Although one beacon is sufficient in establishing the presence of an object, it cannot pinpoint the specific location. Usually, the location accuracy increases with the number of beacons. BLE typically relies on RSSI value to estimate the location of devices defined by the protocols chapter 2.4.

BLE is the IPS solution that we are going to use for our project.

2.2.3 Wi-Fi positioning system

Wi-Fi is a local wireless network that uses radio waves to communicate data. A Wi-Fi positioning system uses several techniques to locate a connected object or device. A Wi-Fi IPS uses APs and already existing infrastructure to calculate a device's location. By analyzing the signal strength of multiple Wi-Fi signals and knowing the locations of the APs, you are able to determine the approximate location of a connected device [32].

The two primary methods for tracking Wi-Fi enabled UE is RSSI and fingerprinting. RSSI of a Wi-Fi signal is inversely proportional to distance, meaning that one value decreases at the same rate that the other value increases. An example of this is that the closer a connected device is to an AP the stronger the RSSI values, and vice versa.

The main advantage of Wi-Fi positioning is accessibility. Many buildings already have a Wi-Fi network infrastructure already installed. It also has higher data transfer and greater range than BLE.

One of its disadvantages is its security. Wi-Fi is such a common technology so its less secure to being hacked [26].

2.2.4 Ultra-Wideband indoor positioning

UWB is a short-range radio technology which uses Time-of-flight (ToF) to get positioning. UWB indoor positioning has a significant advantage over the others. It has the best accuracy with up to 30cm [23], and almost no interference. Its biggest disadvantage is that it only works with iPhone 11 or newer models [34].

	BLE	**Wi-Fi**	**UWB**
Location accuracy	<5m	10-20m	30cm
Compatibility	iOS and Android	iOS and Android	Only iOS, iPhone 11 or newer.
Battery life	Low energy consumption	Moderate	Low energy consumption
Infrastructure	Dedicated infrastructure	Uses existing infrastructure	Dedicated infrastructure

Table 2.1: Comparison of IPS technologies

2.3 Spec sheet comparison of commercial beacons

This table is a quick comparison based on the technical data sheets from the manufacturers of these beacons. The rest of the thesis will go more in depth on the usability differences between these beacons, but it is possible to do a basic comparison based on their specification sheets. This is also part of how we decided which beacons to buy.

Beacon (providers)	Blueup BlueBeacon Maxi	Accent Systems iBKS PLUS	Accent Systems
Supports	ibeacon, Eddystone, Quuppa Emulation	ibeacon, Eddystone	ibeacon, Eddystone
Range [meters]	N/A	up to 100m	up to 50m
Battery life [months]	18-120	120	30-40
Battery type	2xAA	4xAA	1xCR2477
Size [mm]	90x50x29	24x84x84	11.3x52.6mm (circular)
Price [per beacon]	∼ 20€	∼ 23$	∼ 14$
Datasheet	[6]	[21]	[22]

Table 2.2: Comparison of commercial beacons

This table gives us a basic overview of the beacons we had chosen to buy. There are many other beacons on the market with very similar specifications. Because due to the short time line of this thesis it was important to choose beacon providers which had their beacon in stock, and could be shipped quickly and easily.

By using this table we could determine which beacons were not right for us. We needed the beacons to at least last 2 years or 24 months. Its not viable for the university to have to change the batteries every year or so. So that eliminated a lot of the potential beacon options.

They needed to support BLE and either iBeacon or Eddystone. Which all our beacons did thankfully. Their size should be as small and therefor less detectable as possible, considering they where going to be put up in the university. And last but not least, the beacons needed to be capable of being funded. The beacons should be as cost effective as possible, which will help our case for funding.

2.4 Beacon protocols

iBeacon and Eddystone

iBeacon [20] and Eddystone [13, 14] are 2 different profiles developed by Apple and Google respectively. They are used by mobile phone apps on both iOS and Android to communicate with compatible Bluetooth beacons. These 2 beacon formats are different in some of the payload that is sent in each frame.

The iBeacon frame contains:

1. UUID: General information about the beacon (Owner for example)
2. Major: The beacon's most general information (building and/or floor its on)
3. Minor: More specific information about the beacon (beacon is in room 42)

While the Eddystone frame contains:

1. UID: Similar to UUID major and minor, has a namespace and instance
2. URL: a compressed URL that can be directly used by an end user
3. TLM: telemetry data useful for fleet management
4. EID: additional security measure

These extra Eddystone packets mean its more controllable and has better fleet management than iBeacon, but it also makes it more difficult to implement. Eddystone is also open source while iBeacon is proprietary. It's very important for Eddystone to be open source, because somebody could continue maintaining Eddystone after Google shuts it down on April 1st, 2021.[13]

	iBeacon	**Eddystone**
Compatibility	Compatible with Android and iOS, but only native to iOS.	Native to both Android and iOS.
Battery life	A little more than Eddystone.	Bit less than iBeacon depending on which frames is being used.
Profile	Its proprietary software, so the specifications is controlled by Apple.	Its specifications are open source on GitHub. So developers can contribute to make it better.
Usage	For sending messages, iBeacon needs an app on smartphones to do a particular task with the received information. So either create our own app or use a third party app.	Eddystone can send URL directly to smartphones. For iOS, its supported on Chrome. And on Android its supported on the physical web.
Security and privacy	The signal transmitted by beacons using iBeacon is a public signal, meaning it can be detected by any iOS and Android devices.	The EID feature, that constantly change which allows a broadcast signals to only be identified by "authorized clients".
For the user	Less features, more documentation, and simpler to implement.	More features and sends more information, but more complicated when it comes to integration.

Table 2.3: Comparison between iBeacon and Eddystone [16]

AltBeacon

AltBeacon [3] is an open source beacon protocol from Radius Network. It's very similar to iBeacon as it covers the same functionalities as it, only that its open source, not company specific and not widely used [5]. AltBeacon can deliver more data per message than iBeacon, however its main problem is that its not as widely supported. AltBeacon was created to make an open, competitive market for beacon protocols.

Competing against Apple's iBeacon is not an easy task. In time it may increase in popularity due to the fact that it is able to carry more data per message, and being open source.

Despite the fact that AltBeacon was a good option, we decided to use the iBeacon protocol, as it suited our project better.

2.5 Related works

2.5.1 Fighting the global pandemic

In an effort to track, and possibly stop the flow of new Covid-19 cases, Governments around the world are rushing to track the locations of their populace [7]. One way to do this could be to set up indoor positioning systems in malls, universities and schools, and a whole range of other public locations where strangers might run into each other. The government could write a smartphone app which uses Bluetooth technology, and encourage, or even mandate, that citizens download the application. One example of an already existing app in Norway is Smittestopp.

Smittestopp is designed with user privacy in mind. Once you download the app you are already on your way. There is no need to log in, thus you have never told the app who you are. The app never shares data in regards to your movement, but instead "identifies" you for other application users within the proximity using Bluetooth [27]. The way it does so is by sharing an auto generated number. This number is constantly changing, and is not traceable to any particular person in the first place. But all the numbers you "meet" are stored on your phone.

The way the app uses these stored numbers, is when a person reports that he or she is infected by the virus in the app. The person must then log in, and the list of stored number over a 2-week period are sent into a central overview. Phones are then checked and if a number from the central overview is also stored on your phone, you are then considered a close contact, and will have to quarantine [27].

3 Methodology

The main objective of this system is further increase the accuracy of indoor navigation. This chapter discusses what steps that needed to be taken in order to setup our IPS.

3.1 Base Stations

When designing a positioning system, there are typically two components needed. A base station (in our case a Bluetooth beacon) and a mobile station (in our case a mobile phone). In GPS, the satellites acts as the base stations, while people, cars etc. works as the mobile stations. We're using beacons for positioning, so the beacons are the base stations, while smartphones or other smart devices acts as the mobile stations.

When using beacons for positioning there are at least two design options:

1. The beacons are at fixed locations constantly listening for known signals from smart devices.

2. The beacons are at fixed locations transmitting signals that smart devices pick up.

Option 1 is a good option for positioning, however the beacons would do a lot more work than in option 2. This would drain our beacons battery life at a faster rate.

We chose option 2, where the beacons will be our base stations transmitting signals at a predetermined interval, and the mobile station, in this case smartphones, will listen for these signals. This will give our beacons longer battery life, as their only job is to transmit signals.

Some companies prohibit smartphones, however since we are installing these base stations at UiS it shouldn't be a problem.

Requirements for the base stations:

- BLE enabled.
- Either iBeacon or Eddystone compatible.
- Small and mostly unnoticeable.
- Should run on a battery.

A prototype that fitted the regiments was built using the ESP32 series. For fulfilling the last requirement, we connected the ESP32 to a LiPo battery. A powerbank could also had worked.

3.1.1 ESP32

ESP32 is a series of low cost, low power system on a chip microcontroller with integrated Wi-Fi and dual-mode Bluetooth. It is a successor to the ESP8266.

Our base stations will be the SparkFun ESP32 Thing [33]. They will use the Eddystone protocol, which smartphones will pick up.

The SparkFun ESP32 Thing module features:

- Up to a 240MHZ dual-core Tensilica LX6 microprocessor
- 520kB internal SRAM
- Micro USB port
- 2-pin JST connector port
- Integrated LiPo battery charger
- Integrated sleep mode
- BLE compatible
- Configurable

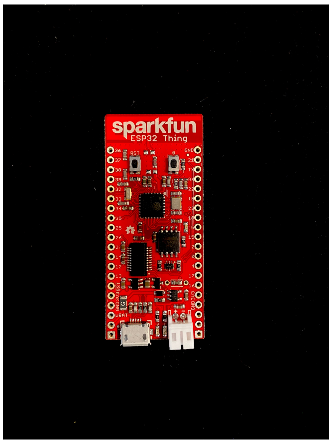


Figure 3.1: Picture of the SparkFun ESP32 Thing Model

Should this system go into production, the ESP32 could be replaced by a different model to further reduce the footprint and cost. We chose the SparkFun ESP32 Thing mainly for its integrated JST connector port. This port made it so that we could connect a LiPo battery for our power source. It also filled all our requirements for a base station.

