MASARYK UNIVERSITY
FACULTY OF INFORMATICS

In-door localization and navigation for Android platform

MASTER'S THESIS

Juraj Beníček

Brno, spring 2015

Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Juraj Beníček

Advisor: RNDr. Mária Svoreňová

Abstract

The thesis focuses on solving the problem of indoor localization with using a Bluetooth technology. The thesis contains fundamental information about Bluetooth technology and also information about localization systems. It offers practical solution for indoor localization with detailed description of developed algorithms. However, source code is not included because the thesis was developed for the company Neogenia s.r.o.

Keywords

Bluetooth, iBeacon, Android, AltBeacon Specification, Trilateration, Proximity, In-door Localization

Acknowledgement

Hereby I would like to thank Mgr. Roman Studený and Neogenia s.r.o for opportunity to work on unique project like Spothill. I would also like to express my gratitude for unlimited help to RNDr. Mária Svoreňová.

Contents

1 Introduction

In informational technologies, the problem of indoor localization aims to determine location of objects, assets or persons in an indoor environment. While this problem is easy to formulate, it is hard to solve. The reason is that this issue is very complex. In the last few decades, there was a lot of development in indoor localization systems with WiFi Access Points, sensors and Bluetooth. There is a great demand for indoor localization systems in fields such as asset tracking, industry, marketing and also schools. With the rise of Internet of Things, there is a growing need for precise localization system.

Indoor localization systems are highly motivated by Global Positioning System (GPS). The traditional approach to positioning is based on GPS. GPS is also know as satellite navigation because positioning is based on a network of satellites. This solution provides global localization system and allows small devices to find their longitude and latitude. This system has proved to be very accurate outdoors but when it comes to indoor localization satellites signal tends to shutter and corrupt caving the GPS positioning system to be ineffective. From this comes the desire for a solution to localization an indoor environment.

Indoor localization systems are often based on an already existing network inside a building or structure such as positioning system based on WiFi technology. WiFi is wireless a technology used for device networking. The system consists of a network of WiFi Access Points or WiFi hot-spots. WiFi devices are then located based on their MAC addresses. Indoor localization calculations measure the strength of the signal device inside a room and based on this information, the system approximates location of the device in the room. Alternatively, the system can approximate the position using signal strength fingerprinting. WiFi positioning systems are highly dependent on the signal strength and obstacles can cause inaccuracy between the receiver and the transmitter. Therefore, this solution depends on the density of WiFi devices in the area and accuracy is increasing with the number of devices.

Different approach to indoor localization is based on sensors. A sensor, in this context, is a small device that transmits signal to monitor the surroundings area. WiFi Access Points can also be seen as sensors, however in our case, a sensor refers to a tiny cheaper device that is easier to deploy. There exist several technical solutions for indoor localization with using sensors. The approach called the Angle of Arrival is based on the angle

from which the signal is carried to the receiver. The angle is measured using the system Time of Arrival. Time of Arrival measures how much time it takes for the signal to travel from the sender to the receiver. This system can be very accurate because the time that the signal needs to travel a particular distance is constant. Once the distance from several receivers is computed, the position of the device can be calculated using mathematical methods such as trilateration or multilateration.

The latest solution to indoor localization using sensors is based on Bluetooth Smart technology. In this work, we design an indoor localization system that relies on Bluetooth Low Energy sensors called iBeacons. The work was conducted with Neogenia s.r.o., Brno-based company that focuses on SMS sending gates and deal offering services. The indoor localization system is designed as a part of the project Spothill. Spothill is an application that brings content and deal offers to users in environment such as shopping malls, museums and large events such as concerts and conferences. Spothill works on a principle of scanning for information from iBeacons in close proximity and displaying the information a user's smartphone. The main purpose of indoor localization for Spothill application is to enhance vendors ability for marketing purposes. Vendors can analyze movement of users inside their stores for better product placement and enhancing their stores. This thesis is closely related to another master thesis at Faculty of Informatics on Masaryk University thesis called Analysis of in-door movement, which develops a framework for displaying the data of users locations in user friendly heat maps.

Note that iBeacons as the hardware solution were chosen by Neogenia before this work was created. iBeacons are Bluetooth sensors running on coin cell batteries. These small sensors are easy to maintain because of long battery life. They are relatively cheap to buy and therefore it is also cheap to build a network for indoor localization system.

The rest of the thesis is organized as follows. In Chapter 2, we are going through history and background of Bluetooth technology and some current Core Specification versions. We continue with Chapter 3 that is focused on the hardware used in our indoor localization and these are iBeacons. In Chapter 4, we dive into indoor localization systems, their principles and how they measure distances. In Chapter 5, we introduce project Spothill by Neogenia and integration of indoor localization into this project with on a basic use case. Our most important results are presented in Chapter 6. Here we present algorithms used to solve and implement the indoor localization and also position displaying in Android application. Finally in Chapter 7, we conclude this master thesis and our findings.

2 Bluetooth technology

In this chapter, we introduce Bluetooth technology, the ideas behind this technology and its brief history. We also go through some versions of Bluetooth that are important for our implementation of indoor localization framework.

2.1 Bluetooth

Bluetooth is a wireless technology widely used in world of informational technologies. Bluetooth is used to connect two or more devices and then allow communication between these devices. Nowadays, Bluetooth can be found in all sorts of devices and areas of consumer goods from mobile phones, cars, speakers to light bulbs. It is becoming more and more popular in modern technologies. Especially, Bluetooth has gained a lot of attention in Internet of Things concept. Internet of Things is a network of diverse physical objects and serves the purpose of control and awareness of these objects. Bluetooth technology comprises of hardware and software solution. It means that a Bluetooth devices is not only a small Bluetooth antenna but also the software for connecting to this antenna.

Bluetooth was originally designed to replace physical cables. This technology sends wireless waves but unlike other technologies like television or FM radio, it sends waves only on a short distance. It is meant to be used within users Personal Area Network (PAN) with ranges up to 100 meters [4]. Range of waves can be affected and corrupted by many influences. In ideal circumstances, Bluetooth can have range up to 100 meters but in buildings, especially when there are barriers and obstacles, Bluetooth waves lose their range dramatically. Range is also affected by the manufacturer who chooses radio used in device implementation. In most countries Bluetooth uses unlicensed spectrum from 2.4 to 2.485 GHz. It uses full-duplex signal. Bluetooth can also reduce the interference with other technologies like WiFi and others that also transmit in 2.4 GHz band[14]. This technology is called Adaptive Frequency Hoping and is very effective in its purpose. It works on the principle of using full band spectrum and also simultaneously avoiding frequencies on which other devices on band are transmitting.

Bluetooth is developed and managed by Bluetooth Special Interest Group (SIG) [12] consisting of more than 2500 members. Bluetooth SIG was originally started by five companies. Members of this organization come from a wide range of fields from industry to academic organizations, where Blue-

tooth is used. This organization is responsible for the development and maintenance of Bluetooth standards. Additionally, Bluetooth SIG support the compatibility between different versions and a huge part of work of this organization consist of verifying new Bluetooth versions.

2.2 History of Bluetooth

Main motives behind creating wireless technology was to replace cables RS-232 or at least come in as an alternate these cables [1]. The first developers and engineers behind Bluetooth were from Swedish company Ericsson. They came with this idea of wireless technology in 1994. It took four more yeas for this wireless technology to get its name by which it is known now. This name is Bluetooth. There is a lot of controversy about the origin of this name. According to Bluetooth SIG [4], the name is a metaphor. It comes from a Danish king who ruled Denmark and Norway in 10th century In danish, he was called Harald Blåtand Gormsen. This is translated to English roughly as Harald "Bluetooth" Gormsson. The king united regions of the country in time of the war to act unanimously. As the purpose of Bluetooth wireless technology is to bring different products from different parts of the market to communicate and work together, the technology adopts the name of the king. Bluetooth logo is also highly inspired with this danish ruler and it has his initial runes in Bluetooth logo. The logo is displayed in Figure 2.1.

Bluetooth gained its name in 1998 when the Bluetooth Special Investment Group was originally founded. This company was started by five companies - Ericsson, Intel, Nokia, Toshiba and IBM. These five companies started the group to take care of Bluetooth wireless technology and at the end of 1998, there were already 400 companies interested in Bluetooth and members of SIG. Year later, the first Bluetooth Specification 1.0 came out. This specification however contained a lot of bugs. It was not until five years later, when the verified Bluetooth Core Specification 2.0 came out in 2004. In this time lapse, there were also two minor version v1.1 and one year before Core Specification 2.0 there was v1.2 in 2003. These two versions consisted mainly of bug fixes and improvements. Meanwhile, Bluetooth was implemented in lots of electronic devices such as mobile phones, laptops and printers.

