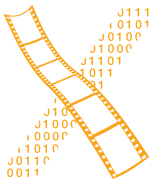

Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

# *Improving Indoor Location Awareness using Bluetooth Low Energy Beacons*

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

*by*
*Janosch Hübner*

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Ulrik Schroeder

Registration date: 11.02.2022
Submission date: 07.06.2022

Zentrales Prüfungsamt/Central Examination Office

# Eidesstattliche Versicherung
## Statutory Declaration in Lieu of an Oath

Name, Vorname/Last Name, First Name

Matrikelnummer (freiwillige Angabe)
Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel
I hereby declare in lieu of an oath that I have completed the present paper/Bachelor thesis/Master thesis* entitled

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Ort, Datum/City, Date

Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

**Belehrung:**
**Official Notification:**

**§ 156 StGB: Falsche Versicherung an Eides Statt**
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.
**Para. 156 StGB (German Criminal Code): False Statutory Declarations**
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.
**§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt**
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.
**Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence**
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Ort, Datum/City, Date

Unterschrift/Signature

# Contents

# Abstract

This work aims to improve indoor location awareness using Bluetooth Low Energy (BLE) beacons by providing a framework for developers written in Swift, a programming language created by Apple. Prior to the framework being introduced and discussed, a general overview over local positioning techniques, including advantages and disadvantages, is given. Furthermore, two frameworks provided by Apple are covered, detailing how they can be and were used for the task at hand. The following chapters will also state relevant thought processes and decisions made during the creation of this work. The entire framework and findings of this thesis have been conducted at the i10 chair of the RWTH university using an array of about 20 Bluetooth Low Energy beacons similar to the ones deployed in the Centre Charlemagne museum in Aachen, Germany. Although frameworks should provide tools to developers without assuming their specific use cases, the beforementioned museum acted as a concrete example to solve. The museum was visited once for testing purposes in a real life scenario. This paper shows that it still is a tricky undertaking in 2022 to work with Bluetooth Low Energy beacons, especially on closed systems like most smartphones these days. Given the wide availability of smartphones today, it makes sense to use them as a platform, however, it also means limited access to the system for less control over data. Results of this work, such as the Swift framework, aim to support most applications with Bluetooth Low Energy beacons.

# Überblick

Diese Arbeit beschäftigt sich damit, das Lokalisieren von Geräten in Innenräumen mit Hilfe von Bluetooth Low Energy Beacons zu verbessern und eine Bibliothek für Entwickler in der Programmiersprache Swift bereitzustellen. Bevor diese Bibliothek im Detail vorgestellt wird, gibt es einen kurzen Überblick über die verschiedenen Möglichkeiten der lokalen Lokalisierung und deren Vor- und Nachteile. Des weiteren werden zwei von Apple herausgebrachte Bibliotheken und deren Nutzen für diese Arbeit besprochen. In den nächsten Kapiteln werden wichtige Denkprozesse und Entscheidungen, welche sich während der Entwicklung dieser Arbeit herauskristallisiert haben, erklärt. Die meiste Arbeit wurde am i10 Lehrstuhl der RWTH Universität vollzogen. Dabei wurden circa 20 Beacons verwendet, welche sehr ähnlich zu denjenigen sind, die im Centre Charlemagne Museum in Aachen verbaut sind. Obwohl eine Bibliothek meist allgemein gehalten werden soll und nicht versuchen sollte, die Anwendung des jeweiligen Entwicklers einzuschränken, wurde diese Bibliothek anhand des Beispiels Centre Charlemagne entworfen. Dementsprechend wurde einmal im Museum getestet. Durch die Arbeit mit Bluetooth Low Energy Beacons wird klar, dass es immer noch kompliziert ist mit diesen in Innenräumen zu lokalisieren, besonders auf geschlossenen Systemen, wie Smartphones. Es ist jedoch sinnvoll, Smartphones einzusetzen, da diese heutzutage sehr weit verbreitet sind, aber es limitiert eben auch die Möglichkeiten, in das System aktiv einzugreifen.

# Conventions

Throughout this thesis we use the following conventions.

- If not specified otherwise, text convention is the default.
- `Source Code, Class or Framework Names` are written in monospaced typewriter-style text.
- BLE beacons, Bluetooth Low Energy beacons, beacons, iBeacons all describe the same devices unless stated otherwise.

# Abbreviations

**Table 1:** Abbreviations.

| Abbreviation | Meaning |
|---|---|
| App | Application |
| BLE | Bluetooth Low Energy |
| GPS | Global Positioning System |
| NFC | Near Field Communication |
| RFID | Radio-frequency identification |
| RSSI | Received Signal Strength Indicator |
| RWTH | Rheinisch-Westfälische Technische Hochschule |
| SVM | Support-Vector Machine |
| UI | User Interface |

# List of Figures

# Chapter 1

# Introduction

Most people know about Google Maps, Apple Maps or even Open Street Map. Maps, as a tool for navigation and understanding the surrounding area, have been around for a very long time[1]. They were presumably most important at sea in order to find the destination or to get home and they were based on someone else having made the same journey before, without a map. Positioning on those maps was rather difficult and tools had to help here, such as a sextant which was used to measure the angle between an object in the sky and the horizon. Not two decades ago, drivers were reliant on large foldable paper maps and had to stop, read road signs and find their own position to be able to tell where to go next. At this time, the Global Positioning System (GPS) was mainly used by the military[2]; and by some individuals for cartography or other civilian applications. It was not meant for road navigation or turn by turn directions and digital maps were still in the early stages. When smartphones started growing in popularity and it became affordable to add GPS chips to them, chances of getting lost were significantly reduced. Everyone who owned a smartphone was suddenly able to effortlessly find their own position with a small error of a few meters.

Maps are important when navigating and many people have one in their pocket these days.

[1] See Brown [1979]

[2] See https://www.scientificamerican.com/article/gps-and-the-world-s-first-space-war/

Maps have much improved and getting lost is difficult these days.

Throughout the years, digital outdoor maps have improved significantly by adding more details, shops, traffic status, smart routing, 3D buildings etc. Most importantly, digital maps are always up to date or at least could be updated without purchasing a new map. Navigating and positioning has never been easier, however this only applies for GPS and outdoor tracking. The next sections will have a look at indoor navigation and positioning.

## 1.1 Motivation

GPS does not work indoors and orienting in large buildings can be difficult.

Locating a device and therefore its user works quite well outside using GPS. However, entering a building paints an entirely different picture. Not only are there no indoor maps widely available; in contrast to Apple Maps, Google Maps, Open Street Map, etc. for outdoors; but furthermore GPS becomes quite inaccurate and does not update position for smaller movements within a building as the circle indicating the potential position is growing larger with the connection to GPS satellites getting weaker. There are a few papers proposing data structures for indoor maps and there is Apple and Google strategically mapping a few buildings for their map apps[3], but other than that, indoor maps are unique in every location and thus do not serve a common purpose resulting in very few indoor maps. Nonetheless, even if indoor maps were as good as outdoor maps, GPS would not be able to help much as it works best in direct line of sight to satellites. Thus orienting becomes more difficult, especially in larger indoor buildings such as offices, airports and train stations. Improving awareness, for where the user is indoors, may help improve orientation and may avoid getting lost. As we will see in chapter 3: Related Work, enabling indoor navigation can also help visually impaired people travel safer on their own and it can also make a visit to museums more fun for children.

[3] See https://developer.apple.com/documentation/corelocation/clfloor

[4] See https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf

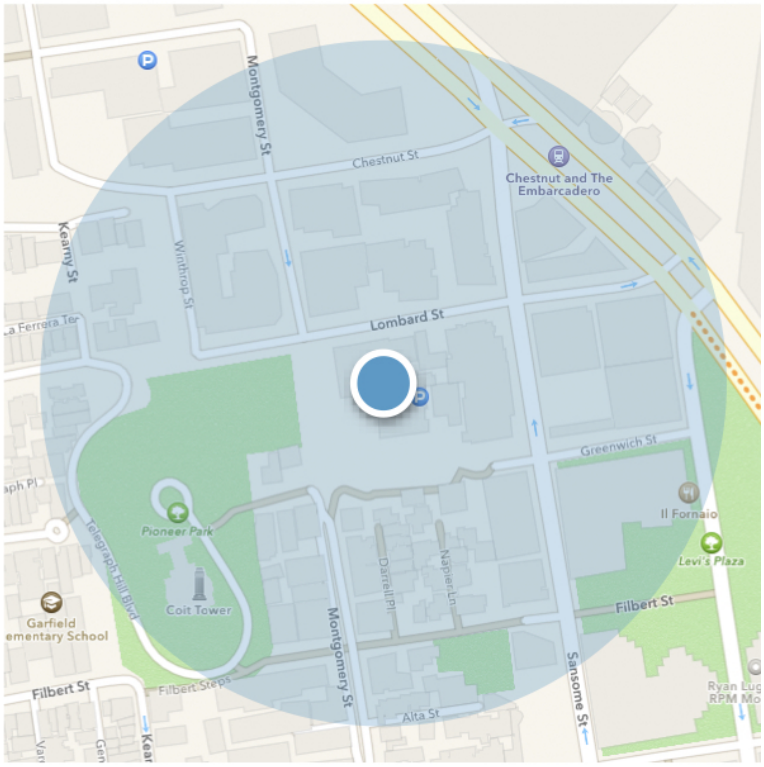


**Figure 1.1:** Device indoors, low accuracy depicted by large blue circle. The device may be anywhere in the circle. Image taken from Apple[4].