The ESP32 could also be replaced by a commercial beacon.

3.1.2 iBKS 105

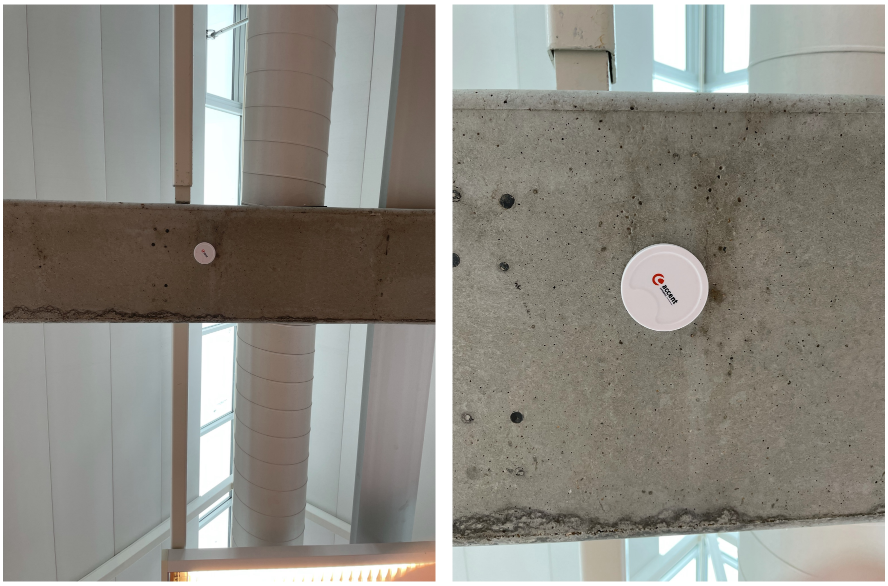
We chose the iBKS 105 [22] model for our commercial beacons. They operate on the BLE protocol, and supports both Eddystone and iBeacon protocol. The iBKS 105 beacons run of a coin battery, and can last for 24-48 months depending on their configuration. Our configuration should give it a battery life of approximately 38 months.

Figure 3.2: Picture of the iBKS 105 beacon

3.1.3 Beacon installation

The beacons were always installed on concrete due to it being a non reflective surface, either on the side of pillars or on the bottom of beams going across the ceiling. For the ESP32's it was important to keep in mind the placement of the ESP32, and the relative position of the battery. The battery is almost the size of the ESP32 and could interfere with it's signal. The battery was always on the side of the surface the beacon was being attached to.

The iBKS 105 could be mounted using the adhesive on the back of the beacon as shown in figure 3.3



(a) iBKS105 shown from a small distance

(b) iBKS105 shown close up

Figure 3.3: An iBKS 105 mounted to the bottom of a vertical beam using the adhesive on the back

The ESP32's were put in the small plastic bag they came in together with the battery, and taped to the surface they were being attached to as seen in figure 3.4.

(a) ESP32 attached to the bottom of a vertical beam going across the ceiling

(b) ESP32 attached to the side of a pillar

Figure 3.4: ESP32's mounted in the hallway of Kjølv Egeland building in the university

3.1.4 Little-endian

When we tested our ESP32 we got a different UUID than we expected. We found out that the ESP32 is a little-endian native architecture, meaning that the ESP32 stores the least significant byte of a word at the largest memory address (Illustrated in figure 3.5).

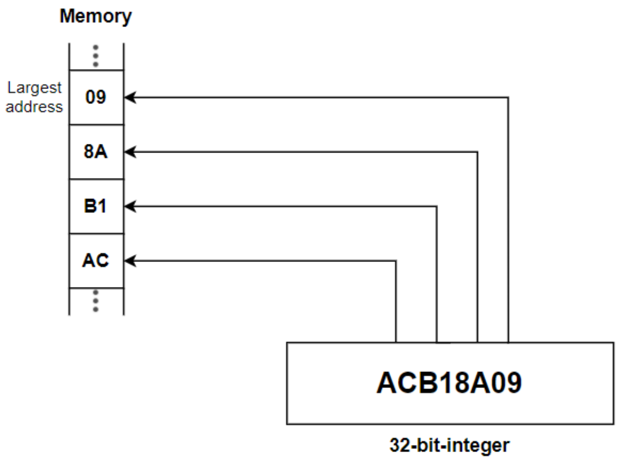


Figure 3.5: Little-endian

So in the case of figure 3.5, the 32-bit-integer ACB18A09 would be stored in the memory as

0x09 0x8A 0xB1 0xAC 0x8A

This is important to note, or else we can interpret data in an incorrect fashion. Endianness is important to know when you are writing and reading the data across networks from one system to another [28]. In our case this data was from our ESP32s to our monitoring app.

3.1.5 Differentiating between beacons

For our system we decided to use iBeacon as the beacon protocol, this means that to identify each beacon there is a UUID, Major, and Minor. In our system the UUID would always be $098AB1AC-28F4-42D6-A080-FD45387B4231$, the Major and Minor combination should be unique on the other hand.

The Major in our case was 1104 on all our beacons. This was chosen because we found that starting the Major with 0 would sometimes create some issues, so we would start counting from 11. The first half would signify what building the beacon is placed in, all beacons in our case were in the same building, so it was the same on all of them. The 2nd half of the Major would signify the floor of the building which in our case is the 4th floor, so it became 04.

The Minor would either start counting from 0 or from 10 depending on which beacon it was. The ESP32's started counting from 0, so since we had 7 of them the ESP32 Minors would range from $0-6$. Meanwhile the iBKS 105' would start counting from 10, so since we had 5 of them their Minors would range from $10-14$.

Figure 3.6: Showing all the beacons with their Major and Minor values using the Beacon Sample app by Apple

3.2 Mobile Phones

When testing how the IPS works we primarily used 2 different phones, 1 Android and 1 iPhone. This was not to compare how they relatively performed, but to show that the IPS works on both platforms. Additionally a different Android phone was used when fingerprinting the area, but it was not used for measuring.

The primary phones used for testing was the Huawei Mate 10 Lite for Android, and the Apple iPhone 12 mini for iOS. The 12 mini was also used for the RSSI measurement app. Additionally a Samsung Galaxy S8 was needed when mapping the 4th floor, due to the Huawei Mate 10 Lite not having the necessary hardware for mapping[4].

Phone	Huawei Mate 10 Lite	Apple iPhone 12 mini	Samsung Galaxy S8
release	October 2017	October 2020	April 2017
Bluetooth	4.2, A2DP, LE, aptX	5.0, A2DP, LE	5.0, A2DP, LE, aptX
OS when testing	Android 8.0 / EMUI 5.1	iOS 14.5.1 (14.4.2 when testing RSSI)	Android 9.0 / One UI 1.0
System on a Chip (SoC)	Kirin 659	Apple A14 Bionic	Exynos 8895
Full specifications	[19]	[25]	[30]

Table 3.1: overview of relevant specifications of mobile phones used

Looking at table 3.1, it can be seen that the phone representing the Android side of things is slightly older than it's Apple counter part. As mentioned previously this is not an issue due to the purpose of this thesis is not to compare the 2 platforms against each other.

Another difference between the 2 main testing phones is that the Android phone had no SIM card in it, meaning that if a part of the building has poor Wi-Fi it could struggle with positioning. This is because it can't fall back on cellular broadband for internet. This could give it a harder time in some scenarios. This should mean that it, thanks to its age and lack of cellular to fall back on, it should be relatively close to a worst case scenario for the type of phone that might be used with the IPS. So if it works with this Android phone it should work with almost anything Android.

3.3 RSSI monitoring app

To help us measure the RSSI values that a phone reads from a beacon, and be able to extract this data and easily plot it, we developed an iOS app written in swift. This connects to an iBeacon, reads it's major and minor values, the time it was last seen, its RSSI, the proximity, and the accuracy. Proximity is a value which gives a very rough estimate of the distance between the beacon, and the mobile phone in 4 possible cases[11].

- unknown: The proximity could not be determined.
- immediate: The beacon is in the immediate vicinity.
- near: The beacon is relatively close to the user.
- far: The beacon is far away.

The accuracy value is used to differentiate between beacons with the same proximity value, but is not intended to be used as a way of finding the precise location of a beacon[1].

3.3.1 The code

```swift
override init(){
    super.init()
    // requests permission to scan for beacons
    locationManager = CLLocationManager()
    locationManager?.delegate = self
    locationManager?.requestWhenInUseAuthorization()
}

func locationManager(_ manager: CLLocationManager, didChangeAuthorization
↪ status: CLAuthorizationStatus){
    // if app is authorized to use location
    if status == .authorizedWhenInUse{
        // monitoring is available for our device
        if CLLocationManager.isMonitoringAvailable(for:
        ↪ CLBeaconRegion.self){
            // ranging is available for our device
            if CLLocationManager.isRangingAvailable(){
                startScanning()
            }
        }
    }
}
```

Figure 3.7: *Line 26-45* in *ibeacon distance measure/ContentView.swift*

Figure 3.7 shows what happens when the app is opened. First the app will create a location manager, and request permission to get the device location on first open. This will only happen the first time the app is opened. After the permission has been given, the app will check if it was given the permission. If it was, then it will check if region monitoring is available for the current device. If it is, then it will check if ranging is available for the current device. After the permission has been checked, and it has been checked if the device the app is installed on, has the necessary hardware to support scanning for iBeacon. The app will call the `startScanning()` function.

```
func startScanning(){
    // "uuidgen" in terminal can generate a uuid
    let uuid = UUID(uuidString: "098AB1AC-28F4-42D6-A080-FD45387B4231")!
    let major = CLBeaconMajorValue("1104")!
    let minor = CLBeaconMinorValue("0000")!
    let constraint = CLBeaconIdentityConstraint(uuid: uuid, major: major,
    ↪  minor: minor)
    let beaconRegion = CLBeaconRegion(beaconIdentityConstraint: constraint,
    ↪  identifier: "TestBeacon")

    // start monitoring for the beacon identified above
    locationManager?.startMonitoring(for: beaconRegion)
    // start raning said beacon
    locationManager?.startRangingBeacons(satisfying: constraint)
}
```