After 10 years of functionality of SIG, the organization was celebrating 10 000 members. Bluetooth was rapidly becoming more and more popular in electronic and non electronic devices. Bluetooth reached an incredible

Figure 2.1: Bluetooth Smart Logo [19].

number of installed devices. Year later in 2009, SIG officially adopts Bluetooth Core Specification 3.0 High Speed. In this year Bluetooth SIG adopts the key features for future specification called Low Energy wireless technology.

2.3 Bluetooth 4.0 Low Energy

In 2010 on 30th of June Bluetooth SIG released Bluetooth Core Specification 4.0 [2] also called Bluetooth Low Energy (LE). This version is commonly known as Bluetooth Smart Ready for host devices and as Bluetooth Smart for transmitting devices such as sensors. It was introduced with low energy technology. Bluetooth devices can now be small with a battery of the size of a coin because of power efficiency of Bluetooth Smart. This allows Bluetooth technology to be implemented into small devices like watches, bracelets and toothbrushes. Bluetooth Smart specification consists of Classic Bluetooth, which includes protocols from older specifications and Bluetooth High Speed specification which was introduced in Bluetooth Core Specification 3.0. This version also includes increased range of waves. In Classic Bluetooth, the typical range of signal waves was 10 meters, but if needed the signal strength can be optimized to about 100 meters. Bluetooth Smart also promises to be inter-operable with devices made by different vendors.

Bluetooth Smart technology was introduced to the market with two options of implementation on devices [18]. The first option is to implement the new technology and Bluetooth Low Energy functions on Classic Bluetooth existing radio. This implementations means that updating to new version of Bluetooth costs manufacturers less than changing manufacturing process all together. This first option is called dual-mode implementation and contains radio from Classic Bluetooth with the newest functionality.

Second option is called single-mode implementation. In this mode only Link layer of low energy technology is implemented. This link layer provides low power idle mode operations and also provides reliable transportation of data from single-mode to multiple hosts. This data are safely

encrypted. Single-mode solution brings extremely low power consumption on these devices meaning long running life on small batteries. Dual-mode also benefits from low energy technology with enhanced battery life.

Bluetooth LE contains changes necessary for implementation of Generic Attribute Profile (GATT). This profile is closely related to Attribute Protocol (ATT) and represents wire application. With this application, clients are able to read or write attributes exposed by server. GATT provides services for establishing operations tied with data transferring and storage. When data are transferred, GATT is responsible for formatting the data. It formats data into group of services and characteristics. Services are data grouped by their purpose to accomplish specific function. A service can also contain data which are referencing to other services. Services are divided into two groups. First group are primary services. These services are in charge of device's main functions. These main functions are referencing the second group of services which are secondary services. These services do not fill main the functions but serve as complimentary services to the primary group of services. After services, there is another group of data which are formatted by GATT and it's called characteristics. Characteristics are values used in services and this data also contains information about representation and access information. In GATT two roles are defined and that is server and client. In Bluetooth LE, these profiles must by implemented because they provide necessary functions for devices to be visible. Also they provide functions needed by other devices to connect to the device.

2.4 Bluetooth Low Energy versions

Bluetooth Smart Ready version of Bluetooth technology was introduced as a direction of development of Bluetooth technology and intentions for future. SIG role is to keep Bluetooth technology on point with trends in technological world. Prior to this there have been two new versions. After Bluetooth Core Specification 4.0 which was released in 2010 and the first update to this technology came in late 2013.

It was officially adopted by Bluetooth SIG in early December of 2013. Bluetooth 4.1 [15] is entirely software update so there are no changes in devices architecture. The main focus of the update was to improve simplicity and usability for consumers. One of the main goals of this specification was implementation of coexistence with the newest cellular technologies like Long-term Evolution (LTE) and avoid band interference between Bluetooth radio and LTE. This coordination of avoiding band interference happens

in background without limiting consumers and letting them know what is happening in the background. Software update was focused also on making developers work easier and the update is mostly related to data transfer. First area of data transfer in which Bluetooth was updated was improvement of control over connections. Another improvement in data transfer was the implementation of bulk data transfer allowing the sensors to send collected data more easily. Another improvement in this update are better exchange rates that are beneficial for transferring data for manufacturers and they enhance data speed exchange.

The latest update was introduced at the end of 2014. Its version number is 4.2 and the main goal of this specification [3] was to keep up with the current trends in informational technologies and the concept of Internet of Things. This latest specification is both software and hardware update, with that older hardware might get some key features for improving privacy. In this specification speed was addressed. SIG says in their press release concerning Core Specification 4.2 that because they increased capacity of Bluetooth Smart packets data transfer improved in speed for about 2,5 times. This also makes data transfer more reliable. Another main issue addressed in the latest specification was privacy. The specification contains the possibility for the user or consumer to adjust privacy. Privacy is enhanced for reducing eavesdropping which can lead into unwanted intruders tracking users. Now there is a way for the user to turn the tracking by sensors and iBeacons off. That means that security and privacy is partially in hands of the customers and they can decide its level. Another improvement that helps involving Bluetooth into Internet of Things the newest protocol Internet Protocol Support Profile which provides IP connectivity [2] and it supports IPv6 connectivity. It enables to connect Bluetooth Smart sensors to internet with IPv6/6LoWPAN.

3 Bluetooth sensors and iBeacons

Bluetooth sensors and transmitters are small devices called Bluetooth Smart. These devices are usually very small and powered by a coin-cell battery on which they can operate for a long period of time. They can be placed anywhere thanks to no need for power source. These sensors and beacons are now used across all mobile platforms such as Android, iOS or Windows but only on devices that are Bluetooth Smart Ready. These sensors are implemented into applications in wide spectrum of scenarios. These devices operate in the single-mode implementation of Bluetooth Smart, allowing them to be configured to transmit waves for long distances and also in high frequency.

IBeacon is Bluetooth Smart transmitter that sends unique identifiers to its local area. With this identifiers, it acknowledges its presence to devices in close proximity. The primarily purpose of iBeacons is to enable devices to perform particular actions when they enter an area where an iBeacon is transmitting. IBeacon was originally introduced by Apple Inc. as an extension to their package of location based services as a proximity location service. Another intention of Apple was to use and integrate it into payment system and as a marketing tool for small retailers. IBeacons are made by multiple vendors, they usually came as small devices with a coin-cell battery. Sometimes they come as USB dongles that don't need battery. In Figure 3.1, we show examples of different groups of devices. IBeacons on the left picture are from manufacturers April Brother and EM Microelectronic. These two types of beacons we had at our disposal from Neogenia s.r.o. for implementation of indoor localization. Both types are running on coin-cell battery but they are slightly different in specification and features. Note that the size of iBeacons is less than 4cm. On the picture on the right, there is an example of a USB iBeacon by manufacturer Radius Networks.

3.1 iBeacons advertisement

IBeacons work on a principle of transmitting an advertisement into it's local area. This advertisement has a special form and contains unique identifiers which can trigger certain action on a receiving phone. The actions performed after seeing iBeacon are based on the implementation of an application. Format of the advertisement sent by iBeacons was defined by Apple Inc.

The iBeacon advertisement has the following form described in [5]:

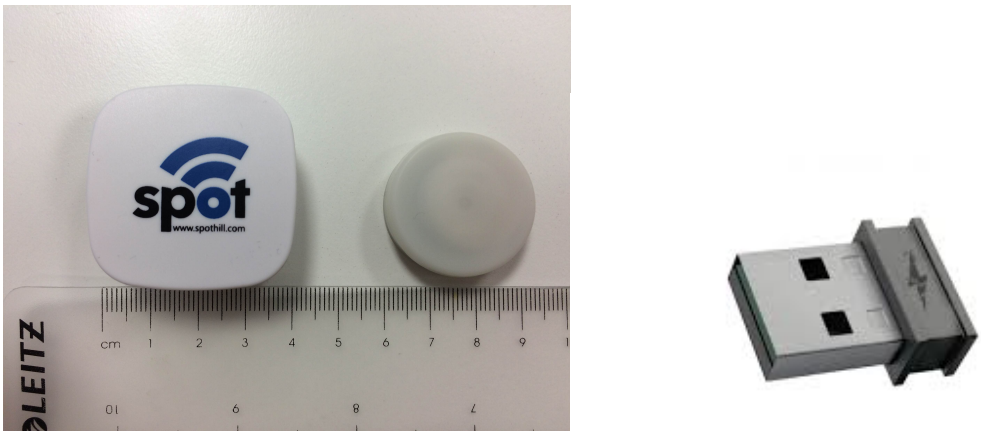

Figure 3.1: iBeacon types [19].