## 1.2 Local Positioning Systems

GPS works best being in direct line of sight of at least four satellites[5]. The same principle applies to most local positioning systems, such as WiFi and Bluetooth Low Energy beacons. RFID and NFC work differently and therefore they do not compare to WiFi and BLE, but can also be used for some sort of positioning. Unlike GPS, local positioning systems are, as the name implies, local and have to be set up individually for each location, which explains that they are not widely used.

[5]See https://www.gps.gov/technical/ps/2008-WAAS-performance-standard.pdf

The difference between active and passive communication is the way data is transmitted, always or on request or interaction.

Before diving into the techniques, it is important to understand the difference between active and passive communication and positioning. A device communicates actively, if it sends out a signal and thus data on its own for others to receive, such as Bluetooth Low Energy beacons. Passive communication is when a device sends out data on request, e.g a credit card transmitting information to the payment terminal when held close. The credit card does not permanently broadcast its information. Now active and passive positioning can be explained using active and passive communication. Active positioning is based on devices such as BLE beacons or WiFi access points that broadcast information actively and passive positioning requires the user to initiate the communication, such as scanning an NFC tag. The user in this case is the opposite to the positioning system. If it is active, the user does not have to do anything and if it is passive, the user has to actively participate.

### 1.2.1 WiFi and Bluetooth Low Energy Beacons

WiFi is more costly and not meant for locating a user out of the box.

WiFi itself is not meant to be used for positioning and often times, even if stores, shops and other buildings offer public WiFi, the setup is not sufficient for calculating a position. WiFi requires multiple access points for determining more than just the presence of a device[6]. The advantage compared to NFC or Bluetooth Low Energy beacons is the signal strength that WiFi has, which allows for a more solid setup. On the other hand, it also means if a WiFi signal is received that the range of possible locations is much bigger because it may go through walls and other obstacles. Another disadvantage is the fact that setting up WiFi for positioning can become quite costly. This is also covered in Related Work (3).

BLE is energy efficient and beacon hardware is preconfigured.

Bluetooth Low Energy is a standard originally developed by Nokia and incorporated into Bluetooth 4.0 in 2009[7]. Its goal was to reduce energy consumption, especially impor-

[6] See Yang and Shao [2015]

[7] See https://web.archive.org/web/20080907170808/http://www.electronicsweekly.com/Articles/2007/06/12/41582/wibree-becomes-ulp-bluetooth.htm

tant in battery powered devices such as smartphones, while retaining the same range as with regular Bluetooth. Some beacons may need to be powered for years on battery without maintenance and therefore BLE was chosen for them. BLE beacons are a bit more expensive than NFC tags, but cheaper than a similar WiFi setup. More on how they work can be found in chapter 2. While RFID and NFC are mostly passive, meaning that they require active input from a user, WiFi and BLE beacons are actively sending data which can be received anytime. BLE beacons communicate one-way only and they do not have any storage. Therefore, the hardware is configured once by the manufacturer and then sent out.

Similarities between WiFi and BLE are that both are active communicators, unlike RFID and NFC. Both need several devices with static locations in order to calculate a position using triangulation. Moreover, both are susceptible to fluctuation as the signals are affected by obstacles and people. Additionally both systems can be tweaked to increase the signal strength for a wider range or for specific applications.

### 1.2.2 RFID and NFC

RFID is mainly used for credit cards, to track inventory and other nearby wireless communication like car keys. RFID is susceptible to data theft within a few meters.

RFID stands for Radio-Frequency IDentification. RFID tags are often used to track inventory in stores[8], which also helps protect against theft but given their wide range of up to 100m, they can also be used for example to track luggage through an airport. There are three frequency levels for RFID: Low (LF), High (HF), Ultra High (UFH). Only the latter allows for up to 100m, the former ones are for 30cm and below. A quick look online shows that RFID-blocking wallets and card sleeves can be purchased in order to protect against theft. This kind of theft works by someone standing nearby, reading the RFID data of a credit card and on the next purchase made by the thief, that data is replayed to the payment terminal. Essentially a copy of the particular credit card is made. The problem does not stop with credit cards, as most passports in the United States have an

[8] See Weinstein [2005]

RFID enabled chip inside and thus are vulnerable to identity theft.

NFC is based on RFID with up to 30cm in distance and used for payments. Tags can store data which can be read by any NFC reader. They are less popular than QR codes as they do not have a unified look.

Near-Field-Communication, short NFC, is a technology mainly used for contactless payments in stores[9]. It evolved from RFID and operates on the high frequency (HF) level which means up to 30cm in distance. NFC is passive in a sense that most NFC chips do not have a power source and are not actively receiving or sending out data. NFC chips on credit cards for example are activated and powered by the respective payment terminal. NFC tags are inexpensive, small devices or even stickers that can store different types of data, such as a website link, text, and other data with a specified format such as a location as coordinates. Once configured, this data can be read by any NFC reader coming in close proximity to the tag and it can be reconfigured anytime using an NFC writer application. Most smartphones in 2022 have an NFC reader and writer equipped allowing for modern applications such as NFC light switches or scanning WiFi passwords for guests. Often times however, people are more used to QR codes for such tasks like joining a WiFi network or opening a digital card in a restaurant because the visual identification of a QR code is much more popular than that of an NFC tag, because those do not have a unified look. They also do not need a unified look because unlike QR codes, NFC tags are not scanned visually using a camera, but by moving a reader close to the tag. Suppose we are in a museum and due to its low pricing[10] we placed hundreds of NFC tags around, then locating the user of a smartphone would work by encoding the location of the tag in itself via a URL that opens our app with the location information, which would then be able update the position on a map. This is considered passive positioning as the location only updates when the user scans another tag. Scanning a tag could combine positioning information with information of the current exhibit closest to that tag.

[9] See Brown and Diakos [2011]

[10] about 50 cents/piece and cheaper when buying more

Positioning with RFID or NFC results in infrequent location updates.

Unlike with WiFi and BLE, using NFC tags for example does not require complex analysis of low level signals and calculations of position based on them. They are also not prone to signal fluctuation or obstacles as the scanning distance is less than 30cm. One scan reveals the exact position, however every other position in between scans is completely unknown, resolving in infrequent location updates. It does not help guide a user from one place to another, it only gives context to where the user currently is, much like signs and maps in a zoo or other large areas indicating the current position with a red dot. Next to WiFi, BLE, RFID and NFC there is also a modern (indoor) tracking solution by Apple.

### 1.2.3 AirTags

AirTags are small, closed devices for finding belongings and are not ready for indoor tracking but the technology is promising.

AirTags are Apple's take on trackable devices in order to solve the problem of loosing or misplacing belongings. Those were first released to the public in 2021 and can be attached to bags, bikes, keys etc. and their location is reported to the owner when any iPhone is nearby. Once the user is in close proximity to the AirTag, a different mode can be activated getting directions and a distance estimation. Powered by an ultra wide-band chip, AirTags are energy efficient with a high bandwidth creating a stronger signal than Bluetooth Low Energy beacons[12]. At the time of writing this paper, both the ultra wide-band technology as well as the AirTags themselves are closed systems unavailable to developers and therefore not eligible as a local positioning system. Moreover, with 30 Euro per device, they are quite expensive compared to Bluetooth Low Energy beacons and not designed to act independently from an Apple user account. The technology, the accuracy and reliability of the precision finding mode might be relevant in the future if they are opened up like Bluetooth Low Energy.

[11] Image from https://support.apple.com/en-us/HT210967

[12] See https://www.techinsights.com/blog/apple-u1-tmka75-ultra-wideband-uwb-chip-analysis


**Figure 1.2:** Precision finding of an AirTag shows the direction as well as the distance precisely and adds vibrations the closer the user gets to the AirTag[11].

## 1.3 Context and setup of this work

Beacons have been set up at the i10 chair to simulate a real world scenario like the one in the Centre Charlemagne museum.