Figure 3.8: *Line 47-59* in *ibeacon distance measure/ContentView.swift*

Figure 3.8 Shows the `startScanning()` function. The function defines a UUID which must be added to the beacon constraint[9], and the beacon constraint must be added to the beacon region[10]. This essentially is telling the app which beacons to look for. The Major and Minor are optional, but they were used in our case to differentiate between beacons with our system for differentiating beacons which was detailed in chapter 3.1.5.

```
1  // when start scanning works we will get a callback
2  func locationManager(_ manager: CLLocationManager, didRange beacons:
   ↪  [CLBeacon], satisfying beaconConstraint: CLBeaconIdentityConstraint) {
3      // sets format for date
4      let df = DateFormatter()
5      df.dateFormat = "HH:mm:ss"
6      // with the range of the first beacon found
7      if let beacon = beacons.first {
8          lastDistance = updateDistance(distance: beacon.proximity)
9          // beacon major & minor, just used for additional identification if
           ↪  explicit major minor wasnt previously given.
10         major = CLBeaconMajorValue(truncating: beacon.major)
11         minor = CLBeaconMinorValue(truncating: beacon.minor)
12         // print the rssi into console, this can be copied into a file when
           ↪  you're done to read whatever information you want to have.
13         print(beacon.rssi)
14         // to make the app more readable information only updates when
           ↪  beacon was successfuly contacted
15         if beacon.proximity != .unknown {
16             lastSeen = df.string(from: beacon.timestamp)
17             rssi = beacon.rssi
18             //
               ↪  https://developer.apple.com/documentation/corelocation/clbeacon/1621551-accuracy
19             accuracy = beacon.accuracy
20             rssiDistance = calcDistanceRSSI(rssi: beacon.rssi)
21         }
22     }
23 }
```

Figure 3.9: *Line 61-83* in *ibeacon distance measure/ContentView.swift*

Figure 3.9 shows what happens when a beacon is found. The `if` on *line 7* will take the first beacon that is found and connect to it to read information from it.

The information read from the beacon is the proximity on *line 8*. The Major and Minor on *line 10-11* which was used to confirm that the beacon the app found, was the same one as expected based on the filters from figure 3.8. *line 13* will print the RSSI, this part of the code is detailed further in chapter 3.3.2.

The `if` statement on *line 15* is only relevant for what is displayed on the phone. It was put behind the `if` statement to try to improve the readability of the information on the phone. If the beacon couldn't be read momentarily, the values wouldn't all jump to 0.

The full source code can be found in Appendix A.

3.3.2 Collecting the RSSI measurements

The testing for RSSI values was done in 2 different environments for 6 different distances, each distance double the previous. The distances were 1, 2, 4, 8, 16 and 32 metres. The locations used for indoor testing, was the East hallway on the 4th floor of the KE building at the university, shown in figure 3.10. The outdoor testing was done in an outdoor football field.

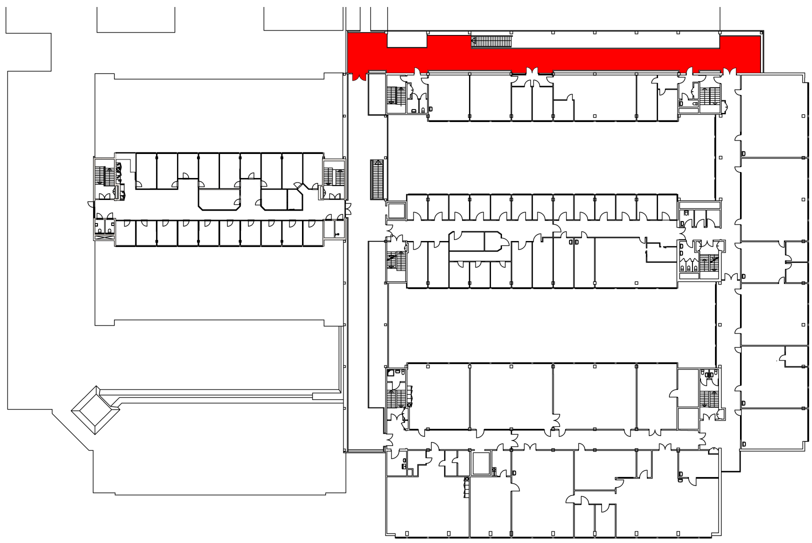


Figure 3.10: The hallway used for indoor RSSI measuring marked in red. Note: This image is flipped 90 degress counter clockwise so North is to the left.

To get the RSSI values the `print` statement from figure 3.9 on line 13 becomes important. This would print the RSSI values as measured by the phone into the debug output, as shown in figure 3.11. Each distance would be measured for 2 minutes doing 1 beacon at the time.

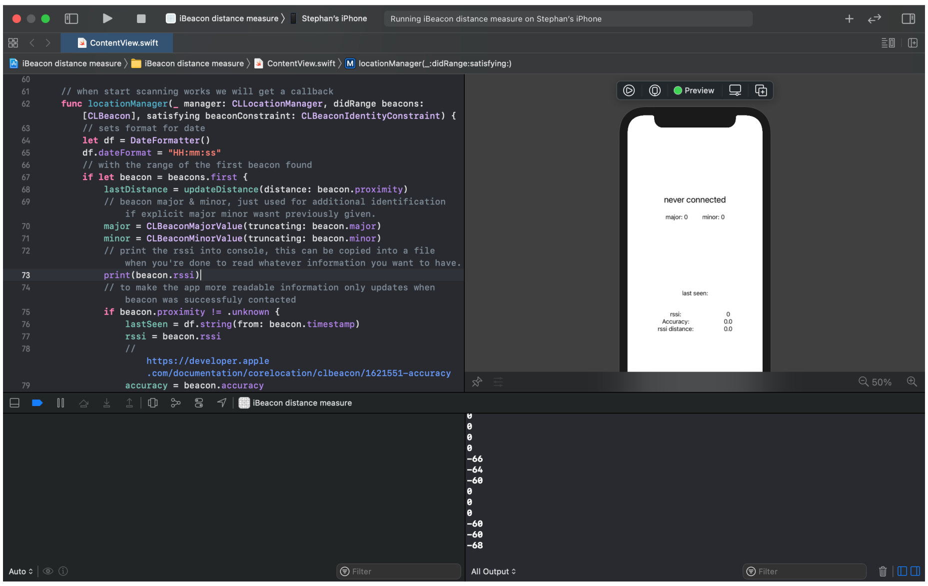


Figure 3.11: RSSI readings being measured can be seen in the bottom right of this figure. Note the preview window in the middle right does not show a live preview of the phone screen, it there to show the layout of the UI.

3.4 Indoor Mapping Service

We chose to use IndoorAtlas as our service to test indoor positioning. This was because they support integration with Mazemap, which the university currently uses as their indoor mapping provider. All testing of how good the location accuracy was before any beacons were installed and fingerprinting had been done were done using Mazemap. Testing after beacon installation and fingerprinting was done using the testing functionality of the IndoorAtlas map creator, and their positioning test app for iOS due to the map creator not being available on any other platform than Android.

3.4.1 Test map

We did 3 separate test runs to determine the positioning accuracy, the first run was done before any installation. The second was done installing 8 beacons in various locations on the 4th floor of the KE building, and third test was done with 12 beacons installed in various locations. This was to see how much the accuracy would improve with a more dense beacon map.

For each of the separate test location, stood we still for 10 seconds, while screen recording the map app to see where the Blue Dot were. The distance that the Blue Dot has to our actual location, would be measured afterwards using a cad model of the building. Due to the inaccuracy of this method, a margin of error off about 0.5-1m should be applied to all results. A map of all the testing points can be seen in figure 3.12. Whenever a test location is referenced, the letters from this figure will be used.

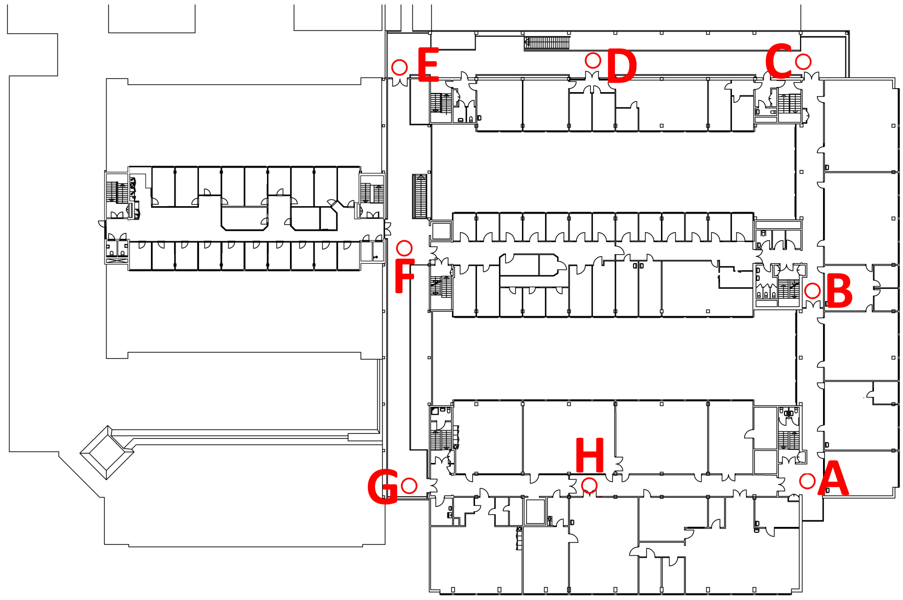


Figure 3.12: Map of where location accuracy was tested in the 4th floor of the KE building rotated 90 degress counter clockwise (north is to the left)

3.4.2 Layout of the Beacons

Beacons were installed in 2 different configurations. 1 medium density configuration and 1 high density configuration. The medium density configuration used 7 ESP32's and 1 iBKS105, while the high density configuration used 7 ESP32's and 5 iBKS105's. This were all the beacons we had.

The beacon maps can be seen in figure 3.13 and 3.14. The medium density map was based on the idea of getting a basic coverage map of the 4th floor to do basic testing with. Based on the results of this test, some beacons were moved and others were added to try to fix some of the areas that didn't have as good location accuracy.