1. iBeacon prefix (9 bytes) - first value advertised by iBeacon is prefix. This value is a fixed advertisement identifier by Apple Inc.

2. UUID (16 bytes) - stands for Universally Unique identifier. It is a 128-bit number representing the advertising company. This identifier is possible to generate with no central registration processes.

3. Major and Minor (2 bytes each) - these two identifiers come in a pair. Their purpose is to serve as identifiers for applications which scan the advertisement.

3.2 Integration of iBeacons into mobile devices

In this section, we discuss possible utilization of iBeacons on mobile devices and we pay special attention to Android-based devices. Currently, platforms such as iOS provide full support for iBeacons whereas other mobile platforms such as Android, BlackBerry in the form of partial support or support with external libraries. There are also some mobile platforms that advertise support in their future updates such as Windows Phone. In mobile world which is constantly changing and generations are changed in a course of years, these support differences for iBeacons can vanish within a year or two and iBeacons can easily become natively supported on all mobile devices.

3.2.1 iOS

Apple Inc. devices run on the operating system iOS. Apple came with specification for iBeacons in June 2013 and provides full support without the need for any external library to be able to read data transmitted into area by iBeacon. Apple includes iBeacons in their location services package to enhance user experience. In iOS, native iBeacon advertisement support was introduced from iOS 7+ and currently 98% [11] of all devices run with version 7 or higher. This is thanks to the popularity of updates and because updates of the operating system are free and available for every Apple device instantly.

iOS offers in it's specification the ability to range iBeacon [6] in region. That means iOS application can be set to listen to certain advertisement. Also there is an option for iOS devices to behave as iBeacons and send advertisement into region. Finally, iOS enables to monitor if devices enter or leave the area monitored by the iBeacon. Applications can then perform specific actions based on this user action such as upon the first entrance of the user to the area or upon leaving.

3.2.2 Android

In this work, we focus on utilizing iBeacons in indoor localization with Android. Android support iBeacons from version 4.3, making implementation of iBeacons possible for more than 50% [10] of all Android devices. This number is based on the data published by Google which were collected in April 2015. The current version of Android operating system, version 5.1, brings even more implementation options for iBeacons. Market shares of Android versions are displayed in Figure 3.2.

Some of the main improvements also mentioned here [16] are:

- Scanning - from Android version 5.0 there has been a lot of improvement in scanning for Bluetooth peripherals, including improved scanning for iBeacons. Scanning is optimized for low power consumption. Scanning does not disable the device from going into stand by mode and more control over scanning is given to developers.

- Peripheral mode - another improvement is the possibility for devices to act as iBeacons. They now can transmit advertisement which developers choose to advertise.

The use of iBeacons with Android devices is naturally connected with the necessity of having a device with Bluetooth 4.0. Since the beginning

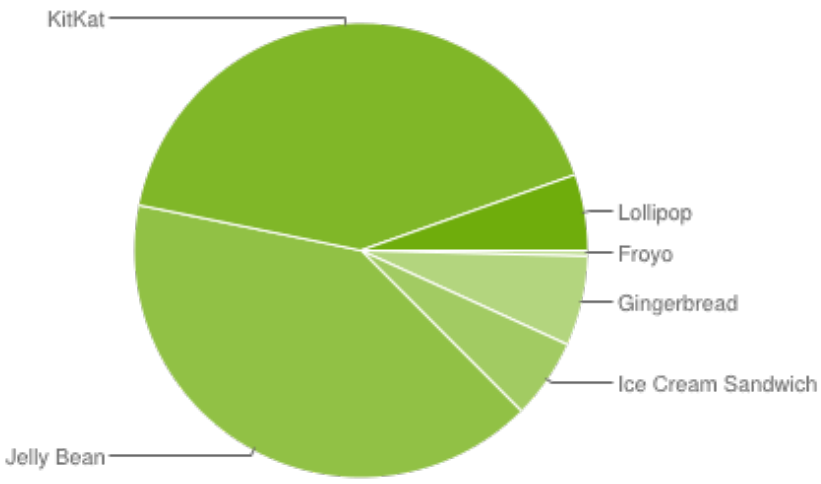

Figure 3.2: Android version share [10].

of 2015 almost all new devices have Bluetooth Smart Ready function implemented. Version Android 5.0 Lollipop already comes with an enhanced support of iBeacons such as possibility of the phone to send advertisement and behave like an iBeacon. An Android application itself then needs a permission from the user to manipulate with Bluetooth. This permission allows the application to use and access Bluetooth devices in the near-by area. There are several different libraries which provide possibilities to read iBeacon advertisement. One of the most popular libraries for Android platform that is also used in our implementation is Android AltBeacon Library.

AltBeacon

AltBeacon is an iBeacon specification created to bring specification for standardized advertisement sent by iBeacons. It is completely open and it was not yet certified by any organization. It was proposed by company named Radius Networks that specializes on iBeacons and proximity solutions. It is proposed to establish standard to unify iBeacon advertisement to ease development for developers as well as manufacturers.

The main goals of AltBeacon are:

- create a simple message format for exchanging information between iBeacons and devices listening to these advertisements,
- maintain compatibility with predefined format from Bluetooth 4.0 specification Protocol Data Unit (PDU),

- make implementation of the message format as simple as possible,
- enable customization for manufacturers.

AltBeacon advertisement is implemented into Bluetooth 4.0 Advertising channel of Bluetooth Low Energy PDU. This advertising channel is a 28 byte file and it contains specific files.

AltBeacon advertisement contains fields described also here [7]:

1. AD length (1 byte) - this field contains length of fields type and advertisement by manufacturer.
2. AD type (1 byte) - type field which defines the type of the advertised data.
3. MFG ID (2 bytes) - represents the identifier number of manufacturer and these are in the format of a little enderian which represents iBeacon company manufacturer. These codes are maintained by Bluetooth Special Investment Group in their global database.
4. Beacon Code (2 bytes) - field which represents AltBeacon code of advertisement.
5. Beacon ID (20 bytes) - identifier number of AltBeacons. First 16 or more bytes represent a unique identifier which on recommendation from specification should be used as the identifier for every AltBeacon used by the company and therefore serve as organizational identifier. When for example this identifier is only 16-bytes long another 4 bytes which are left can be split as needed with different scenarios.
6. Ref RSSI (1 byte) - next filed in line is called Reference RSSI. It's main purpose is to compute the distance between the receiver and the transmitter based on hope that this value is set up right by the manufacturer. This field's 1-byte is an value on scale from 0 to -127. It represent average signal strength received by the receiver. It is measured for one meter and it is averaged from computations.
7. MFG RSVD (1 byte) - last field implemented into AltBeacon Advertisement and it is called MFG Reserved. Its main purpose is to provide some sort of adaptability for manufacturers who create AltBeacons. It is designed to provide program-ability of special features which are provided by specific manufacturers.

Whole AltBeacon Advertisement is graphically shown in Figure 3.3, where we can see this advertisement inside BLE advertising PDU.

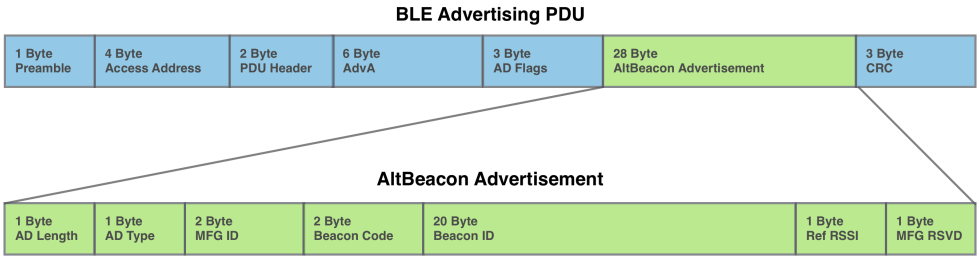

Figure 3.3: AltBeacon advertisement fields [7].