This work aims to solve the problem of indoor positioning for developers in general, however there is a specific use case attached to it. The Centre Charlemagne is a museum in the center of Aachen providing exhibits on the history of the city. There is already an app developed by the i10 chair at RWTH for this museum with past versions of the app having worked with Bluetooth Low Energy beacons and therefore the museum has built-in beacons invisible to the visitor. Previous versions used a support vector machine (SVM) trained with RSSI data which attempted to define the position algorithmically based on data readings and previously trained knowledge. While the project ran for a couple of years, it was only able to detect a small subset of areas and given that obstacles, visitors and other factors may influence the RSSI values, the trained SVM may have predicted incorrectly. This approach of pre-trained data is called Fingerprinting. Instead of that approach, attempts to reduce noise in the live data were made during this work. The goal is to build a framework that can extend the app, enabling indoor positioning for visitors of the museum. More on the framework comes in chapter 4. In order to simulate a setup similar to the museum, about twenty BLE beacons were placed in the i10 chair at each door and were used for testing. The production test in the museum was conducted as a case study in chapter 5.

# Chapter 2

# Bluetooth Low Energy Beacons

Bluetooth Low Energy beacons are small, efficient devices serving a single purpose namely broadcasting a bluetooth signal with a unique identification repeatedly. The counterpart to beacons are receivers who can identify and tell different beacons apart. Most smartphones can become a receiver as well as a beacon through an application installed on the device.

## 2.1 Beacons and their data structure

Beacons broadcast an identifier (`UUID`), `major` and `minor` value for identification.

A beacon nowadays broadcasts three important information. First and foremost every beacon needs a universally unique identifier, short `UUID`. However, this is misleading, because many different beacons can share the same `UUID`, so it actually is not unique per device. Typically, beacons that share the same geographical context, such as a large building or those who are from the same company usually have the same `UUID`. In order to further differentiate beacons from one another, there are two values named `major` and `minor`. Beacons that share a narrower geographical context such as a floor in a building, or are used for a certain application, may share the same `major` number. Last but

not least, beacons that share both the `UUID` and the `major` are differentiable by the `minor` value[1]. For example, all beacons used in this work have the `major` value 80 and the same `UUID`. Those three values are key to identifying a beacon out of potentially hundreds, however they are not the only information in the Bluetooth advertising packet. Two bytes are used for both `major` and `minor` and given that those bytes are interpreted unsigned, one can define up to 65535 values. This shows that there is no need to use different `UUID` or even different majors for an application unless separating the numbers brings additional information that may be useful, such as determining the office building on a campus, a floor in a building with many stories or identifying the correct store if used by a merchant. Sixteen bytes are used for the `UUID` which leaves ten bytes remaining, consisting of manufacturer data, metadata such as length and type of the data and other flags. The last byte of the packet is a constant set during the hardware creation process and it defines the received signal strength indicator (RSSI) at one meter distance, acting as a reference value to compare the measured RSSI with. Some beacons allow changing the broadcasting power flag, which is not same as the constant mentioned before. The higher the power, the wider the range, however if the beacons are battery powered, this has a negative affect on battery life. Another setting that can be changed, depending on the manufacturer, is the advertising interval, which defines how often an advertisement packet is sent out.

[1] See https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf

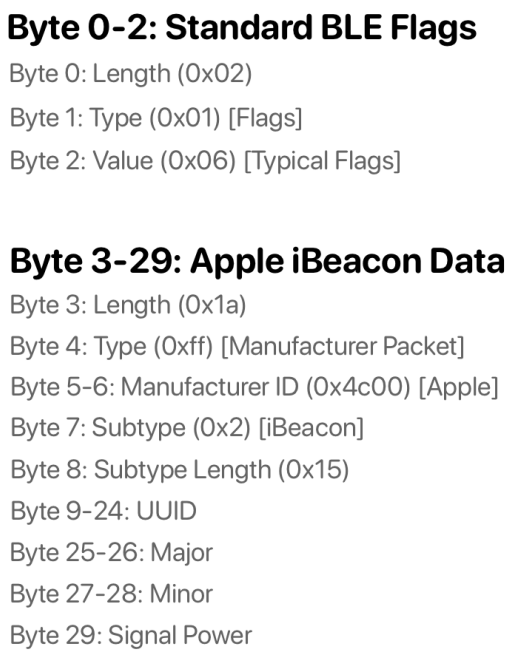


**Figure 2.1:** iBeacon Data Structure with explanation as to what the individual bytes 0 to 29 represent[2].

## 2.2 Working with Beacons

Typically beacons are used for proximity detection only.

After discussing what beacons are, how they work and the data they provide, it is time to look at how they can be used in an iOS application. Apple introduced the iBeacon protocol in 2013, creating an industry standard for how beacon manufacturers designed their hardware. This standard included all three of the aforementioned important data parts. They themselves make use of this technology in their stores, mainly to show reservations to customers when they arrive, their appointments on the smartphone or to welcome you.

[2]Data based on `http://www.warski.org/blog/2014/01/how-ibeacons-work/`

Typically beacons are used for exactly that: simple proximity detection by shops, businesses and other points of interest. A shop owner for example could place beacons near the entrance and detect those in their own shop app. Once detected, the app knows that the user must be close to the shop and can ping an advertisement or a discount to the smartphone. More on that can be found in Related Work (3).

### 2.2.1 CoreLocation and CoreBluetooth

In this work, the focus is on the receiving part of BLE beacons and not about the low level Bluetooth communication, which was already implemented by Apple in their `CoreLocation` and `CoreBluetooth` frameworks. The former was used in order to determine a position based on the received signal strength and the known, static location of the beacons.

`CoreLocation` allows developers to monitor beacon regions and range beacons in order to get the current distance to them.

`CoreLocation` is a framework by Apple which, as the name implies, allows for working with everything related to location and positioning on iOS[3]. This framework is considered closed source and therefore making improvements as well as understanding the internals is difficult. Some fixes are applied by writing wrapper types and classes in chapter 4. Ever since iOS 7, `CoreLocation` supports monitoring and ranging beacons. Monitoring beacons requires less power than ranging, but does not reveal proximity information. The recommended approach is to first monitor a beacon region and then range individual beacons. For example, a region could be placed over the entire museum and once the region is entered, ranging of beacons can start. This way an app is not consuming any power if used outside of the museum. A disadvantage of `CoreLocation` is that updates of beacons are only propagated once every second. For more control over receiving BLE packets, one could use `CoreBluetooth`, however packets are not automatically decoded, plus determining the distance does not come out of the box like proximity in `CoreLocation`.

[3] See `https://developer.apple.com/documentation/corelocation`

Using the `CLLocationManager`, the central class in `CoreLocation`, one can start monitoring a `CLBeaconRegion`, which takes a `CLBeaconIdentityConstraint` specifying a `UUID`, `major` and `minor`. Once every second, all beacons satisfying the constraint are passed back to the `CLLocationManager` delegate. Additionally or alternatively, if presence of beacons is not enough and proximity for them is required, the same class can range beacons.

`CoreBluetooth` can help receive the raw Bluetooth packets in order to take full control over the data after decoding.

`CoreBluetooth` is an iOS framework for working with Bluetooth on a very low level[4]. Using the `CBCentralManager` one can scan for peripherals and handle advertisement packets using the `CBCentralManagerDelegate`. The delegate, specifically the `didDiscover` method, is called with raw advertisement data, which can be decoded in many ways, as well as the RSSI. If the received peripheral allows active connections, one can connect to it in the receiver handler. According to documentation[5] on Advertisement Data Retrieval Keys provided by Apple, the advertisement data may include the key `CBAdvertisementDataTxPowerLevelKey`, which describes the transmission power if the peripheral supports it, which iBeacons do. The TxPower data was described above. Moreover, one can get the local name of the peripheral as well as the manufacturer and the information whether this peripheral can be connected to. Despite `CoreBluetooth` being a low level framework, which could be used to scan for Bluetooth Low Energy beacons, there is no indication that Apple's own framework `CoreLocation` is using it as a dependency to build upon that. In general, unfortunately, there is not a lot known about the internal functionality of both frameworks.

[4] See `https://developer.apple.com/documentation/corebluetooth`

[5] See `https://developer.apple.com/documentation/corebluetooth/cbcentralmanagerdelegate/advertisement_data_retrieval_keys`

# Chapter 3

# Related Work

Given the variety of use cases one can create using Bluetooth Low Energy beacons, it is not surprising that there are many different applications and prior research on them.

### Bluetooth Low Energy in Health Care

BLE played an essential role in contract tracking apps during the pandemic.

In 2020, when the Covid-19 pandemic hit, many countries started developing an application that would anonymously track nearby phones and if they were in close proximity for a certain amount of time, the phones would then exchange keys for the case that if one person reported sick, the other one would get notified. On Apple systems, the `Exposure Notification`[1] framework was built for that specific case. Apple also published Bluetooth specifications for their service[2], showing that essentially every smartphone acted both as a receiver and a beacon at the same time. Shortly after releasing the specifications, Hatke et al. [2020] detailed how to use Bluetooth Low Energy in order to detect proximity and compared different detection algorithms and their performance.