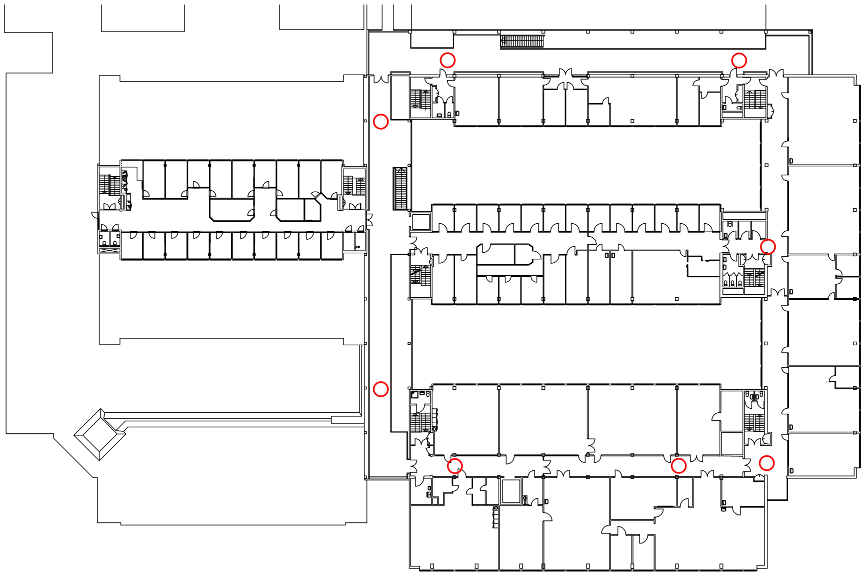


Figure 3.13: The medium density beacon map, where beacons are placed near corners and door to hallways to ensure coverage in important places

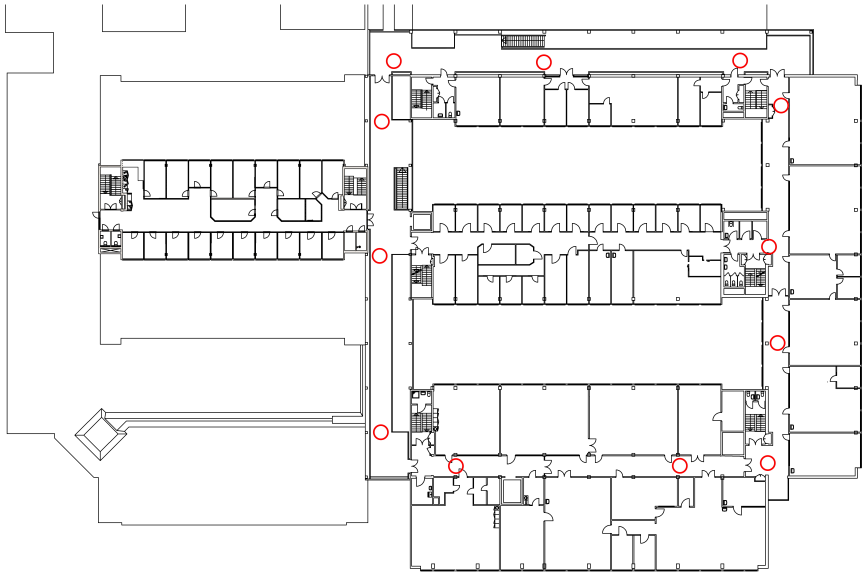


Figure 3.14: The high density beacon map where beacons are placed in the middle of long corridors, and more doorways are covered.

3.4.3 Creating the map in IndoorAtlas

When creating a map in IndoorAtlas, first you must create a venue[12]. This is the building that you are going to be adding floors to. Once the venue has been created you must then add floor plans to it[2]. These floor plans are an image which must be aligned over a satellite image of the address of the venue, and it must be defined what floor the floor plan belongs to.

Once the floor plan has been created, Waypoints must be added to it. These are essentially virtual checkpoints to walk between when fingerprinting. We put a Waypoint in each of the testing locations from figure 3.12. Once this has been done, an Android phone which is supported by Google ARCore[4], can be used together with the IndoorAtlas MapCreator 2 to start fingerprinting. When this is being done, it is important to already have any beacons that are going to be used set up. The fingerprinting should automatically detect the beacons when you walk between checkpoints, and then be able to use the beacons to assist with indoor positioning.

In our case we walked between each Waypoint twice. Once in each direction to make the calibration as accurate as possible. The phone was always held in the same way, and we tried to maintain a consistent walking speed too.

3.4.4 Recording the data

The process of recording the data, to measure the accuracy of the positioning, was to use the screen recording feature built into iOS and the Android skin that the Huawei Mate 10 Lite uses. For each location marked in figure 3.12, we would attempt to stand approximately in the exact same spot each time for a bit over 10 seconds. The spots were marked by physical features in the area. This method likely meant there was a margin of about 1 metre. Additionally, people in the surrounding area could be present during a test, and not during another. This could've affect the results the same way as it would in the RSSI measuring tests.

The app's used were Mazemap in browser as 'before' testing, meaning before any beacon's were installed. This was to get the most accurate possible impression of how the positioning accuracy was before any improvements were implemented. To test after beacon installation, the IndoorAtlas Positioning app was used on iOS, while the IndoorAtlas MapCreator 2 was used to test on Android.

The recordings would be loaded up on to a computer, and played back using software which allows skipping forwards or backwards through time exactly 1 second. In this case, that software was VLC media player. The setting required to replicate this method can be seen in figure 3.15.

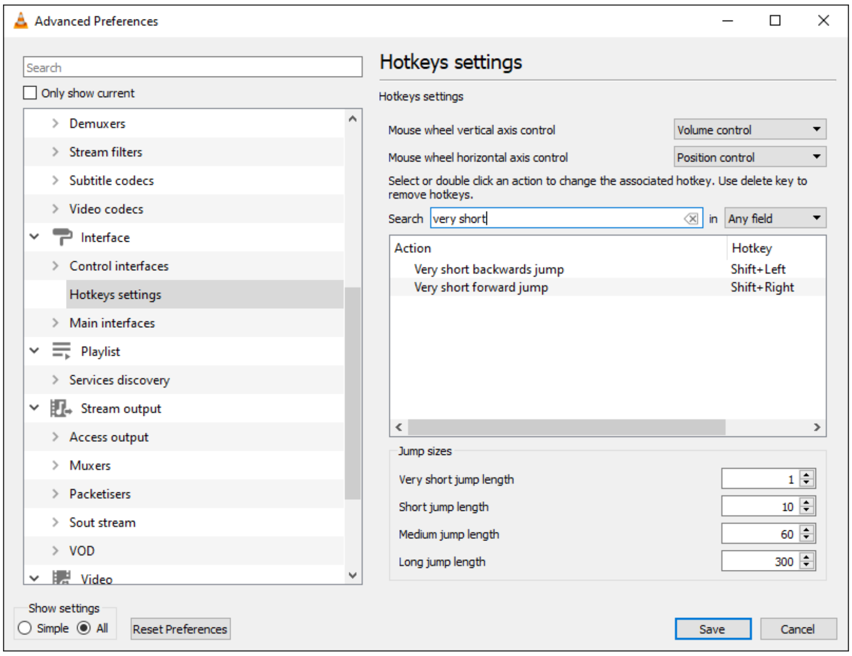
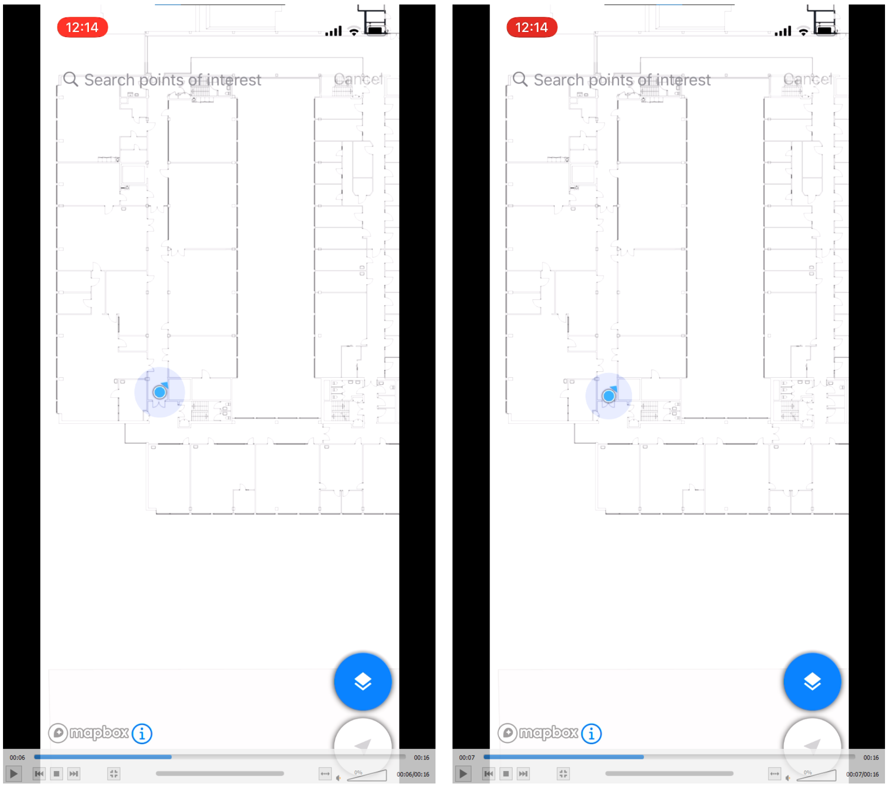


Figure 3.15: The setting required to do 1 second jumps through a recording played back in VLC. Make sure to set 'Very short jump length' from the default value of 3 to 1, and search 'very short' to find what the default key-bind is for your system.

For each recording, we would skip through it 1 second at the time. Then using CAD drawings of the university, we would measure the distance from where we were standing, to the center of the BlueDot. We tried to align it using features of the area, such as walls of other rooms if the dot was in the middle of a room for example.