AltBeacon Android Library

Android AltBeacon Library is a project of Radius Networks to implement AltBecon standard into Android library and use it to scan iBeacons. Library is open source and available on GitHub. It is designed to behave, or come as close as possible to behave, as Apple iOS native support for scanning for iBeacons advertisement [8]. That means, it is designed to range and scan for iBeacons and for Android 5.0+ devices and to send iBeacon advertisement. Although the library is designed to scan mainly for iBeacons that meet AltBeacon standard, it can be altered to work with iBeacons from other manufacturers with a little tweak inside the application. Library works with Android 4.3+ devices with Bluetooth Low Energy hardware and only on this circumstances can devices use this library for scanning for iBeacons. It also provides comprehensive Java Doc API for better understanding and use of this library. The current version used in our solution for indoor localization is Android library 2.+ but there is already a lot of work put into creating new version 3.

3.2.3 Windows Phone

Although Windows Phone supports GATT profile [17] on Windows Phone version 8.1 there is no native support for iBeacon advertisement. The operating system does not allow developers to scan for iBeacons because these functions are locked for system functions. Windows Phone 8.1 allows to scan only for connectable Bluetooth devices. Microsoft is planning to expend it's effort for iBeacons in the upcoming version of Microsoft version 10.

3.2.4 Others

Beside the three major operating systems, there are few others that can support iBeacons advertisement. There is Blackberry that allows developers to scan for iBeacons with native SDK and read their advertisement including reading all parameters and identifiers send by iBeacons. Blackberry operating systems also allows building application written in Android SDK. And this means that BlackBerry can run Android applications, that scan for iBeacons.

4 Indoor localization systems

In this chapter, we go through some of the localization systems that are now used in real applications. Different localization systems can be applied for applications with different use cases. They can be used to calculate physical, relative, absolute or symbolic location. Most systems are created to show physical location i.e., the position on a two or three dimensional map marked with a graphical pointer. Also, in indoor localization systems there usually are a lot of obstacles in the environment that need to be taken into consideration.

4.1 Triangulation

Triangulation [9] measures location with use of geometric properties of triangles. Triangulation generally refers to two groups of different approaches to computing position. The first is called lateration and this principle computes the position based on the distance from multiple devices. The second triangulation technique is called angulation and this mathematical method uses computation with angles. The accuracy of these systems degrades with a blocked line of sight caused by obstacles or moving objects.

4.1.1 Lateration

As mentioned earlier, lateration localization [9] system is based on calculating distance from multiple devices. There exist multiple approaches to calculate the distance from the device to the receiver:

- *Time of Arrival* - in short ToA. This method computes the distance by measuring the time of signal traveling. ToA is highly dependent on perfect time synchronization of the device which sends the signal and the device which is receiving the signal. The distance is then computed directly from the time difference between the time stamp sent by the transmitter and the time on the device which received the signal.
- *Time Difference of Arrival* - this method's idea is computing distance by using time difference of signal arrival on multiple receiving devices.
- *Received Signal Strength* - this method examines strength of the signal which was transmitted from device and signal strength when re-

ceiver caught the signal. The difference between these two values is used to calculate the distance.

- *Road Trip Of Flight* - this method is based on measuring time which takes the signal to travel from the transmitter to the device and back to the transmitter. This method solves the problem with the need for time synchronization with ToA. On the other hand, it can suffer from an unknown variable which is the delay caused by the receiver to process signal and send it back to the transmitter.

- *Received Signal Phase* - this method requires that all transmitters send signal of sinusoidal form and they transmit this signal at the exact same frequency. For computing the distance, phase of signal is used at which it arrives to the receiver.

Position of the user itself is computed with use of distances from multiple points on the map. For example very popular and widely used method intersects circles centered at points from which the distances are calculated.

4.1.2 Angulation

Angulation techniques are based on computing the location with angles. The main advantage of this technique is that it does not need multiple points. The location of the device is computed based only on its distance and angle from a single device in the area. However, this method is complex and requires expensive hardware.

4.2 Scene Analysis

Algorithms based on scene analysis work in two phases. First phase analyze the scene by collecting so called fingerprints from different location. These fingerprints are different characteristics of signal at certain positions. The second phase runs in real time and collects signal characteristics from user's device. This data are then compared with location fingerprints to estimate user position.

4.3 Proximity

Proximity based localization systems detect user location only relatively. That means that these systems only tell which antenna is the closest to the

user. Accuracy of these systems is highly dependent of the number of antennas in the area. These systems works on the principle that when a device catches a signal from an antenna, it tells that the user is in proximity of this antenna. When the device catches signal from more antennas, it tells it is in proximity of the antenna whose signal is the strongest.

5 Spothill

Spothill is a project of company Neogenia s.r.o. with residence in Brno. Neogenia's main subject of business is SMS sending gates and deal offering services. Project Spothill started in early 2014 an it's main purpose is to exploit full potential of iBeacons in marketing services and customers satisfaction. The outcome of the project is a smartphone application that is available for free and an administrative tool in form of web application for vendors. It was fully available for use in late 2014 and now it is in phase of continual enhancement of user experience. The project currently offers application an for Android 4.3+ and iOS 7+. As mentioned in Section 3.2.4 there is also a way to install the application on Blackberry devices thanks to support for Android applications. The only requirements for using Spothill application is having a device with this Bluetooth 4.0 hardware to be able to read iBeacon advertisement.

There are three main areas, where Spothill has its scope. The first is marketing. Retailers and big brands can now advertise special deals and offers to customers with installed application. In this context, Spothill was already used for cooperation with Kotva department store in Prague and also with Meatfly retail store in Brno. Spothill is designed to enhance customers shopping experience and help retailers draw their attention in real time. For example, when a customer is passing by a shop, the Spothill application on the customer's smartphone recognizes iBeacon inside the shop and sends notification directly to customer's phone with a special offer. Second, Spothill is very useful in traveling. It can show information about place or historical sites and serve as a guide book. In tourism it also finds its use in museums. It is currently used in Technical Museum in Brno where it gives additional information about expositions. Finally, Spothill can be used at events such as musical concerts and conferences to provide information to visitors regarding ongoing program and possible changes. This functionality was tested on Beat Fest 2015 hosted by Beat Radio in Brno.

Spothill's administrative tools are web applications. That means they are platform free and the content can be edited from everywhere. The Spothill application is based on iOS 7+ and native support of iBeacons from Apple Inc. For Android, project is restricted for version 4.3+ and AltBeacon library which brings support for iBeacons very similar to native support on iOS. IBeacons for this project are from manufacturers April Brother and EM Electronics. Spothill application works on the principle of scanning in background for iBeacons. It reads all available advertisement and based on

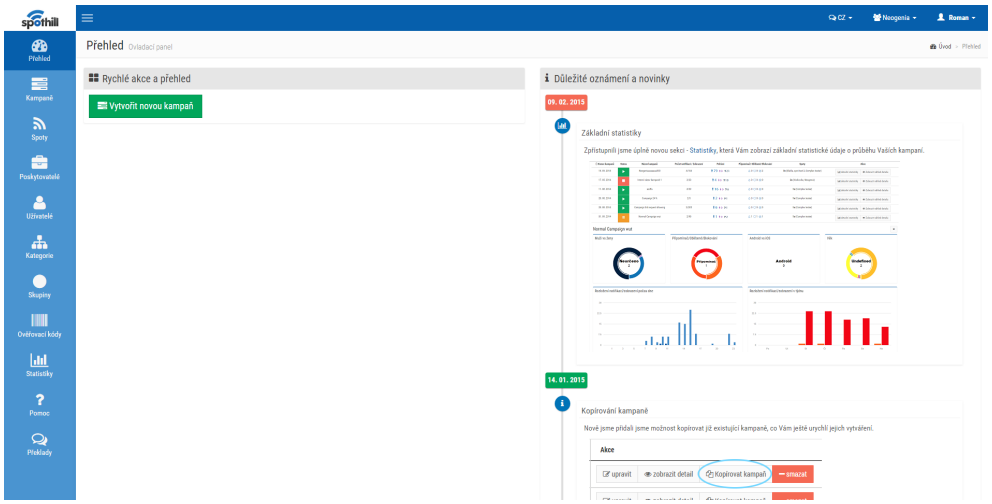

Figure 5.1: Screenshot of Spothill web application

configuration it displays chosen campaign data it shows certain campaigns. Campaigns can be also modified to be shown when the user enters, leaves or stays for some time in a given area where the iBeacon is advertising its data.