[1] See `https://developer.apple.com/documentation/exposurenotification`

[2] See `https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ExposureNotification-BluetoothSpecificationv1.2.pdf`

This is by far not the only application in a public health related field as Iqbal et al. [2018] study whether utilising Bluetooth Low Energy would help in tracking medical staff and equipment. Yang et al. [2015] supported patients or visitors of patients finding their departments. Also Lin et al. [2015] helped medical staff track their patients efficiently using Bluetooth Low Energy beacons.

### Applications in other fields

Given that this work focuses on implementation, it does not go into detail about signal strength, reception and other low level parts, however there are some interesting papers on that. For example, Bertuletti et al. [2016] analyse the relationship between RSSI and distance; and compare mathematical models to find out which one performs the best.

WiFi has a higher accuracy but is not trivial to set up.

In the first chapter, alternatives to Bluetooth Low Energy as local positioning techniques have been compared and one of them was WiFi. Chen et al. [2016] show that while BLE can only be as accurate as a few meters, using a WiFi setup they were able to get accuracy as close as a meter or below. However, setting up WiFi is not as straightforward as with BLE beacons which work out of the box and do not require any fingerprinting.

The related work named Ghosts aims to gamify museum visits by helping Ghosts that popup on the screen find their belonging exhibit.

In contrast to low level signal strength research, there are also many different smartphone applications, some of which are similar to this work. However, most implement a receiver application tailored to their use case while this work aims to generalise the problem by making a framework that can be used for many different use cases. One of those applications is Ghosts by Nilsson et al. [2016] which aims to gamify museum visits. Their research also shows that BLE signals are unstable, however they were able to use that to their advantage by incorporating it into the gameplay. The way it works is that when a beacon is detected, a ghost would pop up on the screen asking for help to be brought back to its belonging exhibit. This way, people would learn about certain exhibits while being on a mission to help a ghost get home.

The ghost would also provide commentary depending on how close the user was to the beacon, but it almost is binary as strong signal strength yields one text and weak signal strength yields another, given that otherwise the inaccuracy may result in wrong text being displayed and the user being confused. They also found out that the direction the user was facing had a direct impact on the RSSI values received and that larger crowds were able to block the signal entirely. Therefore, there is always a risk of showing incorrect data to the user.

GuideBeacon attempts to solve navigation for visually impaired people, especially when traveling.

Other than to making museum visits more fun and modern, Cheraghi et al. [2017] attempt to solve indoor way finding for the blind, the visually impaired and the disoriented with their app called GuideBeacon. Like with Ghosts, they also figured out that RSSI values are far from being reliable or consistent in one place. They note that large indoor areas may also be difficult to navigate for the sighted thus helping everyone feel more comfortable. Additionally, they conclude that due to everyone having a smartphone and the world becoming more digital, personell in large areas like an airport or a train station has decreased in numbers and when signs are not an option due to low vision, then there is a need for a low-cost, easy to use and reliable indoor way-finding system to serve blind or visually impaired people. In that paper, it is also discussed how navigation works for the blind or visually impaired, because instructions to e.g turn left are not as easy to follow if you can't see the compass pointing in the right direction. In order to solve that, they used a combination of text to speech and speech to text to allow anyone to easily use their system.

Proximity based advertising sends promotions to nearby users.

As mentioned before, beacons work best for proximity detection in shops and stores, like Apple does with their retail stores. Zaim and Bellafkih [2016] addressed this topic by proposing a system that upon customer detection sends information to a server which then may return promotional offers to the user.

### Maps

Maps are the most common and most useful display of position as it provides context of the environment.

Both Ghosts! and GuideBeacon use maps for displaying information such as position. It is the most common way to display position as it gives more context of the surrounding area. Wacker et al. [2016] evaluated map based audio guides versus regular audio guides in museums and concluded that it is beneficial to also use a map based approach, so that surrounding exhibits are discovered easier or at all. This shows that going with a map based approach is always a good idea, however given that this work features a framework, a map must not be assumed to always be the way of representing position.

### Lessons learned

Most, if not all, of the related work shows that BLE beacons are easy to get started with, that they do not cost a lot and that they can enable great applications. However, it is also clear that the received signal strength indicator can be very inaccurate due to many different and dynamic factors and thus deriving position from it and showing it to the user is more difficult to get right.

# Chapter 4

# Flair - A Swift Framework

This framework aims to provide beacon-based position information using `CoreLocation` beacon ranging. It supports iOS 9 and upwards with modern API, such as publishers from the `Combine` framework, available if the target application supports newer iOS versions. Many different features were implemented for the framework.

## 4.1 The Basics

### 4.1.1 CoreLocation improvements

`CoreLocation` is one of the oldest frameworks provided by Apple and has been around since iOS 2. Beacon support has been introduced in iOS 7 and with iOS 13 the framework has seen some changes in regards to ranging beacons. Even with the recent update, weaknesses remain in the usability of `CoreLocation`, which needed to be improved and are part of Flair. One downside is that with the introduced class `CLBeaconIdentityConstraint`[1] one may

Fixes were applied to allow detecting beacon minors without majors as well as a combination of different majors and minors.

[1] See `https://developer.apple.com/documentation/corelocation/clbeaconidentityconstraint`

specify a `major` value, without specifying a `minor`. This makes sense, given that one could monitor all beacons with a specific `major` value. It is, however, not possible to just monitor a set of `minor` values without setting a `major` value. The only way to monitor a set of beacons is if they share a `major`, which is a reasonable constraint, however it is not a necessary constraint, so an alternative approach was developed for this work. This approach listens to all beacons with the same `UUID` and does the filtering based on the given sets of majors or minors allowing for a wide range of beacons to be monitored at once.

### 4.1.2 Flair Beacons

Flair uses two types of beacons to differentiate between static and ranged beacons.

While CoreLocation has `CLBeacon`, Flair divides between two beacons namely `Flair.MapBeacon` and `Flair.RangedBeacon`. A map beacon defines its position within the region, `UUID`, optional `major` and `minor` and whether it is currently enabled or not. The position on a map can be relative to the map size or absolute, which makes sense for hard coded, non changing maps. Ranged beacons on the other hand represent beacons found by `CoreLocation`, therefore having an optional proximity, `rssi` value and an observation date. Beacons that were used for this work also provide their own battery percentage in a mixed value of `major` and battery percentage, therefore this value has been added as well. This distinction between a map beacon and a ranged beacon has been made, so beacons can be easily defined on a map without needing much information and then ranged beacons can be associated with the static map beacons once received.

### 4.1.3 Scanning Beacons

Picking up from `CoreLocation` and `CoreBluetooth` (2.2.1), scanning beacons using Flair can be done using the `Flair.BeaconProvider` class. In order to start ranging beacons, knowing the `UUID` for the beacons is necessary. Moreover, filters and transformers may be required and are

discussed in 4.3. As mentioned in chapter 4.1.1, Flair applies filters to all beacons with the same `UUID` enabling a more dynamic system than that of `CoreLocation`, as it supports recognising a mix of majors and minors.

### 4.1.4 The scale problem

The scale is used to translate pixels to meters in the real world.

One of the ongoing problems with applying beacon positioning to real world buildings is the scale of the map. It is important to understand the relation between pixels on screen and meters in the building, otherwise any display of data might be completely off. If a building has a hallway that is 15m long, and a map that is 1500 pixels (px) wide, then 100px relate to 1 meter. Given the fact that there are not many and not many good indoor maps, the scale has to be calculated for every building individually. This problem of finding the scale cannot be solved by software and therefore the scale has to be part of the input for the framework.

## 4.2 Shapes

The initial idea for the framework's main feature was to define areas on an indoor map with arbitrary shapes, allowing someone to define bounds for each area, define where the walls are and where valid space for walking is. Shapes have been considered, as they solves the following problem.

Having too few beacons in range creates a problem of not knowing on which side the user is.