An example of the process can be seen in figure 3.16. This figure is showing the 4th and 5th measurement taken when testing with a high beacon density on iOS at location A. It can be seen how the BlueDot has moved slightly down between these 2 frames, and that can also be seen in the data shown in chapter 4.2.2. This was how the measurements would be taken. Go through the video 1 second at the time for 10 seconds. If the BlueDot moved, the distance the BlueDot has from the location we were standing would be measured.



(a) Sample #4 (b) Sample #5

Figure 3.16: shows 2 different samples from the recording used to measure the data for high beacon density on iOS at test spot A

4 Results

4.1 RSSI Measuring app results

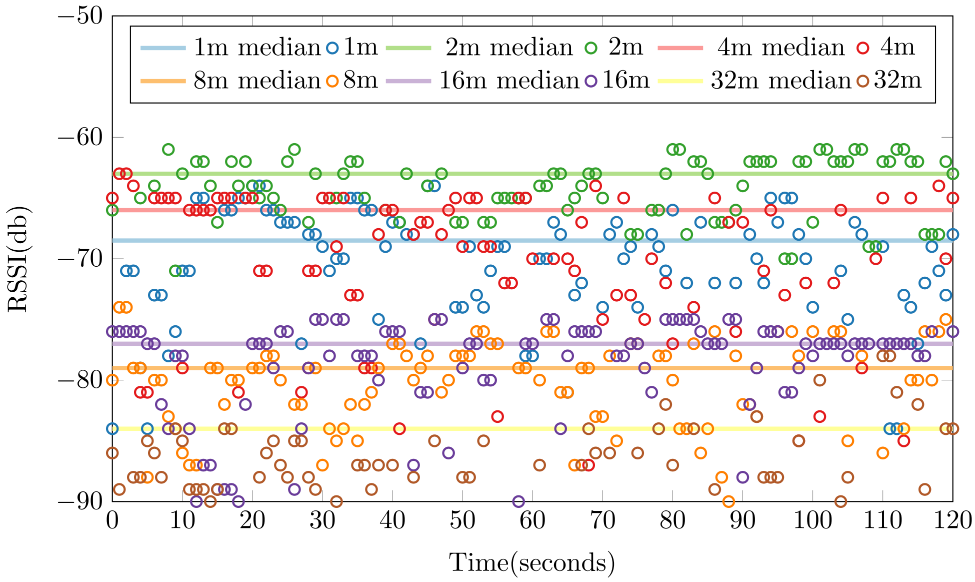
In this chapter we will analyse the RSSI measurements between the iBKS 105 and the ESP32 beacons, both indoors and outdoors. First they will be analysed individually and then compared against each other.

4.1.1 iBKS 105 Results

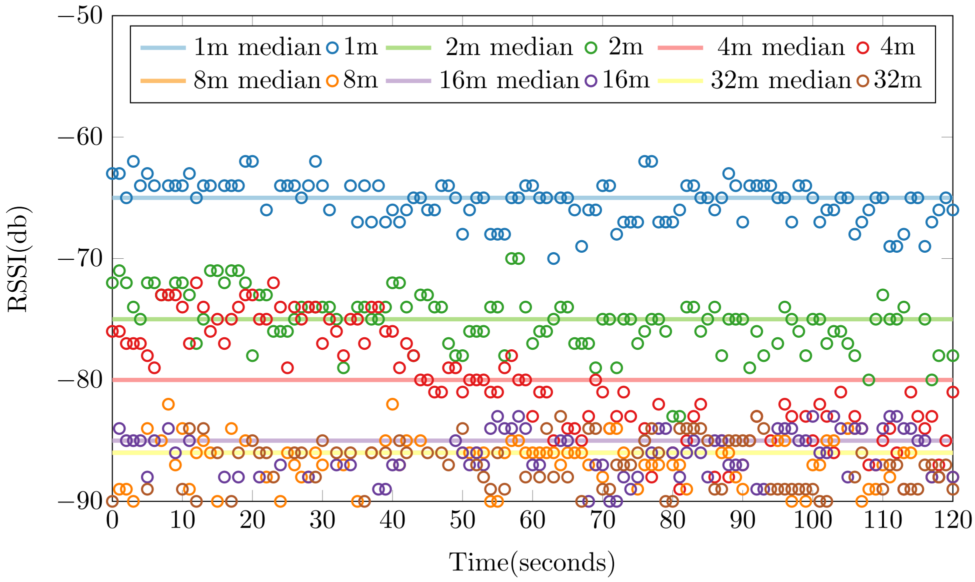
Starting with the iBKS 105, we will look at the raw measuring results in figure 4.1. Looking at these scatter plots its rather obvious how noisy RSSI can be. Some of this noise is inherent, however, it can also be seen how the figure showing the RSSI measurements outdoors is significantly less noisy. Although that one also has some unexpected aspects to it.

First of all, something which is immediately noticeable is how some of the medians are in the wrong order to what was expected. It is expected for the RSSI to decrease as the distance increases, but this was not the case after studying the data we collected. The 2m median on the indoor test is higher than the 1m median, and the 4m median is also higher than the 1m median. Although it at least managed to be lower than the 2m median. These distances are so close to each other that it could be argued that it doesn't matter so much if they get mixed up. But then if we look at the results for 16m and 8m, we see that these have also gotten mixed up. This would seem extremely unexpected except as will be seen later, RSSI does not decrease linearly. So the chance to get the further distances wrong in a perfect environment, is more likely than getting the close distances wrong.

Looking at figure 4.1b, which shows the outside measurements of the iBKS 105, the results are closer to what is expected. The reduction in signal noise either created by reflective surfaces or by devices in the surroundings, which are all fighting over the same Bluetooth frequency, has created a much cleaner scanner plot. Where the 1m and 2m results are closer to what is expected. What can also be seen is that as described previously the gap between the median RSSI decreases as the distance increases.



(a) iBKS 105 indoors



(b) iBKS 105 outdoors

Figure 4.1: 2 figures showing the iBKS 105 RSSI measurements over the 2 minute testing periods, including the median of each test distance.

Figure 4.2 is a result of using the same data from figure 4.1 and plotting it in a box plot. This reveals something which was not visible in the scatter plot. At 32m the box extends all the way up to 0db. This is because when the beacons RSSI could not be determined, it will be read as 0. This means that for a large amount of the 2 minute test, the iBKS 105 beacon could not be read, meaning that if this would happen again in a real world scenario, the beacon would almost give no assistance to positioning accuracy if a person was standing 32m away from it.

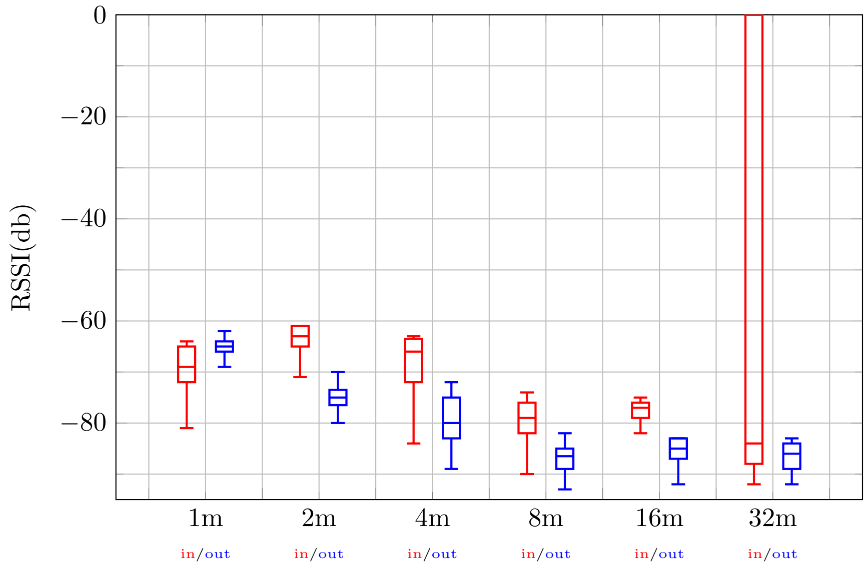


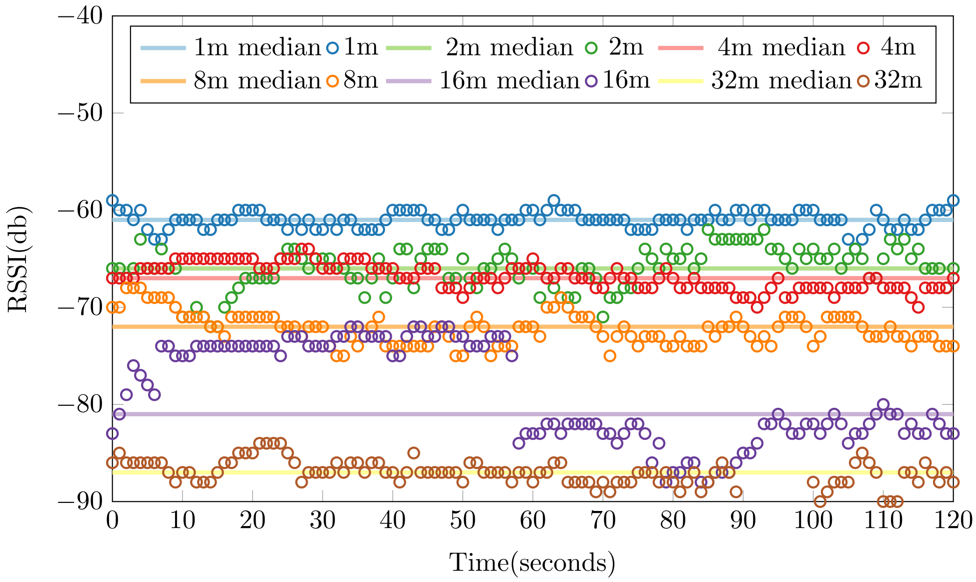
Figure 4.2: Comparison of RSSI measured on the iBKS 105 indoors and outdoors

Another thing that can be seen when looking at the data in the box plot, are the blue boxes for the outdoor readings. As the distance increases the RSSI values flatten out. This shows more clearly what was mentioned previously about RSSI not being fully linear. As the distance increases, the RSSI readings will relatively speaking come closer together.