In Figure 5.2 there are few screenshot of mobile application. These two screenshots display mobile application while first screen shows homepage of application second screenshot displays campaign detail after clicking on campaign on home page. In this detail we implemented displaying indoor localization. Another screenshot shown on Figure 5.1 is basic interface of Spothill web administrative tool.

5.1 Spothill and indoor-localization

Although indoor localization is not the primarily goal of the Spothill project, it makes a perfect use case of complementary function for project Spothill. In many ways, Spothill is all about marketing your products and it may seem it is about directly speaking to customers with vendor's offer and sharing real-time information into customers phone. But it can also be used for secondary communication with user and therefore enhance customers experience. With this idea indoor localization was developed. It anonymously collects data about customers movement inside vendors shop. The data are analyzed in the Spothill web administration and the vendor can for example see where people are moving inside his shop or where are the

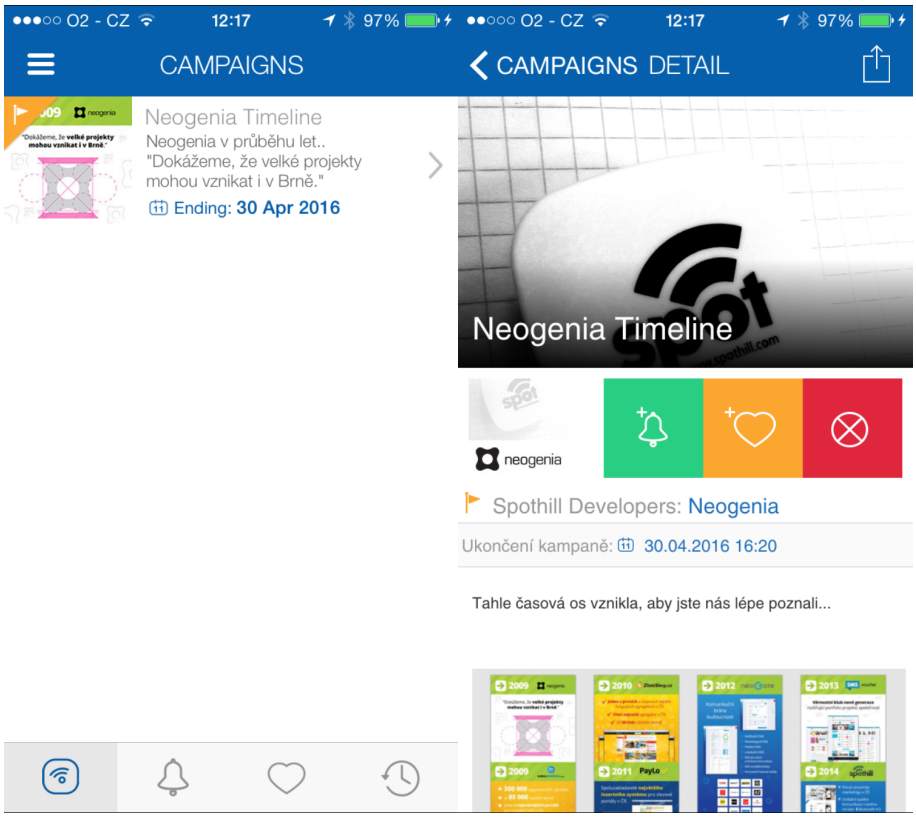

Figure 5.2: Screenshots of Spothill mobile application

bottlenecks of this movement via heat maps. Heat maps and in-door position analysis is a subject of the second thesis related to Neogenia called "Analysis of in-door movement". With this tool the vendor can optimize the product placement inside his store and use this tool for collaborator marketing and enhance their ability to attain more customers. Thanks to the low cost of making grid of Bluetooth iBeacons, it is an affordable solution for all vendors.

6 Implementation

In this chapter, we go through implementation steps which were taken into final solution but also some steps which were taken but did not necessary make it into final implementation. The reason behind this is that this implementation of indoor localization was designed using trial and error approach to find the solution that meets the needs of Neogenia s.r.o. In Figure 6.1, we show the architecture of the project and in the following sections, we describe the functionality of the crucial parts of the project.

6.1 Scanning for iBeacons

In this section, we describe the necessary steps to enable our application to scan for iBeacons and to read their advertisements. This section also contains basic settings of our application. In our solution to indoor localization problem, we use Android Altbeacon Library. This library serves to support iBeacons in the same way as they are supported natively in iOS, see Section 3.2.2.

6.1.1 AltBeacon Library implementation

We use Android Studio for project development. Android studio uses Gradle, an advanced toolkit for custom build application and administration of dependencies within Android project. Android application is based on Activities which are components that provide user interaction. To allow compilation of Android AltBeacon library within our application, we only needed to edit Module Build File that allows us to edit dependencies of the module under development. In the higher build file called Project Build File, there are main dependencies such as Gradle version. But in our project we only needed to edit the module build file. In this file we added the library into dependencies with line:

```
compile 'org.altbeacon:android-beacon-library:2+@aar'
```

This guarantees that we are building with highest possible implementation of AltBeacon Library version 2+. Additionally, variables `minSdkVersion` and `targetSdkVersion` were defined. These two variables restrict versions of Android operating system on which the application can be run.

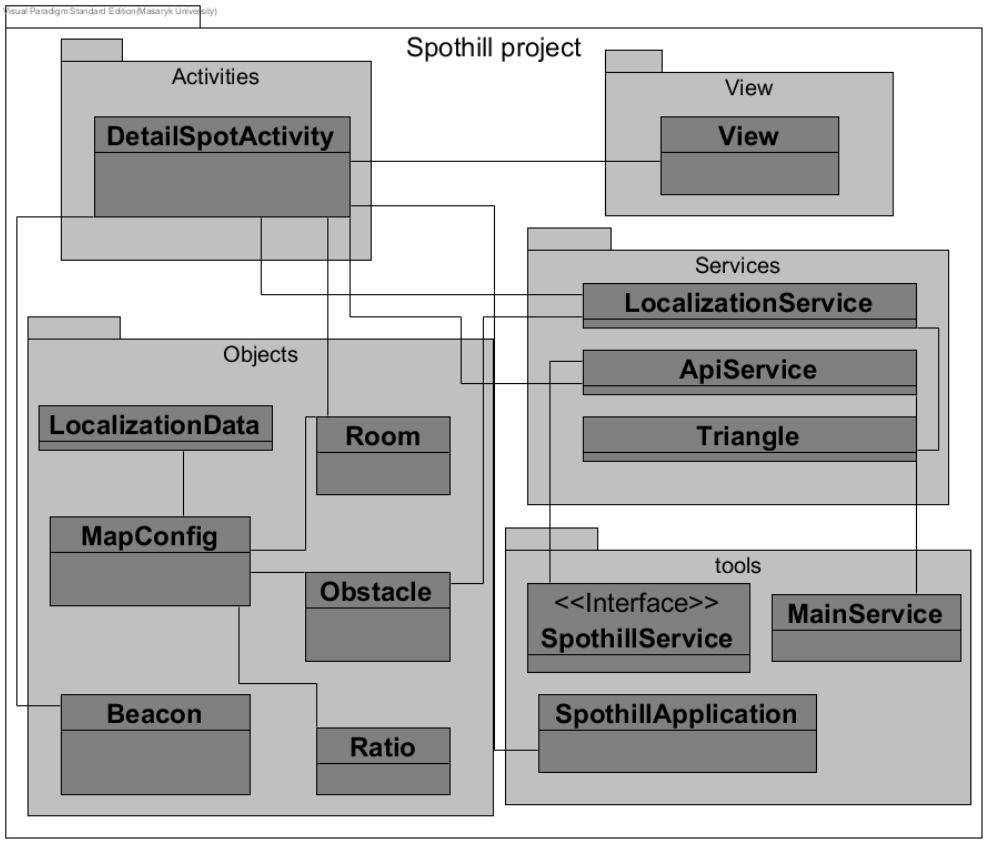

Figure 6.1: Class diagram of project

The variable `targetSdkVersion` defines the version which is optimal for the application and `minSdkVersion` defines the minimal version of Android system.

```
minSdkVersion 18

argetSdkVersion 21
```

6.1.2 Permissions

Besides configuring the module build file, it is necessary to edit AndroidManifest.xml located inside root directory of the application. This file carries all essential information about the application. It is written in XML and for our purposes we need to add a permission for Bluetooth to allow scanning and reading iBeacon advertisement. We then added two permissions into our file namely:

```
<uses-permission android:name='android

.permission.BLUETOOTH'/>
```

and another permission

```
<uses-permission android:name='android

.permission.BLUETOOTH\_ADMIN'/>
```

These permissions grant our application the access to Bluetooth and allow to discover and connect to Bluetooth devices. The values inside parameters of uses-permissions are constant values.