Suppose only two beacons are received, then it is unclear on which side of the beacons the receiver actually is. However, in the image below, there is only one possible side, given that on the left side there is no walking area for visitors. Having that information allows for positioning in situations where the amount of beacons is not enough. Furthermore, the information of walls may help in determining the actual distance to beacons, given that walls block parts of signals. Beacons seem further away than they actually are, which with the information of walls can be fixed. However, shapes have been discarded for two reasons with the first one being that indoor maps are no longer the only

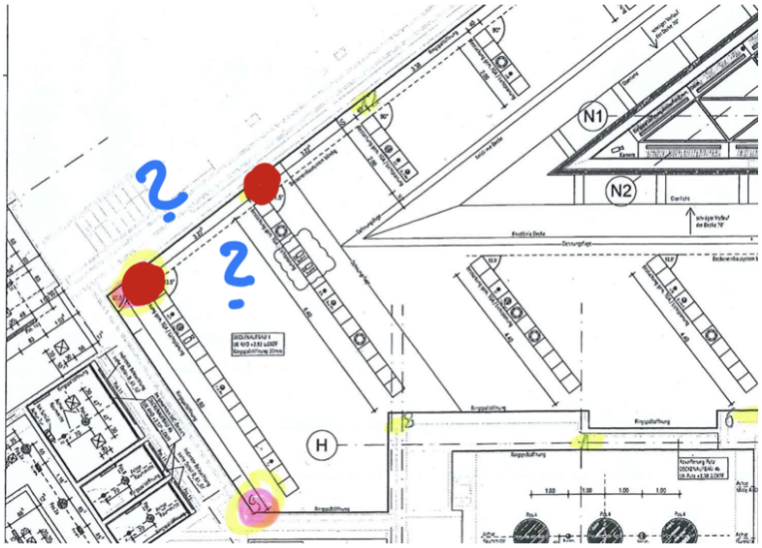


**Figure 4.1:** When only using two beacons, it is unclear on which side of the beacon the user is standing on.

requirement, but also information about the areas, shapes and walls need to be plugged in the framework and depending on the shapes, this may turn out to be a lot of preparation before being able to use the framework, which is not exactly plug and play. Additionally, the decision process of someone being in one area or the other is a binary decision, which means that if someone is on the edge between two areas, jumping may occur without any movement by the user. Instead of distinct areas, a more continuous approach was chosen which brings up the topic of Filters and Transformers.

Shapes were discarded as an idea due to prior setup required and the complexity of shape intersection.

Another decision for not pursuing the concept of shapes was the complexity of deriving a position-shape out of all the surrounding circles representing the beacon range. The problem starts with two circles overlapping each other, which results in a shape of common points. The problem growths in complexity the more circles there are overlapping or overshadowing each other and the reality is that the more beacons there are, the easier it should be to get a good location out of those information, which is contrary to the complexity of solving the path position problem. Interestingly, the only additional information received by solv-

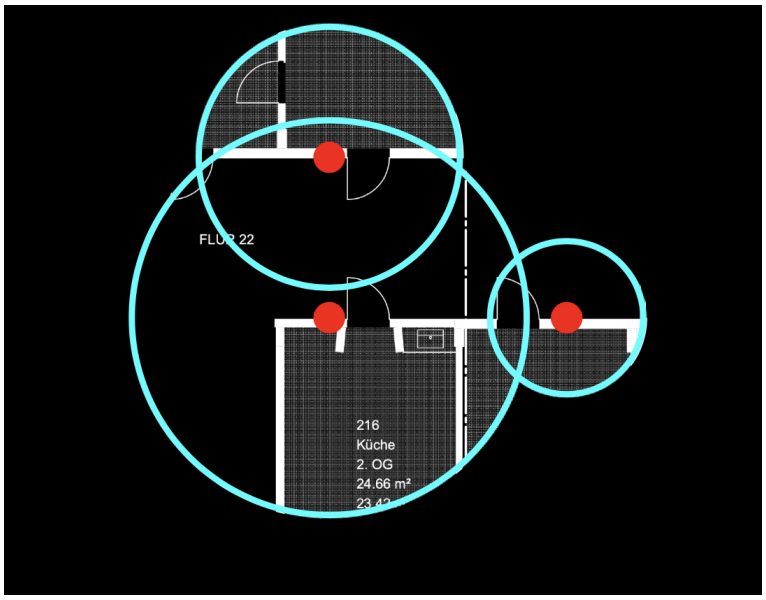


**Figure 4.2:** The problem of deriving potential positions by creating intersections of circles is easy visually but with an arbitrary amount of beacons it gets very complex algorithmically.

ing this is another axis. In both the i10 chair, as well as in the museum, the hallways tend to be longer but narrower, which means that Bluetooth Low Energy beacons aren't even accurate enough to determine wether someone is on the left side, in the middle or on the right side of a hallway. Knowing this led to the decision of approaching the 3D problem in a 2D space and making linear calculations. More on how the linear algorithms work and how the fallback for non-linear buildings looks like can be read in Positioning Algorithms (4.4).

## 4.3 Filters and Transformers

Flair provides three filters and two transformers out of the box and provides support for extending it with new filters or transformers by developers.

Filters are used to filter out unwanted beacons

One of the filters is the `Flair.RegionFilter`, which filters out all beacons that do not belong to the current beacon region. A beacon region is defined by a `UUID`, optionally a map including an image, size and a scale and beacon groups. Beacon groups themselves contain a list of `Flair.MapBeacon`, an optional name for reference and entire groups can also be disabled. This means developers have full control over individual beacons as well as beacon groups within a region and the `Flair.RegionFilter` makes sure that no disabled beacon or group is propagated back to the developer. The `Flair.BeaconConstraintFilter` can help filter out beacons that do not match a certain constraint of `UUID`, majors or minors. Last but not least, Flair provides a filter for narrowing down proximity of beacons, which means that one could monitor beacons that are within 10m of range for example. Filtering based on accuracy may result in loosing beacons or having beacons appear and disappear at every tick if they are around the range of the filter.

Transformers help shape the data and get rid of noise and other side affects.

Transformers are, as the name implies, used for modifying and transforming beacons. The reason behind transformers is that beacon signals are noisy and in order to make them more reliable, different techniques can be implemented to adjust or predict the closeness of a beacon. Beacons used in this work make use of their two byte `major` value by also writing the battery percentage in there, this means the value will always be different and changing, so we need to transform it in order to get the actual `major` value. This is a very specific use case and may not apply to other beacons, however the option is available if needed. In a more general application, Flair also implements the accuracy transformer. This transformer allows creating beacons based on a strategy like sliding window, which takes the last n readings and creates a new beacon out of it replacing the currently received beacon. In chapter 4.5.2 the Kalman-Filter is discussed which helps improve noisy data and therefore a transformer named `KalmanBeaconTransformer` has been created using external dependencies that should not reside within the Flair framework but can be passed to it using protocol based programming.

## 4.4 Positioning Algorithms

Flair provides both advanced features, extending on existing `CoreLocation` features, as well as positioning a device based on received beacons, which is done through a mixture of different algorithms.

### 4.4.1 Group Detection

Flair supports detecting entering or leaving group of beacons.

Before diving into positioning the user, Flair also supports proximity detection, which is the primary use for beacons. It extends `CoreLocation` by allowing an arbitrary amount of beacons to form a `Flair.BeaconGroup` and get updates on when a group was entered or left. A group is entered as soon as any of the enabled beacons within the group is detected and a group is left if no beacon of that entered group is detected anymore. This is similar to `CLBeaconRegion` of `CoreLocation`, which can be used to detect entrances and exits and once the user is within a region, the beacons can be ranged in order to get their current distance from the receiver. This, over ranging all beacons, is more efficient as ranging requires more energy.

### 4.4.2 Average Positioning

This algorithm frames beacons, finds the center and averages the radius.

The first of two positioning algorithms is simple averaging of the closest `n` beacons. For this, the closest beacon is the starting point and if there is no other, then this one is used. The idea is to draw a rectangle around all selected beacons in order to find the center point. Therefore, the top, left, bottom and right beacon based on the position of the closest one are taken into consideration but they must not be unique as the same beacon can both be e.g the left and top beacon. From those beacons, a position is calculated based on the average range. Suppose we had four distinct beacons $(b_1, ..., b_4)$ with distances $d_1 = 2\text{m}, d_2 = 4\text{m}, d_3 = 8\text{m}$ and $d_4 = 18\text{m}$. Then the average distance would be $32\text{m} \div 4 = 8\text{m}$.

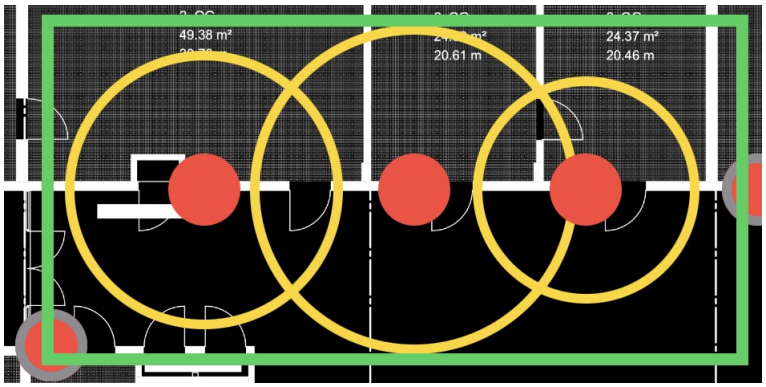


**Figure 4.3:** A rectangle (green) is spawned over the closest beacons (red) and their radius (yellow) in order to define the center position, which is the center of the rectangle.

### 4.4.3 Weighted Average Positioning

Weighted Average Positioning improves Average Positioning by giving more weight to smaller and thus more accurate beacons.