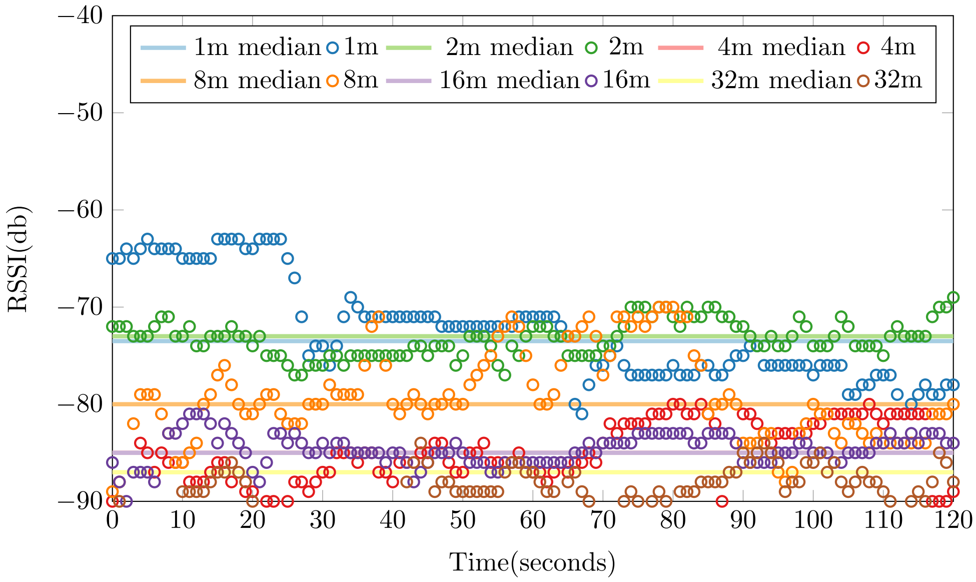
Overall the interquartile range for each box plot in the outdoor measurements is smaller than the one for the indoor measurements. The upper and lower whiskers also don't extend as far out from the interquartile range, which again shows that the overall inconsistency of the readings were lower when measuring in the outdoor environment, though not always so. The plot for 16m indoors appears to be slightly smaller than the one for 16m outdoors. So sometimes indoors readings can be quite consistent, but not always.

4.1.2 ESP32 Results

Looking at the raw RSSI measurements for the ESP32 in figure 4.3 a lot of the same themes as from chapter 4.1.1 can be seen. In the indoor test it managed to get the order of the medians correct, which is good although the measurements for 16m seem to have jumped from higher than expected to a bit lower than expected, while keeping the overall median mostly as expected. This is likely due to the fact that when testing, we were not the only people in the area. If anyone were to walk past with Bluetooth headphones this could create noise in the environment, which in return could affects the results. This would seem like a negative thing, but this is exactly what would happen in the real world, and that's what the indoor plot shows more than anything. It shows how the RSSI readings would look like in the real world. Inside of a building where there are lots of devices fighting over a small frequency range, and there are lots of reflective surfaces that can also add noise.



(a) ESP32 indoors



(b) ESP32 outdoors

Figure 4.3: 2 figures showing the ESP32 RSSI measurements over the 2 minute testing periods, including the median of each test distance.

Looking at the outdoor results in figure 4.3b, unexpectedly, they appear to be worse than the indoor results. The difference between each distance is smaller, and the measurements have a lot more overlap. The reason for this could be relative, due to the inconsistencies of doing a more 'real world' style test for the indoor results. The ESP32 could also have been 'lucky', and that the test were done at a time where there was relatively little signal noise in the environment. However, this would still not explain why the outdoor measurements are so poor.

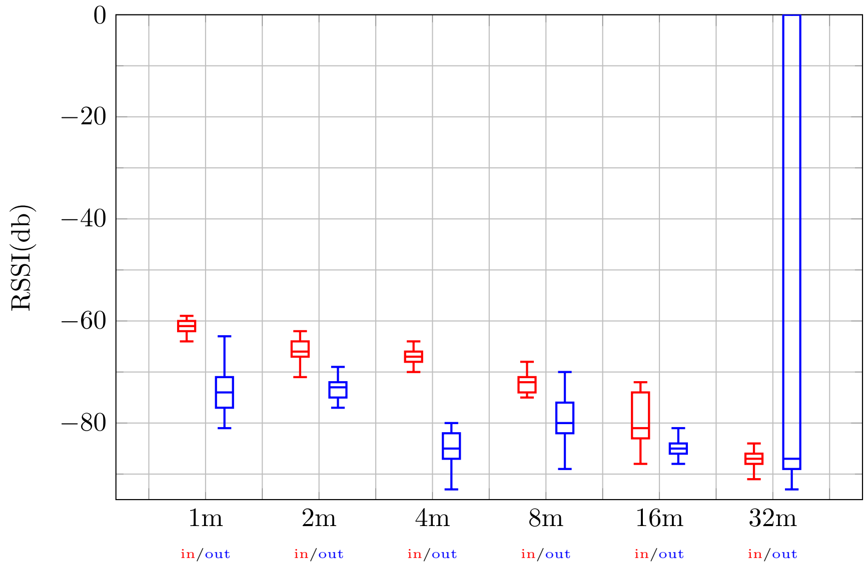


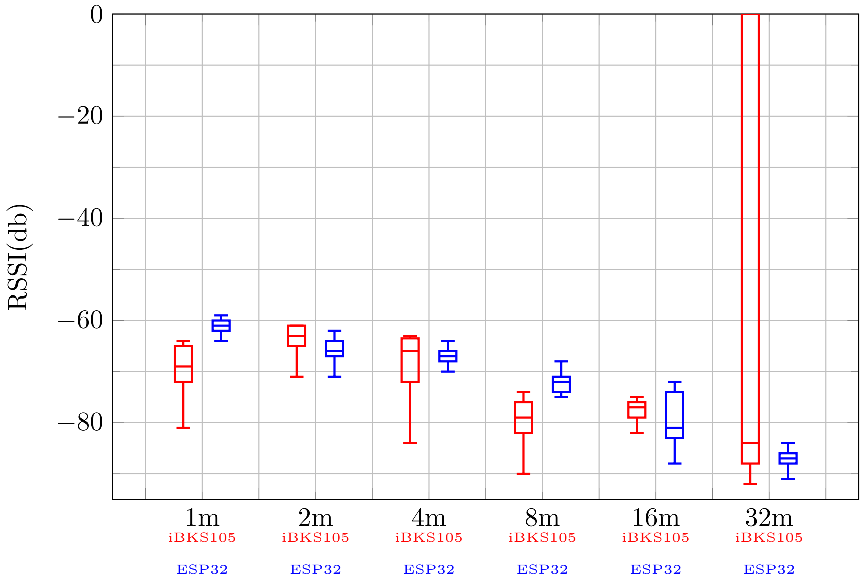
Figure 4.4: Comparison of RSSI measured indoors and outdoors

Putting the measurements from figure 4.3 into a box plot, we end up with figure 4.4. This one again shows the unexpected fact that the indoor measurements have turned out better than the outdoor ones. In fact, the outdoor 32m test couldn't establish the RSSI for a large amount of the test. Overall except for the 16m test, the measurements were relative consistent and stayed within a small range. Although the results for 1m, 2m, 4m, and 8m are closer than expected.

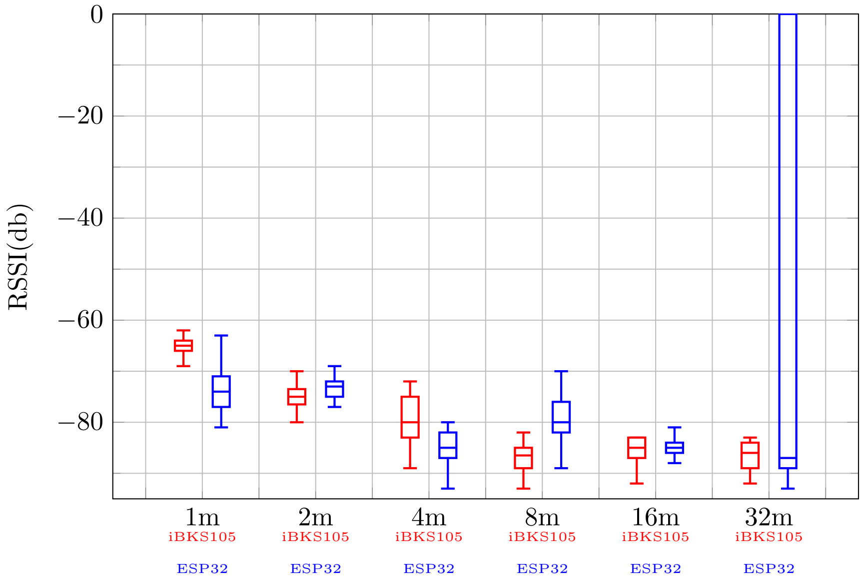
4.1.3 Comparison between the iBKS105 and ESP32

Comparisons of the iBKS 105 and the ESP32 can be seen in figure 4.5. For the indoor tests in figure 4.5a, it can be seen that overall both beacons performed very similarly. As mentioned previously, the ESP32's results were good in the indoor environment which can be seen in figure 4.5a. But when looking at figure 4.5b, the ESP32 relatively to the iBKS 105 looks the same as the iBKS 105 looks to the ESP32 in the indoor test.

Regardless of the strange switch around of their relative performance indoors and outdoors, the ESP32 in terms of RSSI performs extremely similarly to the iBKS 105. While it could be argued that the outdoor measurements are a more 'true' representation due to it being a more controlled environment. The thing that actually matters is how they perform in the real world, which would be inside. In short conclusion, the iBKS 105 and ESP32 perform so similarly that in terms of signal performance we're calling them identical.