6.1.3 Scanning

Scanning for iBeacons is done using the Android AltBeacon Library. With this library, we configured our Activity in which we are computing and displaying location. For reading beacons there exists an interface in AltBeacon Android Library. We made our Activity to implement this interface called `RangeNotifier`. This interface then calls a method called `didRange-`

`BeaconsInRegion`. This method has parameter `java.util.Collection<Beacon>` and this collections has inside it field of object called `Beacon`. This object contains multiple fields and methods to identify and work with iBeacon.

Important fields and methods:

- *mIdentifiers* - this field represents list of identifiers needed to represent beacon. Important methods are `getId1(),getId2(),getId3()` and they return as mention in order : UUID, major and minor.
- *mDistance* - this field is represented by `java.lang.double` and it is estimated distance from receiver to beacon and how far is it in meters. Estimated distance is available with standard `getDistance()` method.

To start ranging for iBeacons, we first create `BeaconManager`. It is created once in a lifetime of the application with method `getInstaceForApplication`. This method has parameter `android.content.Context`. There was the following issue with `BeaconManager` in our thesis. AltBeacon Library is designed to be used and to support natively beacons with AltBeacon specification but we had in our disposal beacon with other specification so without certain steps it was not possible to read these beacons. To be able to read beacons we added new `BeaconParser` to our `BeaconManager`. This `BeaconParser` has `beaconLayout` attribute and we set this layout to match beacons at our disposal. `BeaconLayout` has parameter in `java.lang.String` which in our case we had to set for:

```
'm:0-3=4c000215,i:4-19,i:20-21,i:22-23,p:24-24'
```

This parameter describes values that can be on specific positions of our iBeacon prefix.

6.2 Computing position

In this section, we go deeply through the implementation of position calculation. We are going through different scenarios of position calculating and we explain used methodologies. In this section, a map refers to a set of rooms that represents the basic area of movement of the users. A room may contain obstacles meaning any physical object, where the user cannot stand. Finally, campaign refers to the information shown in application based on

iBeacon advertisement. Simple example of a campaign is in the leftmost picture on Figure 5.2. We start with describing the data.

6.2.1 LocalizationData class

`LocalizationData` class stores our data for computing position. These data are received from Spothill API. In our project we are using Android library called Retrofit for communication with API. We are not asking for this data with all iBeacons, only when API tells us that our campaign is localization type campaign. In case of user opening localization type campaign, we display different information than with shopping, tourism and events. When user enters detail of localization campaign we not only call the API method that gives us configuration data of map but we start scanning iBeacons inside this Activity. If campaign is not localization type we are not scanning because it is not necessary and we are saving battery.

Data we receive from API look like this:

```
class LocalizationData

{

String title;

String mapUrl;

MapConfig mapConfig;

...

}
```

First is the `title` of the location where we are counting location. We use `title` as a navigation bar title on the top of our Activity. This value is for informational purposes. Parameter `mapUrl` is a hyperlink to the map image for our displaying position. The last parameter in the data received from API is an object created by us called `MapConfig`.

`MapConfig` contains crucial data for implementation of localization system we designed.

```
class MapConfig
```

```
{

array <Room> room;

array <Obstacle> obstacles;

array <Beacon> beacons;

Ratio ratio;

...

}
```

It contains an array of objects called `Room`. This object contains basic data about the room and where are the borders of the room. This data are created in the application PlanMaker which is the outcome of master thesis "Analysis of in-door movement". The second array inside `MapConfig` object is an array of objects named `Obstacle`. This object stores obstacles that are inside the room. It contains coordinates of obstacles and their dimensions. The last array is called `beacons` and it contains all iBeacons that are located inside the building. This array contains objects of type `Beacon`. We receive parameter for every `Beacon` object from API. These parameters are x a y coordinates of the iBeacon location in the building and we receive height in which iBeacon is placed. Height of iBeacon is measured approximately from waist height. This is given by the fact that most users carry mobile devices in their pockets or in their hand. Complementary parameters are identifiers of iBeacon major and minor. Last parameter which is received is object `Ratio`. This object contains ratio of pixels which are one meter on the map.

6.2.2 Computing the number of visible spots

Before we calculate the position itself, there are few necessary steps we need to take. The computation depends on whether we see only one, two and three or more beacons. Therefore, we must first find out how many beacons we see that are in our configuration file. If we see an iBeacon which is not in the configuration file, we cannot use it because we don't know its exact location inside the building. We first receive all available iBeacons in our area

through AltBeacon Library. We call a function in our `LocationService` which we implemented and is called `numberOfVisibleSpots`. This function accepts two parameters and returns the number of beacons that we see and are in our configuration file.

6.2.3 Assigning current distances

After calculating the number of visible spots, we assign actual distance values to beacons that are in the configuration file. At the beginning, every beacon has this value set on 1000 meters and we reset the value after calculating one position. We iterate through received iBeacons from AltBeacon Library and if we find an iBeacon that is in our configuration file, we assign the distance to this iBeacon.

6.2.4 Position calculation

Position calculation itself comes after we assign the distance values. There are three algorithms to find out where the user is located. The reason behind three algorithms is that we are calculating differently when we see various numbers of iBeacons. After we assign current distances, we sort iBeacons in our configuration file by the distance value. For localization purposes we only use up to three iBeacons. It might be the case that less than three iBeacons are visible. We continue with the algorithm that corresponds to the number of visible iBeacons.

In our implementation after our calculations we use an object of type `android.graphics.PointF` to store the computed position and this object holds two variables `x` a `y`. This two variables mark the coordinates in primitive type float and define pixel position on the map.

Position with one iBeacon

This situation happens very rarely when it comes to localization because most of the times, there is a dense grid of iBeacons. But when this situation happens we calculate position with a simple algorithm. We take location, where the iBeacon is located and we place the position randomly near this iBeacon at a computed distance. The distance from the iBeacon was obtained from AltBeacon Library and we multiply it by `ratio`. This is because AltBeacon Library returns distance calculated in meters. To avoid placing user location inside some obstacle, we go through obstacles. In case we placed user location inside one of the obstacles, we place the user posi-

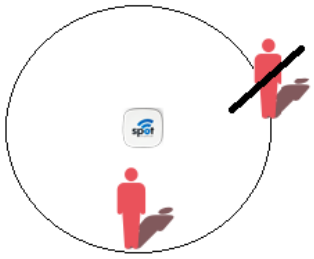

Figure 6.2: Position calculation using one iBeacon

tion randomly in different position from iBeacon at the same distance and repeat process of finding out if the user position is in an obstacle. This process is shown in Figure 6.2, where the crossed position is inside obstacle and the bottom one is returned. After we find a correct position, we return it to Activity. The position is encoded in object `PointF`.

Position with two iBeacons

For the purpose of calculating position, we design an algorithm that is based on moving the position of user along a line. We receive two closest iBeacons and draw a line between them. We take the closer iBeacon and multiply the distance from AltBeacon library with map `ratio`. After we do this, we move the position of the user from closer iBeacon away along this line. This moving on the line is displayed in Figure 6.3. Then we check the obtained position with our algorithm of checking if position is inside of obstacle. If the location of the user is inside an obstacle, we have implemented algorithm which corrects the location based on previous position of the user . This algorithm then returns altered position and it is more thoroughly described in Section 6.2.5. After we check the location for collision with obstacles, we return the position to Activity in `PointF` format.

