University of Hradec Králové
Faculty of Informatics and Management
Department of Information Technologies

MASTER'S THESIS

Radio Fingerprint Acquisition Using Smartwatch

Author: Bc. David Sucharda

Study programme: Applied Informatics

Supervisor: Ing. Pavel Kříž, Ph.D.

Hradec Králové April 2018

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a uvedl jsem všechny použité prameny a literaturu.

Declaration

I declare that I have elaborated this thesis independently and listed all the sources and literature.

Hradec Králové day 26th of April 2018 Bc. David Sucharda

Poděkování

Rád bych zde poděkoval Ing. Pavlu Kříži, Ph.D. za odborné vedení práce, podnětné rady a čas, který mi věnoval.

Acknowledgement

I would like to thank to Ing. Pavel Kříž, Ph.D. for professional guidance, incentive advices, and the time he provided me.

Anotace

Název práce: Sběr rádiových fingerprintů pomocí chytrých hodinek

Diplomová práce se zabývá možnostmi sběru rádiových otisků (fingerprintů) za pomoci chytrých hodinek. Tyto otisky se používají k lokalizaci uvnitř budovy. Hlavním cílem této práce je prozkoumat možnosti sběru otisků a návrh aplikace která bude tento sběr umožňovat. V první části práce je potřeba zjistit, jestli je tento sběr na hodinkách vůbec možný. V další části je zpracování aplikace pro mobil a hodinky. Poslední část této práce je zaměřena na sběr otisků a jejich analýza. Jeden z osobních cílů je zpracovat tuto aplikaci aby byla co nejvíce uživatelky přívětivá a jednoduchá na použití.

Annotation

The Master's thesis deals with possibilities of collecting radio fingerprints with the help of smartwatches. These fingerprints are used in indoor localization. Main aim of this thesis is to explore possibilities of fingerprint collection and creation of application that will collect them. First part is to figure out if this collection is even possible using smartwatches. Next part deals with creation of such application, not only for watch but also for the phone. And finally there is testing of fingerprint collection and data analysis of collected data. One of the personal goals is to make this application as user friendly and easy to use as possible.

Content

List of figures

List of tables

1 Introduction

As the technology evolves it unlocks more and more possibilities. Just a few years back there were no smartwatches or phones but at this time they are important part of our lives. As they evolve there is the need for them to have more functions and features. One of these features is to be able to locate its position on the map. This information is very useful since it can prevent people from getting lost, figuring out path to drive, used by military and in countless more cases.

Finding such position is possible using Global Navigation Satellite System (GNSS). Multiple implementations of this system exist, such as GPS, GLONASS or Galileo. All of these systems provide location using sufficient number (at least four) of satellites [1, 2]. GNSS solution requires clear line of sight between satellites and the receiving device because signal is not able to pass through buildings. This makes it the main reason why it cannot be used for indoor localization.

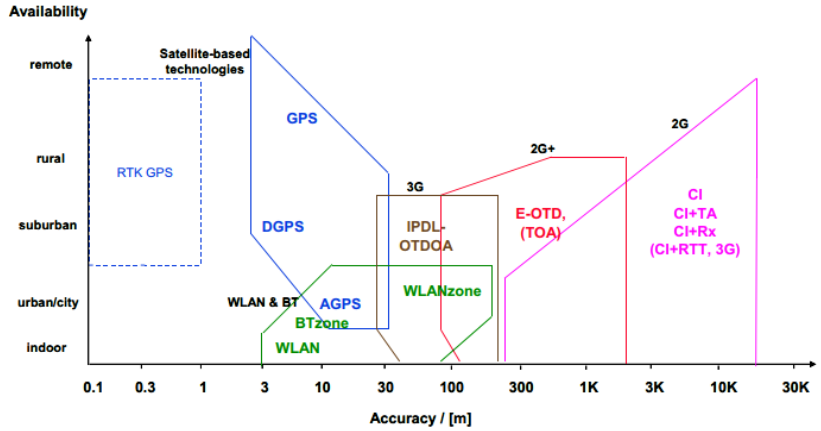

Figure 1.1: Comparison of Positioning Technologies (source: [4])

There are multiple approaches to find out location inside the building. They can be divided into three main types. First, using wireless signal ranging approach with multiple kinds of data such as Time of Arrival (ToA). Second, using special equipment like active bats (Ultrasonic). Final, based on Signal Strength Fingerprint Maps (SSFM), in which first part

is to collect signal strengths from the environment and construct fingerprint maps. These maps are then used to match with current signal to obtain device location [3].

In addition to these types of localization there are also multiple algorithms used in indoor environments. Some of them are location fingerprinting, triangulation, proximity and dead reckoning [5]. Few of these algorithms will be described in Chapter 2.

This thesis is focused on method using radio signal strength (RSS) fingerprinting by collecting data from Bluetooth, wireless and cellular devices.

1.1 Goals of this thesis

Main goal of this thesis is to explore possibilities of fingerprint acquisition using smartwatch technology. The first question that needs to be answered is if this can be done. Is smartwatch capable of RSS data collection? And the answer to this question is yes, since smartwatches have the similar specifications as low-end smartphones, containing Bluetooth and WiFi chips, which means it can be done.

One of the goals for this thesis is to create an application for Android phone and wear device which handles RSS fingerprint collection. Problem with smartwatches is their diversity in operational systems because a lot of watch creators have their own custom systems which can complicate things. Luckily there is a version of Android operating system called WearOS (used to be Wear 2.0) and it is basically a port of Android system to wearable devices.