(a) Indoors



(b) Outdoors

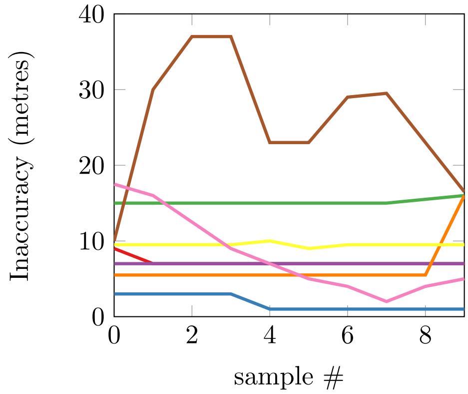
Figure 4.5: iBKS 105 vs ESP32 RSSI measurement comparison indoors and outdoors

4.2 Indoor positioning accuracy

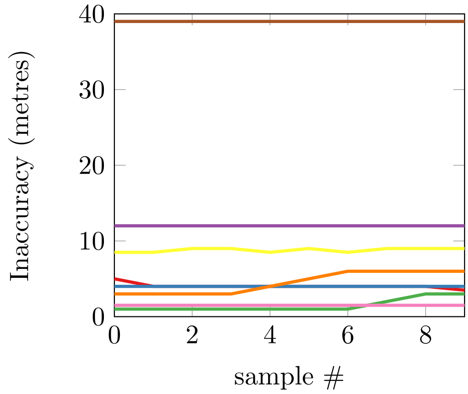
In this chapter we will analyse the inaccuracy of no beacons, medium beacon density and high beacon density on Android and iOS. Each platform is split into it's own sub chapter to avoid comparison, because the purpose is not to compare these 2 platforms.

4.2.1 Indoor accuracy results on Android between no beacons, medium density and high density beacon placement

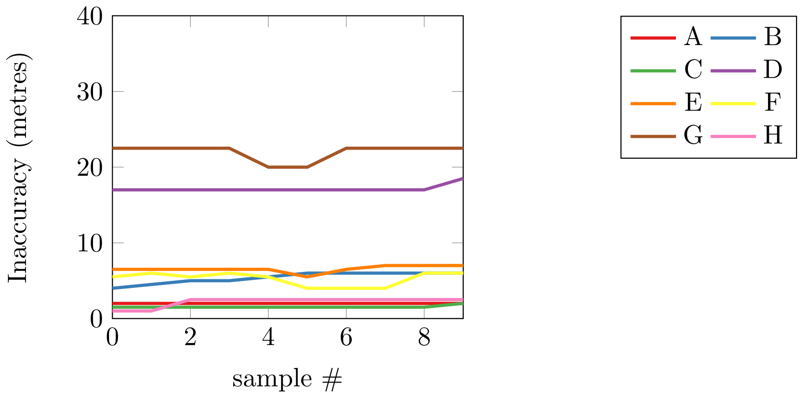
Starting with Android, we will look at the raw location measurements in figure 4.6 below:



(a) No beacons



(b) Medium beacon density



(c) High beacon density

Figure 4.6: 3 figures showing the inaccuracy measured at each testing point for 10 seconds, where 1 sample represents 1 second. Note: The same legend applies to all 3 figures.

In figure 4.6 we see in all 3 sub figures that some locations, like location G for example, is consistently more inaccurate than others. Like location A which consistently performs relatively good regardless of the beacon setup. It can also be seen that setting up beacons and mapping the area, doesn't always guarantee a better result. Location G for example, is stuck at a 39m inaccuracy for the entire 10 second test. Location D is stuck in a similar way, where it consistently for the entire duration of the test, is placing the BlueDot incorrectly, as can be seen in 4.6b and 4.6c. This could be due to a calibration error when fingerprinting or due to a loss of signal. Though over all, it is clear that as the number of beacons increases, the inaccuracy decreases. Which can be seen by the number of plots that are bunching together near the bottom on the Y-axis in 4.6b, and even more so in 4.6c.

Taking the data from figure 4.6 and using it in a table, we end up with table 4.6. It should be noted due to the inaccuracies of the testing methods used, all values have been rounded up to the nearest half a metre.

point	$\bar{x}$	$\tilde{x}$	s
A	7	7	0.5
B	2	1	1
C	15	15	0.5
D	7	7	0
E	6.5	5.5	3.5
F	9.5	9.5	0
G	26	26	8.5
H	8	6	5.5
total	10	7	2.5

(a) No beacons

point	$\bar{x}$	$\tilde{x}$	s
A	4	4	0.5
B	4	4	0
C	1.5	1	1
D	12	12	0
E	4.5	4.5	1.5
F	9	9	0.5
G	39	39	0
H	1.5	1.5	0
total	9.5	4.5	0.5

(b) Medium beacon density

point	$\bar{x}$	$\tilde{x}$	s
A	2	2	0
B	5.5	6	1
C	1.5	1.5	0
D	17	17	0.5
E	6.5	6.5	0
F	5.5	5.5	1
G	22	22.5	1
H	2	2.5	0.5
total	8	5.5	0.5

(c) High beacon density

Table 4.1: 3 tables showing the mean, median and standard deviation of the measured inaccuracy at each testing point. The total average is the average of all averages, median is the median of all the medians, and the total standard deviation is the average of all the standard deviations.

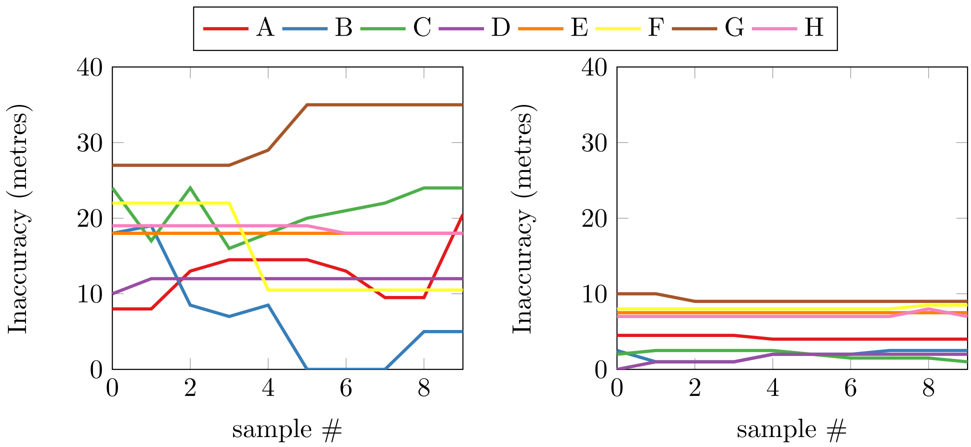
Table 4.1 shows in numbers what was previously discussed concerning figure 4.6. Exactly as could be seen previously, location G struggles quite badly with its accuracy in the medium density beacon setup. It can also be seen that the accuracy becomes slightly better in the high beacon density setup, and that the standard deviation isn't 0. This would imply that at least part of the issue in the medium density setup is a lack of Wi-Fi reception, causing the BlueDot to be unable to move. Although, it's still quite far off so the fingerprinting getting it wrong is also a strong possibility.

Locations like A, B & H performed quite well in all 3 tests, but didn't always improve as the beacon density increased. Notably B got worse every time the beacon density increased. This could be due to bad luck when testing because as detailed in chapter 4.1, RSSI is not very reliable especially indoors.

Overall, the total results show that adding beacons and setting up an Indoor Positioning System does improve the accuracy, however increasing the density of beacons didn't increase the accuracy of the results substantially. While the total average inaccuracy of the high beacon density setup, is lower than the average inaccuracy of the medium beacon density setup, the median actually increased a bit.

4.2.2 Indoor accuracy results on iOS between no beacons and high density beacon placement

Let us now consider the results on iOS, it should be noted that this sub chapter will not go through medium beacon density results, because this is here to show the IPS working on both Android and iOS. Once again we will start with the raw location measurements which can be seen in figure 4.7 below:



(a) No beacons (b) High beacon Density

Figure 4.7: 2 figures showing the inaccuracy measured at each testing point for 10 seconds, where 1 sample represents 1 second. Note: The same legend applies to both figures.

In figure 4.7 it can clearly be seen that there is a big improvement between the tests where there was no IPS, and the tests where there was. There's a clear improvement across the board, although testing at point B with no beacons did manage to get our location exactly right, but it unfortunately started getting it wrong again after a few seconds. This on its own shows that the IPS worked on both Android and iOS. They didn't get the same results, however they didn't have equal hardware conditions either.

If we put the numbers from figure 4.7, and put them in a table we end up with table 4.2. Again, the values have been rounded up to the nearest half a metre due to inaccuracies of the testing methods.

point	$\bar{x}$	$\tilde{x}$	s
A	12.5	13	4
B	7	6	7
C	21	21.5	3
D	12	12	0.5
E	18	18	0
F	15.1	10.5	6
G	31	32	4
H	18.5	19	0.5
total	17	15.5	3

(a) No beacons

point	$\bar{x}$	$\tilde{x}$	s
A	4	4	0.5
B	2	2	0.5
C	2	2	0.5
D	1.5	2	1
E	7.5	7.5	0
F	8	8	0
G	9	9	0.5
H	7	7	0.5
total	5	5.5	0.5

(b) High beacon density

Table 4.2: 2 tables showing the mean, median and standard deviation of the measured inaccuracy at each testing point. The totals are calculated in the same way as in table 4.1

Table 4.2 shows the same general theme as figure 4.7. Across the board there is a big improvement between no beacons and the high beacon density setup. The average inaccuracy is quite low, and overall would make navigating an indoor environment only using the map on your phone quite viable.

5 Conclusions & Recommendations

5.1 Conclusion

5.1.1 Research question 1

In the beginning of the thesis, we asked if making your own DIY beacon is viable versus buying a commercial ready made solution. In chapter 4.1.3 we said that in terms of signal performance, they perform more or less identical over all, which would suggest the answer is yes. But there's a lot more to the beacons than just how they performed in this measurement.

Another thing to keep in mind is their relative battery performance. The iBKS 105 as previously mentioned, has an estimated battery life of 38 months with the settings we used for testing. The ESP32 has an estimated battery life of 15 days[17]. While these batteries are rechargeable, that is still relatively abysmal. The only way to make the ESP32s a viable option versus commercial beacons, would be to wire them to permanent electricity, but this adds a lot of cost to the equation.