Simple averaging is quite unfair for beacons that are closer than others because the closer a beacon is, the more accurate is the distance, therefore the average algorithm has been tweaked to account for that. For this, the relative radius for each beacon is calculated with closer beacons having a smaller relative radius. Then, the relative radius is subtracted from 1 (`(1 - (beaconRadius / totalRadius))`), meaning the smaller the relative radius is, the closer is the beacon and the higher is the the weight of the beacon in the calculations. Picking up the example from above, with four beacons $(b_1, ..., b_4)$ and distances $d_1 = 2\text{m}, d_2 = 4\text{m}, d_3 = 8\text{m}$ and $d_4 = 18\text{m}$, gives us the following weights:

$$
\begin{aligned}
w_1 &= 1 - (2/32) = 0.9375 \\
w_2 &= 1 - (4/32) = 0.875 \\
w_3 &= 1 - (8/32) = 0.75 \\
w_4 &= 1 - (18/32) = 0.4375
\end{aligned}
$$

The average distance is now: $(w_1 * d_1 + w_2 * d_2 + w_3 * d_3 + w_4 * d_4)/4 = 4.8125m$ which is closer as it is giving more weight to closer beacons.

### 4.4.4 Linear Positioning

The x-coordinate is based on the beacons and the y-coordinate is based on the slope between two or more beacons.

In the beginning, the idea was to calculate both the x-axis and y-axis value based on surrounding beacons. However, with the noise in the data, the problem became quite complex, as mentioned in chapter 4.2. Most buildings and especially the chair and museum had large linear hallways, which could be modelled as a 2D problem, instead of a 3D problem. This allowed focusing on the x-axis, however the y-axis was still relevant, in case the path of beacons was not horizontal. The way this works is by finding a predecessor and successor beacon based on the closest beacon. From there the left most and right most position of the predecessor is calculated based on the position of the beacon and its range. The same is done with the successor. Next, the weight of both beacons is calculated, meaning that if the successor is closer, it will have a larger weight. In order to determine the correct y value, the slope between the predecessor and the successor is determined and if the slope is 0, we are in a horizontal scenario and we can just return the calculated x value and y being the y position of the beacons. Otherwise, if the slope is not 0, we calculate `y = mx + b`, which allows us to get the y-value based on our calculated x-value and the slope `m`.

#### Combined Positioning

Fortunately or unfortunately, not everything in the real world is linear. Therefore, this positioning strategy determines linearity based on the received beacons and applies linear positioning, if the beacons are ordered linearly, otherwise it falls back to average positioning.

## 4.5 Noise Reduction Algorithms

With every wireless communication, noise is a factor that can heavily affect the outcome. There are several options to reduce noise or to limit its effect and they can be applied on different levels of processed data and can also be combined.

### 4.5.1 Sliding Window

This sliding window algorithm takes the last n values and builds an average using them.

The first algorithm is called sliding window and it actually is a general algorithm and not specifically for noise reduction. It works by laying a window of size `n` over a sorted array of data, starting from index 0 to `n-1`. It then proceeds to shift the window one step at a time, so from 1 to `n` and 2 to `n+1` and so on. Flair makes use of this algorithm over the estimated distance values provided by `CoreLocation` and given that only the latest `n` values are relevant, the very last window of all windows over that data is used. This leads to the latest received value replacing the oldest one in the last window repeatedly. Every time the window changes with a new value being added, the average of those values is calculated and used. The size of the window can be configured by developers using the framework and it defaults to 5. There is one major disadvantage of this algorithm however, which is that all values weigh the same, for example if a small value drops by adding a large value, the average shifts by a lot and vice versa. Additionally, taking up to `n` measurements with one measurement a second can cause a delay of movement that may be visible to the user.

### 4.5.2 Kalman-Filter

Kalman-Filter reduces noise by gaining confidence with continuous measurements and predicts the next value.

The Kalman-Filter reduces gaussian noise in two phases, which is the measurement update phase as well as the prediction phase. It works similar to the Monte Carlo Localization[2] except for the fact that Kalman-Filter operate on con-

[2] See `https://ieeexplore.ieee.org/iel5/6243/16780/00772544.pdf`

tinuous data. Both algorithms are used a lot in self driving cars to predict the state of a system, such as other cars, the current position and distance to obstacles[3].

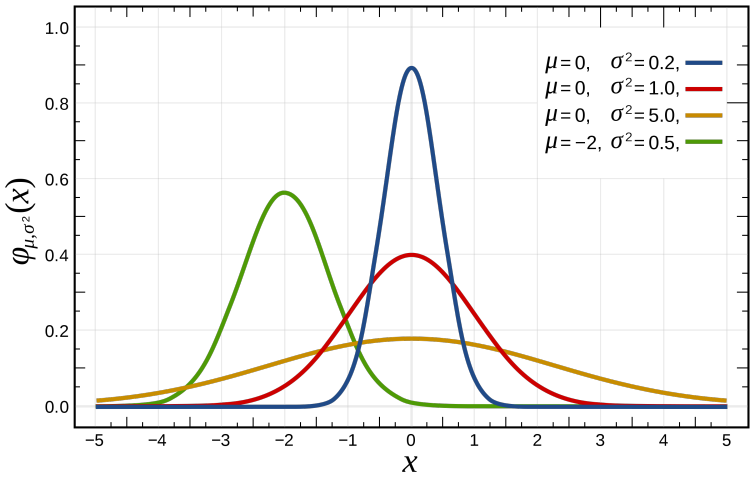


**Figure 4.4:** Gaussian curves are symmetric with a peak in the middle, the x position of the peak is called mean $\mu$. The width of the curve is defined by the variance $\sigma^2$. Image taken from Wikipedia.[4]

The framework only considers a 1D Kalman-Filter as values like the RSSI and the distance to beacons are one dimensional. The initial update phase, which happens when a new measurement is taken, updates the mean as well as the variance based on a fixed formula.

$$mean = (var_2 \times mean_1 + var_1 \times mean_2) \div (var_1 + var_2)$$
$$var = 1.0 \div (1.0 \div var_1 + 1.0 \div var_2)$$

The prediction phase takes a movement into account, which itself is represented as a gaussian curve. The mean and variance are updated again based on the movement's mean and variance.

$$mean = mean_1 + mean_2$$
$$var = var_1 + var_2$$

[3] See https://www.udacity.com/course/artificial-intelligence-for-robotics--cs373

[4] See https://en.wikipedia.org/wiki/Normal_distribution#/media/File:Normal_Distribution_PDF.svg

# Chapter 5

# Using the Framework - A case study

The goal of this framework is to be used in production. This section describes the process of integrating Flair in an existing application for the Centre Charlemagne museum. Integration tests are very important to notice the actual requirements developers have.

### 5.0.1 Adjusting to a new playing field

The difference between the i10 beacon setup and the actual museum scenario was immediately clear on the first visit.

Most of the framework was developed at the i10 chair as mentioned in Context and setup of this work (1.3), therefore some assumptions were made unconsciously based on that setup, a linear hallway with a door in the middle. For example, one of those assumptions was that beacons at the end of the list were not received as strong as closer ones. This, however, is not the case in a triangle, like the shape of the museum, because the last and the first beacon in the list are very close to each other physically. A visit of the museum revealed this issue immediately. Moreover, not all things are linear, a triangle has three linear sides, but when the lines connect at the corners, linearity can no longer be obtained with the beacons in that corner. For this case, Combined Positioning (4.4.4) has been introduced, which can detect linearity and depending on whether there is a

linear context or not, performs linear positioning, otherwise it averages the center and accuracy of the surrounding beacons. While the beacons were clearly visible at the i10 chair and in about 1.5m height for best reception when holding a phone, it was not exactly clear where the beacons were in the museum. This had an effect on the data because the closest that visitors could get to the beacons was 2-3 meters given that they were behind walls and exhibitions.

### 5.0.2 Integration of Flair into an application

Using Flair requires defining beacons by code or using a setup application, a scale and size of the area.

Due to the fact that Swift Packages are becoming more popular, Flair is also distributed as one. The advantages are that this is fully integrated and supported in Xcode, as well as still having backwards compatibility until iOS 9 and potentially earlier. This, however, removes the option to integrate Objective-C code into the bundle, which is still possible when designing Frameworks distributed as `.framework` or through CocoaPods. For this project it was not necessary to include Objective-C code, however the application that made use of Flair was able to include Objective-C code and still interact with Flair due to its protocol based approach.

There are two options to defining the static beacons, either programmatically or via a JSON file.

First and foremost the framework needs to know about beacons and the way they are set up in a region. Those are the static beacons, their identifiers and position. The position may be relative to the map size or absolute. Based on a list of built-in beacons in the Centre Charlemagne, a `Flair.BeaconRegion` is created. A beacon region can either be a map region or an arbitrary region with a size and a scale. In order to get the exact position of the beacons for the Centre Charlemagne, a setup app has been created which allows placing beacons on a map and exporting their positions and data. There are two possibilities to tell the framework about beacons: programmatically and using a structured file generated by the setup app. Part of that data is the scale of the map image to the physical location as discussed in chapter 4.1.4. The advantage of Flair is that every step of the process to getting a user location can be customised and fit to the individual use case.