Position with three iBeacons

This scenario is the most frequent one from the scenarios that we mentioned earlier. We use the three closest iBeacons. First, we create an object `Triangle` that contains these variables and methods:

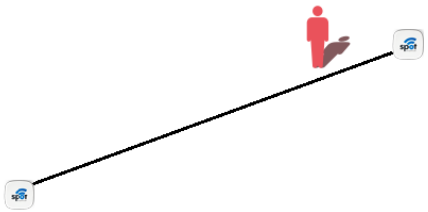

Figure 6.3: Position calculation using two iBeacons

```
class Triangle

{

PointF edgeA;

PointF edgeB;

PointF edgeC;

constructor Triangle(Beacon beaconA, Beacon beaconB,

Beacon beaconC)

{

edgeA = new PointF(beaconA.getX(), beaconA.getY());

edgeB = new PointF(beaconB.getX(), beaconB.getY());

edgeC = new PointF(beaconC.getX(), beaconA.getY());

}

function centreOfGravity()

{
```

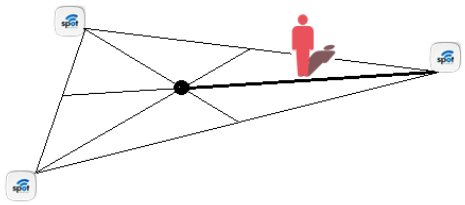

Figure 6.4: Position calculation using three iBeacons

```
returns centreOfGravity;

}
```

This object consist of the positions of the three closest iBeacons and represents the triangle which these iBeacons form. After we create this object, we also multiply the distance from the closest iBeacon by the map `ratio`. This value is later used in the algorithm. We then use function `centreOfGravity()` that is implemented in the object Triangle and this function returns the center of gravity of the triangle created from iBeacons. This function returns the center of gravity of the formed triangle in `PointF` format. The next step of the algorithm creates a line between the center of gravity and the closest iBeacon. We find the point on this line whose distance from the closest iBeacon corresponds to the earlier calculated distance of the user from the iBeacon. Using the same approach as in the previous section, we then move the point along the line away from the iBeacon to find the position that does not coincide with any of the known obstacles. Graphical sample on how this works is in Figure 6.4.

Distance iteration

For a more accurate position computing, we implement a mechanism that computes the position only after four consecutive scans for iBeacons. It holds that the closer the iBeacon is, the more accurate is the information about its distance. Therefore, we choose the minimum of the four distances obtained from each iBeacon. This simple mechanism makes the calculation of the new position a little bit slower. However, it highly improved jumps which have been caused by reflected or delayed iBeacon signals.

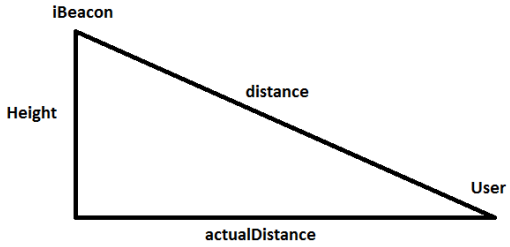

Figure 6.5: Use of Pythagorean theorem

Utilizing height

To further improve the calculation position, we consider the height in which the iBeacon is placed. Due to the diversity of location and building constructions, we might be forced to place iBeacons at different heights. We implement a simple feature to handle such situations. From API, we receive with each iBeacon location also the height from waist height to the actual iBeacon placement. We use this feature to fix the problem of the inaccuracy when iBeacon is placed up high and Bluetooth signal therefore travels longer period to the receiver. We use a simple application of Pythagorean theorem.

The situation is shown in Figure 6.5. The Pythagorean theorem computes the distance as follows:

$$actualDistance = \sqrt{distance^2 - height^2}.$$

This alternation of distance returned by iBeacon is used in all three algorithms.

6.2.5 Finding a collision-free position

In the above listed algorithms for finding the location we use the following algorithm to find a position that does not collide with any of the obstacles. This process is handled by algorithm which works with an old position and a new position. Old position refers to the previous user position calculated for the previous time step and new position is the position calculated fot the current time step. Every time we calculate and display a new position, we save it also into variable `oldPoint`. This variable in `PointF` format then serves as a short term history for handling positions inside obstacles. When function `checkForObstacles` is called, it receives as a parameter the new

location. First, we iterate all obstacles and find out if the new location is inside one of them. The algorithm for finding out if location lies inside of an obstacle was developed in master thesis "Analysis of in-door movement" which was likewise issued by Neogenia s.r.o. If the location does not fall into any of them, we return the new location unchanged. Otherwise, we move point out of the obstacle in the direction of the `oldPoint`. We return the changed position.

6.3 Computing position - alternative

Before the algorithm in Section 6.2 was designed we tried a different approach using trilateration. Trilateration is used for mathematically calculating position based on the known distances with circles, triangles or spheres. We implemented Tulip Alternative Trilateration [13]. This trilateration uses simple formulas to compute x and y coordinates of the position. These formulas only need the locations of iBeacons and distances of these iBeacons. Formulas for counting coordinates are:

$$\begin{aligned} x = (([(d_1^2 - d_2^2) + (i_2^2 - i_1^2) + (j_2^2 - j_1^2)] \cdot (2 \cdot j_3 - 2 \cdot j_2) - [(d_2^2 - d_3^2) \\ + (i_3^2 - i_2^2) + (j_3^2 - j_2^2)] \cdot (2 \cdot j_2 - 2 \cdot j_1)) \div [(2 \cdot i_2 - 2 \cdot i_3)(2 \cdot j_2 - 2 \cdot j_1) \\ - (2 \cdot i_1 - 2 \cdot i_2)(2 \cdot j_3 - 2 \cdot j_2]) \\ y = [(d_1^2 - d_2^2) + (i_2^2 - i_1^2) + (j_2^2 - j_1^2) + x \cdot (2 \cdot i_1 - 2 \cdot i_2)] \div (2 \cdot j_2 - 2 \cdot j_1) \end{aligned}$$

Variables called `i` are x coordinate of iBeacons. They are numbered randomly because in trilateration order does not matter. Variables called `j` are y coordinates also numbered randomly. Finally, variables `d` are the distances from the device to iBeacons.

We used this implementation in our solution for Neogenia but it did not meet requirements by Neogenia. In small rooms such as a room with dimensions 3x4 meters trilateration worked fine and could be even more accurate than our implementation in Section 6.2. But in scenarios with bigger scale rooms and open space trilateration failed to be useful and accurate. We used iBeacons from EM Electronics and these iBeacons showed distance from our position in maximum value of approximately 7 meters. And this property of iBeacons turned out to be a problem for Tulip trilateration. In big open spaces, where all iBeacons were returning distance values around 5-7 meters, the position computed using trilateration was always calculated in the center of those three closest iBeacons.

However, the implementation of trilateration was left in our application because it turned out to be useful inside small rooms. Although it is not used in the current version of the application, it is highly possible that this implementation will be used in future versions for calculating position on maps with highly dense grid networks of iBeacons in small rooms. But for large areas trilateration requires much more iBeacons to be deployed for appropriate accuracy and is much more expensive to use.

6.4 Displaying position

In this section, we describe how the computed position is displayed in the Spothill application. Displaying the user position itself is in Activity, where the detail of the campaign is located. The activity is altered in a certain way to show indoor localization only in case when a campaign of type localization is passed to Activity.

6.4.1 Map displaying

When we faced the problem of displaying images of maps on Android devices, there were three main features of a map that we needed to handle. First requirement was to be able to handle large images with building plans. This can be a problem with simple `ImageView` in Android due to `OutOfMemoryError` issue. Second requirement was to have the feature pinch to zoom on our map. Pinch to zoom is a gesture with two fingers on display which zooms in or zooms out at the image based on the motion of fingers. Last requirement was to be able to use panning gestures with image itself. This is basically scrolling with fingers on the map.

Due to the nature of the thesis and its main purpose in indoor localization, we decided to use an external library and not implement these functions by ourselves. We tried several Android libraries until we found a library most suitable for our purposes. The library is very simple and has good documentation so it is also easy to use. The library is called Subsampling Scale Image View. It is built on top of the Android Image View and implements more advanced features. The library works with bigger images with sub-sampling technique and first only low resolution base of the image is displayed. We first need to compile this library with our application and for this we used this line in Module Build File:

```
compile 'com.davemorrissey.labs:subsampling-scale
```

Figure 6.6: Pin used for displaying the user location on the map

```
-image-view:3.1.0'
```

We then have every feature of this library at our disposal. For our purposes of showing user location, we created our own `View` that implements interface `SubsamplingScaleImageView`. We override a method called `onDraw` and with this method we are drawing a pin on the map with coordinates we calculated with position calculating. We designed our own image to be displayed as a pin and it is displayed in Figure 6.6.