Appearance while subjective is another factor. Commercial beacons almost always come in a nice plastic housing with some adhesive to stick them to surfaces, or alternatively with screw holes to screw them into things. On the ESP32 side of things, as previously mentioned we had to put ours in the little plastic bag it came with. This looked so bad with our taping to the wall solution, that while we were setting up the beacon from figure 3.4b, we had somebody walk by and ask if we were installing a bomb. Obviously this was a joke, but it gives an accurate description of what these things looked like. The ESP32's would absolutely need some proper plastic housing for them to sit in, and this would add cost. Adding more cost would be an issue, because as things stand right now, the ESP32 costs about as much as the commercial beacon. Adding more cost would tip the scale further and further away from favouring the ESP32.

In conclusion, the ESP32 can be on par with a commercial solution, but economics of scale are a real thing. Overall, it doesn't make sense to make your own beacons in this way instead of buying a commercial ready made beacon.

5.1.2 Research question 2

The other question asked, was if the improvement of an IPS is worth it? While the improvement is there, as can be seen in Chapter 4.2, it was still far from perfect. The results from chapter 4.2.2 are relatively closer to what we wanted, but clearly these results didn't happen on both platforms. This could have been because of bad luck, or it could have been due to the age of the Android phone used. More testing would have to be done with a wider range of devices to know for sure.

Overall, using a medium beacon density setup or maybe even a lower density setup, and fingerprinting the environment would be worth it, but not always everywhere. The 4th floor of the KE building has a relatively simple layout, so it's very unlikely somebody would get lost walking around there. While an IPS makes navigating around a bit easier, the layout is so simple that we would say it's not necessary there. Setting up the IPS in an area with a more complex layout would be much more worth it.

This does not mean we recommend against setting up IPS in a place like the 4th floor of the KE building. We suggest that other places in the university should be prioritised first, because it would probably make more sense in other places like the 1st floor of the KE building, which has a more difficult layout to navigate.

5.2 Future Work

The results from this thesis could be expanded or improved upon given more time and/or money. Some potential improvements would have been to source a stand for the beacon and phone to sit in when doing the RSSI testing. This would guarantee a more consistent distance, and that we don't end up in a situation where we 'hold it wrong' between tests, and accidentally affect the results. Additionally, testing more distances between multiple different units of the same beacon would be good way to check the validity of the results.

For the IPS side of things, an interesting potential topic to look more into is how the accuracy changes based on beacon density. Could also do the test with more phones. In this case only 1 phone was tested per platform, and the phones were drastically different. Therefore a platform comparison was not possible, however a platform comparison would be interesting to do. It would just require more phones, which could be compared more fairly if they have a similar age and price point.

We would have liked to implement our IPS to a mapping service like MazeMap, Apple Maps or Google Maps. Unfortunately, due to limited time on this project we were unable to complete this goal. For the future we would like to further develop our IPS and continue testing, as our initial test results looked promising. With a fully integrated IPS at the University of Stavanger, it would not only help students and faculty navigate their way around campus, but could also be used during emergencies. In the event of a fire the IPS could help track and locate people inside the buildings to make sure everyone has been evacuated.

Developer documentation

[1] *accuracy — Apple Developer Documentation.*
URL: https://developer.apple.com/documentation/corelocation/clbeacon/1621551-accuracy.
(last accessed: 13.05.2021).

[2] *Adding a Floor Plan.*
URL: https://indooratlas.freshdesk.com/support/solutions/articles/36000050481-adding-a-floor-plan.
(last accessed: 13.05.2021).

[3] *AltBeacon Protocol Specification v1.0.*
URL: https://github.com/AltBeacon/spec.
(last accessed: 16.02.2021).

[4] *ARCore supported devices.*
URL: https://developers.google.com/ar/devices.
(last accessed: 10.05.2021).

[9] *CLBeaconIdentityConstraint — Apple Developer Documentation.*
URL: https://developer.apple.com/documentation/corelocation/clbeaconidentityconstraint.
(last accessed: 13.05.2021).

[10] *CLBeaconRegion — Apple Developer Documentation.*
URL: https://developer.apple.com/documentation/corelocation/clbeaconregion.
(last accessed: 13.05.2021).

[11] *CLProximity — Apple Developer Documentation.*
URL: https://developer.apple.com/documentation/corelocation/clproximity.
(last accessed: 13.05.2021).

[12] *Creating a Venue (Location).*
URL: https://indooratlas.freshdesk.com/support/solutions/articles/36000050478-creating-a-venue-location-.
(last accessed: 13.05.2021).

[13] *Eddystone format.*
URL: https://developers.google.com/beacons/eddystone.
(last accessed: 20.01.2021).

[14] *Eddystone protocol specification.*
URL: https://github.com/google/eddystone.
(last accessed: 20.01.2021).

[20] *iBeacon - Apple Developer.*
URL: https://developer.apple.com/ibeacon/.
(last accessed: 20.01.2021).

Hardware specifications & datasheets

[6] *BlueBeacon Maxi.*
URL: https://www.blueupbeacons.com/docs/BlueBeaconMaxi_DS_Ver3_2020.pdf.
(last accessed: 10.02.2021).

[19] *Huawei Mate 10 Lite - Full Phone Specifications.*
URL: https://www.gsmarena.com/huawei_mate_10_lite-8857.php.
(last accessed: 16.04.2021).

[21] *iBKS Plus Datasheet.*
URL: https://accent-systems.com/wp-content/uploads/iBKS_Plus_datasheet_rev6.pdf.
(last accessed: 10.02.2021).

[22] *iBKS105 Datasheet.*
URL: https://accent-systems.com/wp-content/uploads/iBKS105_datasheet_rev3.pdf.
(last accessed: 10.02.2021).

[25] *iPhone 12 mini - Full Phone Specifications.*
URL: https://www.gsmarena.com/apple_iphone_12_mini-10510.php.
(last accessed: 16.04.2021).

[30] *Samsung Galaxy S8 - Full Phone Specifications.*
URL: https://www.gsmarena.com/samsung_galaxy_s8-8161.php.
(last accessed: 10.05.2021).

[33] *SparkFun ESP32 Thing.*
URL: https://www.sparkfun.com/products/13907.
(last accessed: 07.04.2021).

The rest of the bibliography

[5] Austin Blackstone.
Understanding the different types of BLE Beacons.
URL: https://os.mbed.com/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon.
(last accessed: 16.02.2021).

[7] *Bluetooth tracking and COVID-19: A tech primer.*
URL: https://privacyinternational.org/explainer/3536/bluetooth-tracking-and-covid-19-tech-primer.
(last accessed: 19.02.2021).

[8] W. Bulten, A. C. V. Rossum, and W. F. G. Haselager.
"Human SLAM, Indoor Localisation of Devices and Users".
In: *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI).*
Apr. 2016,
Pp. 211–222.
DOI: 10.1109/IoTDI.2015.19.

[15] Vincent Gao.
Proximity and RSSI.
URL: https://www.bluetooth.com/blog/proximity-and-rssi/.
(last accessed: 20.01.2021).

[16] GoTekt.
Difference Between Eddystone and iBeacon.
URL: https://medium.com/@teksuntracker/difference-between-eddystone-and-ibeacon-f5f959484332.
(last accessed: 11.02.2021).

[17] *Guide to Reduce the ESP32 Power Consumption by 95%.*
URL: https://diyi0t.com/reduce-the-esp32-power-consumption/.
(last accessed: 14.05.2021).

[18] Dave Hollander.
How AoA & AoD Changed the Direction of Bluetooth Location Services.
URL: https://www.bluetooth.com/blog/new-aoa-aod-bluetooth-capabilities/.
(last accessed: 20.01.2021).

[23] infsoft.
Indoor Positioning With Ultra-Wideband.
URL: https://www.infsoft.com/technology/positioning-technologies/ultra-wideband.
(last accessed: 12.05.2021).

[24] Inpixon.

Bluetooth Location Tracking & Positioning.
URL: https://www.inpixon.com/technology/standards/bluetooth-low-energy.
(last accessed: 19.02.2021).

[26] U.S. Department of Justice.
The United States Department of Justice.
URL: https://www.justice.gov/opa/pr/justice-department-announces-actions-disrupt-advanced-persistent-threat-28-botnet-infected.
(last accessed: 01.03.2021).

[27] Finn Jarle Kvalheim.
Den nye Smittestopp-appen er langt mer tillitvekkende.
URL: https://www.tek.no/nyheter/guide/i/1EMB4L/den-nye-smittestopp-appen-er-langt-mer-tillitvekkende.
(last accessed: 19.02.2021).

[28] *Little endian and Big endian Concept with programming example.*
URL: https://aticleworld.com/little-and-big-endian-importance/.
(last accessed: 11.05.2021).

[29] Brian O'Keefe.
In: *Fiding Location with Time of Arrival and Time difference of Arrival techniques.*
2017.
URL: https://sites.tufts.edu/eeseniordesignhandbook/files/2017/05/FireBrick_OKeefe_F1.pdf.

[31] Maxim Shchekotov.
"Indoor Localization Method Based on Wi-Fi Trilateration Technique".
In:
2014.

[32] Beom-Ju Shin et al.
"Indoor WiFi positioning system for Android-based smartphone".
In: *2010 International Conference on Information and Communication Technology Convergence (ICTC).*
2010,
Pp. 319–320.
DOI: 10.1109/ICTC.2010.5674691.

[34] *Ultra Wideband information.*
URL: https://support.apple.com/en-gb/guide/iphone/iph771fd0aad/14.0/ios/14.0.
(last accessed: 13.05.2021).

[35] *Understanding RSSI.*
URL: https://www.metageek.com/training/resources/understanding-rssi.html.
(last accessed: 17.03.2021).

[36] Martin Woolley.
Bluetooth Direction Finding - A technical Overview.
URL: https://www.bluetooth.com/bluetooth-resources/bluetooth-direction-finding/.
(Version: 1.0.2).

Appendix A

Source Code

The source code for the RSSI measuring app and text files with the results of our measurements can be found in 2 compressed files called $data.7z$ and $iBeacondistancemeasure.7z$ additionally the ESP32's code to make it work as an iBeacon can be found in $Sparkfun\_esp32\_iBeacon.7z$

iBeacondistancemeasure.7z

data.7z

sparkfun_esp32_iBeacon.7z