Developers get positioning out of the box or they can take over by receiving beacons.

This means that developers can get the beacons through the framework including applying Filters and Transformers (4.3) or the user position directly based on a positioning strategy. Once a `Flair.UserPositionProvider` is allocated and initialised using a `Flair.BeaconRegion` and transform options, the linear, combined or average position can be received either via callbacks or using a `Combine.Publisher` depending on the application target and preference. Given that there are situations, such as missing beacons, in which the algorithms cannot determine a position, the callbacks may receive a `nil` value. `Flair.LinearPosition` is represented by a center position and a minimum and maximum x-value, so visually it is represented as a bar. On the other hand, the average position has a center point as well as a radius, which is the average range of the beacons nearby.

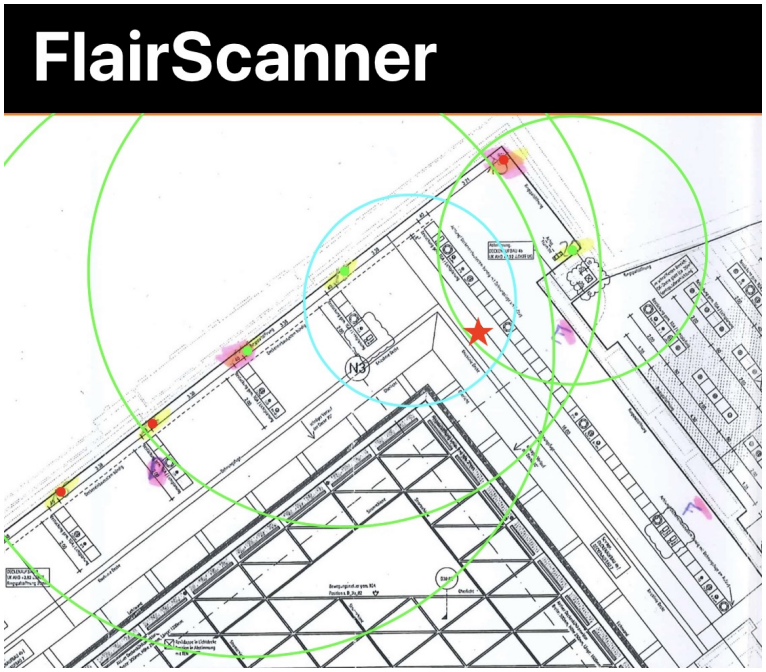


**Figure 5.1:** Average positioning in a corner of the triangle that is the Centre Charlemagne museum. The code automatically detected a non-linear alignment of the received beacons and decided to fallback to averaging. The red star shows the actual position the screenshot was taken at.

# Chapter 6

# Visualisation of position

Going into the Maps app by Google, Apple or someone else, GPS position is usually displayed as a circle around the position. If the position is accurate, the circle is small and the worse the accuracy, the larger the circle. This section covers the different visualisation techniques for indoor positioning. Previous chapters show that this visualisation can also be applied indoors and bars can be used for linear positioning in hallways.

## 6.1 Visualising Uncertainty

Uncertainty outdoors is an ever growing circle, indoors, location should either not be displayed at all or using a distributed approach like Heat Maps.

As mentioned above, uncertainty outdoors is visualised by a growing circle in which every point being a possible location. This may work outside, however buildings are not designed this way and a growing circle is a fallback option, such as with Flair and its fallback for when beacons are not in a linear setup. There is a huge advantage with how Flair manages beacons, since the static beacon location is known via the `Flair.MapBeacon` list. So as long as the user is in an area with beacons, the worst case is highlighting the closest beacon received. Additionally, previous locations can be used to assume the next location, even if there is noise in the data. This is the idea discussed in chapter 4.5.2: Kalman-Filter.

In the case of indoor positioning, the question is whether one should always display a location, no matter how inaccurate it might be. As an example, Apple recently introduced AirTags, which allow for much better and much more reliable proximity detection than Bluetooth Low Energy beacons, however they cannot be used the same way the beacons were used in this work. When searching for an AirTag, the screen will tell you if it cannot detect the signal or if it is too weak and it won't even try to estimate a direction or a distance in this case. Only when one gets close enough to the tag and the confidence increases, the screen changes to display the orientation and distance in meters.

### 6.1.1 Heat Map

Heat maps show the distribution of position rather than the position itself.

A heat map is useful when a distribution of values and their frequency needs to be displayed. Everyone knows a literal heat map from weather reports showing warmer regions in red and colder regions in blue. This concept can be applied to position and location by marking frequently visited locations warmer than less frequently visited ones. The available region could be separated into smaller pixels, as small as BLE allows them to be in terms of accuracy and error and then the colour could be based on how often a pixel has been visited. Nearby pixels could also be affected in a lighter colour, since getting the correct pixel might be tricky.

**Figure 6.1:** A heat map showing the locations visited most often indoors. Warmer colours indicate more frequent visits[1].

## 6.2 Being confidently wrong

Due to the noise and beacons not broadcasting, the position may be completely wrong even if the data provides it.

The initial plan based on shapes and areas created the idea to position the user into smaller and even smaller areas. So the decision would be a tree with areas as leafs. This binary decision between two areas has been discarded as the primary approach for a continuous one and the reason for this is that a binary decision is either 100% correct or 100% incorrect. In a binary positioning context, one beacon can make a difference of ten or more meters, depending on the size of the areas. In a continuous approach, one beacon affects the position way less, especially when using a noise reduction algorithm.

## 6.3 Using the compass

A compass may help solve the problem of direction.

The compass is a very useful tool for improving location awareness. If it points slightly to the right combined with e.g an image it points to and the user recognises that image slightly to the right when looking up from the phone, it sparks confidence. Before the compass was added to iPhone, users of maps had to walk in a certain direction for a few meters to see how the map updated and only then they could know if they were walking in the right direction to get to their destination. Although not yet implemented, a compass indicating the next beacon in line e.g to implement a guided tour from one exhibit to another in the museum, could be displayed.

As mentioned by Cheraghi et al. [2017], however, a compass is not always accurate either, especially indoors.

[1]Screenshot taken from https://www.mocaplatform.com/blog/dynamic-heatmap-for-indoor-location-analytics

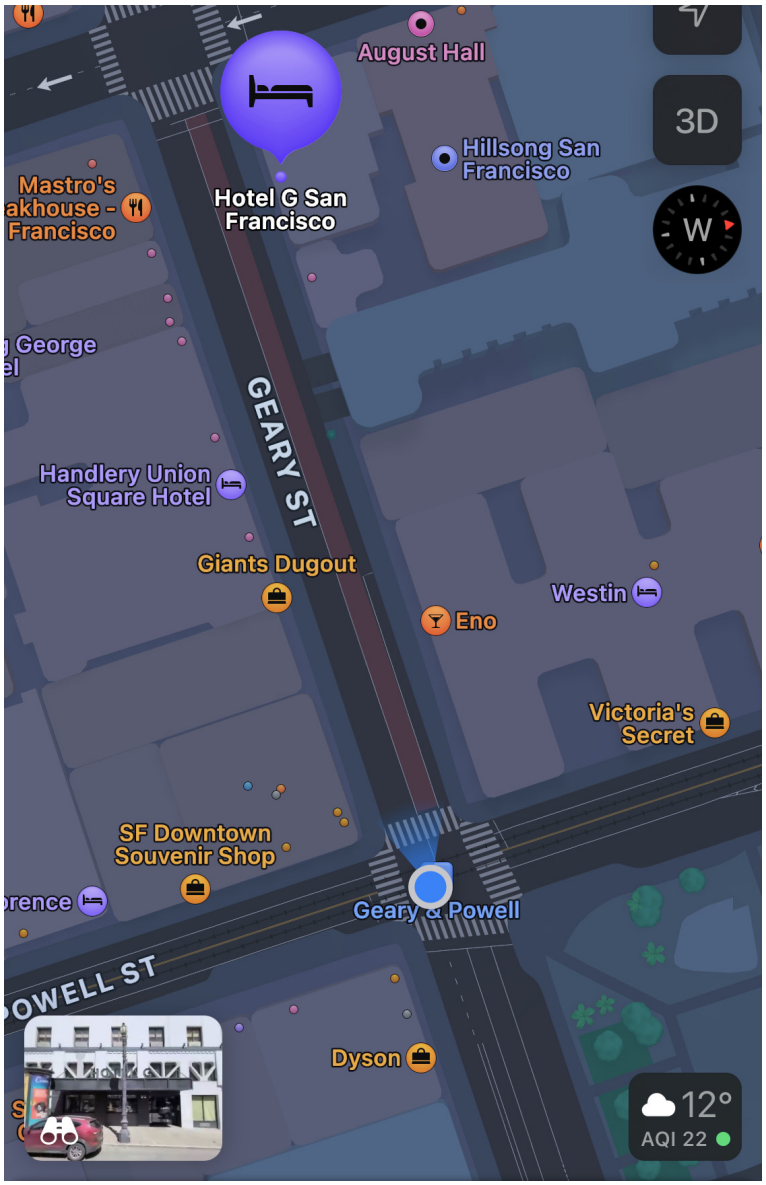


**Figure 6.2:** A combination of a compass and a visual indicator of the direction of the phone solves the orientation problem.