In the `View`, we also implemented dynamic location update. Method has only one parameter and it is `PointF` object which marks the user location on map. This method then proceeds to call `onDraw` method which draws the pin showed in Figure 6.6 on the map. In Android SDK, we also need to implement our `View` into activity layout. The layout is described in XML file and we add this code to the XML file of Activity :

```
<View

android:id="@+id/viewIdentifier"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:visibility="gone" />
```

We set the visibility of our `View` to value `gone`. It causes that the `View` cannot be seen by default when we start Activity in which we are showing the user location. But when we recognize campaign of localization type, we

Figure 6.7: Button for transition zoom in

set the visibility of this element to value `visible` with this line:

```
findViewById(R.id.viewIdentifier).setVisibility

(View.VISIBLE)
```

After the above change, the location on the map is displayed by simply calling the method `setPin()` with the location.

6.4.2 Transition mechanism

One of the requirements by Neogenia for indoor localization was to implement transition into user location. The transition means zooming in into user location with animation after button click. This is also one of the reasons why we picked Subsampling Scale Image View which allows animations of zooming into positions on image. We first implemented a graphical element for this feature and this was a button. The button was also custom designed for project Spothill and it is displayed in Figure 6.7.

Then we continued with setting up the function of animation. First we implemented `onClickListener` for our button. This `onClickListener` reacts to a click and calls a method performed by `AnimationBuilder` which zooms on our location with animation. We only need to supply location in `PointF` object and the library takes care of the rest.

6.5 Implementing on other platforms

One of the last requirements from Neogenia on implementing indoor localization on Android operating system was to be able to easily transform the used patterns and formulas on other platforms such as iOS. Every algorithm was created without any special functions from Android Development Kit. The only exception in this case was the object called `PointF` which is located in `android.graphics` package. But this object can be

easily replaced with a simple object with two float variables for x and y coordinates. Thanks to this, every algorithm could be easily implemented in iOS. On the other hand, implementing displaying location on the mobile phone of a user must be implemented with use of iOS specific functions and libraries. This is due to differences in development kits and also structures of the operating systems itself.

Although indoor localization was not yet implemented into iOS Spothill application, we implemented the designed indoor localization in web application using PHP/Nette framework, we did not faced any problems implementing these algorithms on this platform. Besides the platform specific functions for displaying location.

7 Demonstration

In this chapter, we demonstrate the use off the Spothill application in real scenario. Consider the international trade fair AMPER 2015. This event was held in Brno Exhibition Center in March 2015 and Spothill application was used to demonstrate possibilities of iBeacons and also to promote company Neogenia s.r.o.

The trade fair was held in a warehouse 100x100 meters. In this area, 60 iBeacons were used to provide precise location.

In Figure 7.1 (a), we display one of the warehouse corners and the location of the iBeacons. The precise coordinates and identifiers of the iBeacons are listed in Table 7.1 and it contains major and minor identifiers of iBeacons. Columns x and y marks iBeacons' coordinates on map image. Last parameter is height, which is the height difference between placement of iBeacon and waist height. Consider we are standing near iBeacon with major 1 and minor 301. In Figure 7.1 (b), the position is marked with a black cross. These are data which we obtained from the three iBeacons :

1. iBeacon 1 300 : 6.5893 meters
2. iBeacon 1 301 : 5.3755 meters
3. iBeacon 1 311 : 1.2788 meters

We calculate the position with using algorithm described in Chapter 6 and display the resulting position in Figure 7.1 (b) using the pin. In this particular case, the error is about 1 meter. However, the accuracy scales with the number of iBeacons deployed in the area.

Table 7.1: Coordinates and identifiers of iBeacons

Major	Minor	x	y	height
1	300	1076	4358	1 m
1	301	1419	4470	1 m
1	311	1075	4070	1 m

(a) Amper 2015 iBeacon placement (b) Amper 2015 localization example

Figure 7.1: Amper 2015 map

8 Conclusion and possible future extensions

In our master thesis, the goal was to implement an efficient solution to indoor localization problem. We investigated several solutions for this complex problem and finally produced a solution suitable for needs of Neogenia s.r.o. We describe it in our thesis with all algorithms and partial solutions which ultimately lead to the final implementation. The final implementation meets all requirements of Neogenia. The resulting indoor localization system was implemented in Android application called Spothill. The implementation was successfully used on trade fair AMPER 2015 where we tested the accuracy of our indoor localization system. The accuracy was from 0,1 to 5 meters. The accuracy is highly dependent on the density of iBeacons network. While implementing our solution we took into consideration possible future implementation on iOS.

In the process of the solution design, we came across few obstacles. Most of them came from the nature of Bluetooth itself. The signal of iBeacons have a tendency to be very fragile and can be easily shadowed. This was causing some inaccuracies in distance calculations although we partially solved this issue with our algorithm. In the end, iBeacons as Bluetooth advertisement transmitter turned out to be usable for indoor localization system. Beside some issues with signal, their cheap price and easy installation in every environment make iBeacons a suitable hardware for indoor localization.

In pursuit of more accurate and advanced indoor localization system there are few possible options:

- WiFi network - create a hybrid system with use of iBeacons and WiFi network for more precise solutions.

- Mobile phone sensors - sensor integrated inside smartphone such as gyroscope and compass can be used for user movement prediction and therefore faster reaction times of system.

- Navigation system - create a system for finding shortest path possible and draw a line to a destination on the map of environment.

- Map Library - improving user experience by creating own Android library for map displaying with enhanced options such as rotating map with finger gestures and implementation of compass arrow for better orientation. This point is a follow up on the previous point,

where a new own system for drawing path for navigating would be necessary.

Bibliography

[1] History of the Bluetooth Special Interest Group. Bluetooth SIG. Bluetooth [online]. Available from: http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx

[2] Specification Adopted Documents: Adopted Bluetooth Core Specifications. Bluetooth SIG. Bluetooth [online]. Available from: https://www.bluetooth.org/en-us/specification/adopted-specifications

[3] New Bluetooth® Specifications Enable IP Connectivity and Deliver Industry-leading Privacy and Increased Speed. Bluetooth SIG. Bluetooth [online]. Available from: http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=220

[4] A brief tutorial on Bluetooth wireless technology. Bluetooth SIG. Bluetooth [online]. Available from: http://www.bluetooth.com/Pages/Fast-Facts.aspx

[5] Getting Started with iBeacon: Version 1.0. Apple Inc. [online]. Available from: https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf

[6] Region Monitoring and iBeacon. Apple Inc. [online]. Available from: https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/LocationAwarenessPG/RegionMonitoring/RegionMonitoring.html

[7] AltBeacon Protocol Specification v1.0. [online]. Available from: https://github.com/AltBeacon/spec

[8] Android Beacon Library: An Android library providing APIs to interact with Beacons. [online]. Available from: http://altbeacon.github.io/android-beacon-library/

[9] Mathieu Bouet, and Aldri L. Dos Santos. (2008). "RFID tags: Positioning principles and localization techniques," 2008 1st IFIP Wireless Days [online]. pp. 1-5

[10] Dashboards: Platform Versions. [online]. Available from: https://developer.android.com/about/dashboards/index.html?utm_source=suzunone

[11] App Store Distribution. [online]. Available from: https://developer.apple.com/support/appstore/

[12] About the Bluetooth SIG. [online] Available from: https://www.bluetooth.org/en-us/members/about-sig.2015-05-21

[13] TULIP Algorithm Alternative Trilateration Method. [online]. Available from: https://confluence.slac.stanford.edu/display/IEPM/TULIP+Algorithm+Alternative+Trilateration+Method

[14] A Look at the Basics of Bluetooth Technology. [online] Available from: http://www.bluetooth.com/Pages/Basics.aspx. 2015-05-21

[15] Updated Bluetooth® 4.1 Extends the Foundation of Bluetooth Technology for the Internet of Things. [online] Available from: http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=197.2015-05-21

[16] Android 5.0 APIs: Wireless & Connectivity. [online] Available from: https://developer.android.com/about/versions/android-5.0.html

[17] Bluetooth: Connectivity. [online]. Available from: https://dev.windowsphone.com/en-us/OEM/docs/Driver_Components/Bluetooth

[18] The Low Energy Technology Behind Bluetooth Smart. [online]. Available from: http://www.bluetooth.com/Pages/low-energy-tech-info.aspx

[19] Connect to the Broadest Range of Devices. [online]. Available from: http://www.bluetooth.com/Pages/Bluetooth-Brand.aspx

[20] Radius Networks RadBeacon USB - Proximity Beacon with iBeacon and AltBeacon Technology. [online]. Available from: http://www.amazon.com/Radius-Networks-RadBeacon-USB-Technology/dp/B00JI69JZY