# Chapter 7

# Summary and future work

## 7.1 Summary

Flair can be extended due to its architecture. Working with BLE beacons is still tricky and poses a risk to confusing the user.

Due to the extensibility of Flair, the framework can be used for many different applications. It can be integrated easily into existing projects and works well with any type of map, as it does not provide any fixed UI that needs updating over time. In terms of the underlaying technology, Bluetooth Low Energy is no longer the most modern tech for proximity tracking, yet it is the most accessible and affordable. This summary joins the sentiment of some of the related work in saying that BLE beacons work well enough for some use cases but the system is not stress resistent. Flair does not serve a singular purpose, yet Bluetooth Low Energy reliability is an entirely different story as it was designed for proximity detection, and not for detailed positioning within a room or area.

## 7.2 Future work

As mentioned in this paper, Apple has introduced AirTags which are unavailable to developers at the point of writing.

If developers get access to that type of connectivity in the future and may build small boxes based on that technology, it will improve indoor positioning reliability further. Pricing, next to company policies, might be a factor slowing this process down.

The compass functionality mentioned in Using the compass (6.3) could also be implemented into Flair, given that it already works with `CLLocationManager`.

### 7.2.1 Without Maps - Guided Tours

Most of the related work used maps. Flair supports a setup without a concrete map in mind. AirTag type UI could also be used for BLE beacons.

Maps prove essential for positioning indoors as context is given to where the user is. It would be interesting to consider a guide for a museum that does not use any map. It could act as a guided tour from the first to the last exhibit displaying the image of the next exhibit, along with an arrow indicating the compass and the rough direction to walk in. Then, if possible, a distance in meters could be displayed as well, similar to the user interface of finding an AirTag nearby. This type of navigation would be very strict from one exhibit to another and would work well with QR codes or NFC tags, so BLE beacons are not even necessary, however it may also not be as effective as seeing a dot on the map and seeing the building structure surrounding it. The effectiveness will need to be evaluated.

### 7.2.2 BLE powered Pong

Using a beacon to the left and a beacon to the right, how well does a game work that takes proximity as moving directions?

This game might be difficult to play given that the paddle is affected by fluctuation or the required space to play it is not available. The speed of the game should be low in general to allow for actual movement in a larger space. Apple provides a framework namely `Multipeer Connectivity`[1], which allows for discovering other devices and create a session with them. This could then be used to play the game competitively.

[1] See `https://developer.apple.com/documentation/multipeerconnectivity`

The green circles in the image below indicate beacons on the left and right side of the player. Players move the paddle by walking closer to one side or the other.

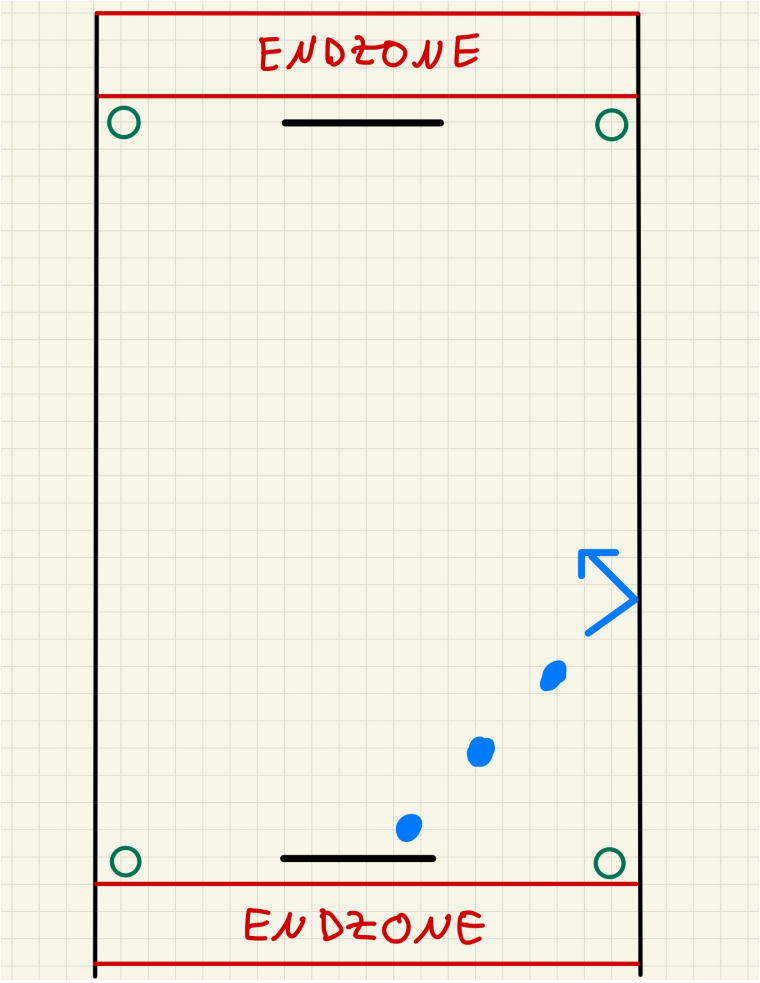


**Figure 7.1:** Pong played by two people using BLE beacons. The green circles are the beacons and so the position between them is calculated for both players.

# Bibliography

S. Bertuletti, A. Cereatti, M. Caldara, M. Galizzi, and U. Della Croce. Indoor distance estimated from bluetooth low energy signal strength: Comparison of regression models. In *2016 IEEE Sensors Applications Symposium (SAS)*, pages 1–5, 2016. doi: 10.1109/SAS.2016.7479899.

Lloyd Arnold Brown. *The story of maps*. Courier Corporation, 1979.

TWC Brown and T Diakos. On the design of nfc antennas for contactless payment applications. In *Proceedings of the 5th European Conference on Antennas and Propagation (EUCAP)*, pages 44–47. IEEE, 2011.

Chen Chen, Yan Chen, Hung-Quoc Lai, Yi Han, and KJ Ray Liu. High accuracy indoor localization: A wifi-based approach. In *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6245–6249. IEEE, 2016.

Seyed Ali Cheraghi, Vinod Namboodiri, and Laura Walker. Guidebeacon: Beacon-based indoor wayfinding for the blind, visually impaired, and disoriented. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 121–130, 2017. doi: 10.1109/PERCOM.2017.7917858.

Gary F. Hatke, Monica Montanari, Swaroop Appadwedula, Michael Wentz, John Meklenburg, Louise Ivers, Jennifer Watson, and Paul Fiore. Using bluetooth low energy (ble) signal strength estimation to facilitate contact tracing for covid-19, 2020. URL `https://arxiv.org/abs/2006.15711`.

Zohaib Iqbal, Da Luo, Peter Henry, Samaneh Kazemifar, Timothy Rozario, Yulong Yan, Kenneth Westover, Weiguo Lu, Dan Nguyen, Troy Long, Jing Wang, Hak Choy, and Steve Jiang. Accurate real time localization tracking in a clinical environment using bluetooth low energy and deep learning. *PLOS ONE*, 13(10):1–13, 10 2018. doi: 10.1371/journal.pone.0205392. URL `https://doi.org/10.1371/journal.pone.0205392`.

Xin-Yu Lin, Te-Wei Ho, Cheng-Chung Fang, Zui-Shen Yen, Bey-Jing Yang, and Feipei Lai. A mobile indoor positioning system based on ibeacon technology. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 4970–4973. IEEE, 2015.

Tommy Nilsson, Alan Blackwell, Carl Hogsden, and David Scruton. Ghosts! a location-based bluetooth le mobile game for museum exploration. 07 2016.

Philipp Wacker, Kerstin Kreutz, Florian Heller, and Jan Borchers. Maps and location: Acceptance of modern interaction techniques for audio guides. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1067–1071, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858189. URL `https://doi.org/10.1145/2858036.2858189`.

Ron Weinstein. Rfid: a technical overview and its application to the enterprise. *IT professional*, 7(3):27–33, 2005.

Chouchang Yang and Huai-Rong Shao. Wifi-based indoor positioning. *IEEE Communications Magazine*, 53(3):150–157, 2015.

Jingjing Yang, Zhihui Wang, and Xiao Zhang. An ibeacon-based indoor positioning systems for hospitals. *International Journal of Smart Home*, 9(7):161–168, 2015.

Dalal Zaim and Mostafa Bellafkih. Bluetooth low energy (ble) based geomarketing system. In *2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–6, 2016. doi: 10.1109/SITA.2016.7772263.

# Index

Janosch Hübner