Final goal is to test created application and figure out if data from wear device increase precision of indoor localization or not.

1.2 Reason for selection of this topic

The reason behind selection of this topic is rather simple. I was introduced to Android during my studies at the University Hradec Králové but it was just basic knowledge. That is why I later decided to go for a study abroad to deepen my knowledge. Part of that study was to work for a selected company where we developed rather complex Android application. Its core part was using multiple APIs, user authentication and data encryption but it was still focused only on a singe device, that being the phone. So next thing I wanted to try was working with multiple kinds of devices and since WearOS is rather new I wanted to test it out. So the main reason is to get more experienced with Android and as a developer.

2 Localization techniques

This chapter describes most common techniques and methods for localization. Most of these approaches have multiple implementations and can be also used in parallel to improve accuracy. Fingerprinting for example can be used to increase precision of other methods.

2.1 Triangulation

Methods based on Triangulation use geometric properties of triangles to determine target's position. This can be divided further into Lateration and Angulation [6]. There are multiple sources of data these methods can use, such as distance estimation between device and specific transmitters, measurements of the signal propagation-time (TOA: Time Of Arrival and TDOA: Time Difference of Arrival[7]) and the direction of received signal (AOA: Angle of Arrival[8]) [9].

2.1.1 Lateration

Lateration refers to the technique of determining position based on distance measurements which are calculated using specific devices that know their own position. Mainly used types of Lateration are Trilateration and Multilateration.

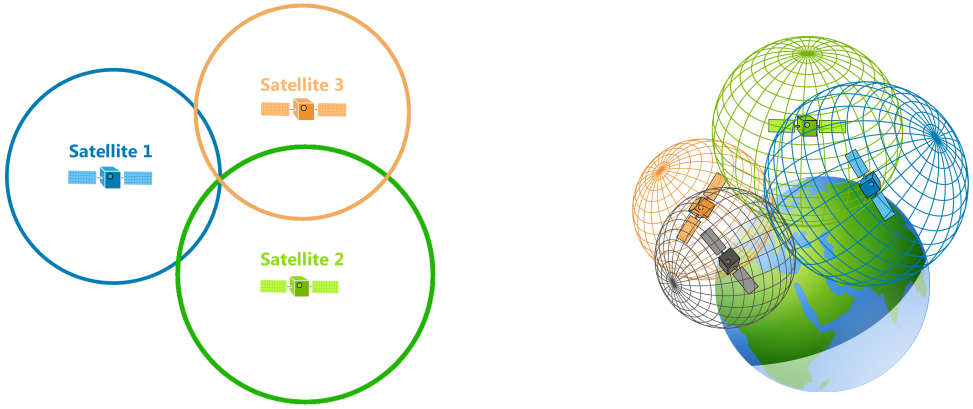

Figure 2.1: 2D and 3D Trilateration (source: [10])

Trilateration uses distance measurements from at least three devices in particular, as "tri" in the name suggests [6]. Figure 2.1 illustrates usage of Trilateration in 2D and 3D environments. While working in 2D plane will result with only one specific location point, moving to the 3D plane can create a problem because signal is send in a sphere which could result in more than one position. That is the reason why some systems use at least four signal sources to get single location, example of such system is GPS [2]. Advantage of this approach is easy implementation and simple calculations. One downside of this approach is that all devices must have synchronized clock [6] to prevent localization errors.

Multilateration, also known as hyperbolic positioning is using Time Difference of Arrival (TDOA) instead of Time Of Arrival (TOA) used in previous case. This approach uses intersections of hyperbolas rather than circles as shown in Figure 2.2. Main advantage of this method is that only receiving devices must have synchronized clock instead of all [12]. Multilateration was developed for tracking aircraft position and it is widely used not only for this case.

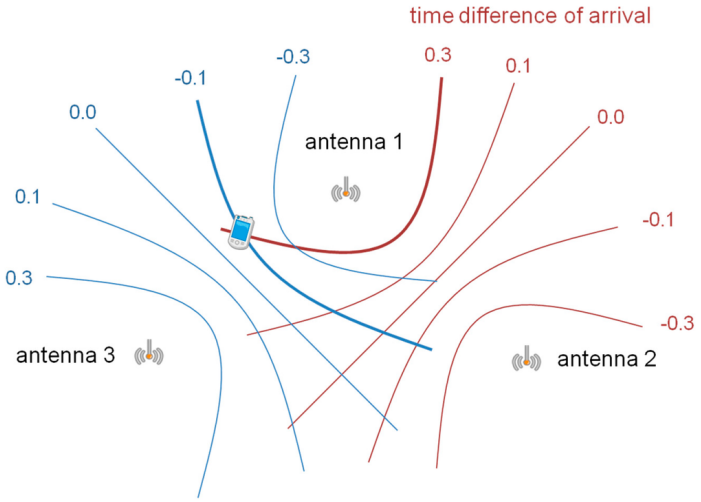

Figure 2.2: Multilateration (source: [11])

Note: At this time term Multilateration is not as strict as it used to be. It can now refer to Lateration with more than three devices.

2.1.2 Angulation

This technique uses Angle of Arrival (AOA) of radio signals to determine location and it requires highly directional antennas or antenna arrays. Same as Lateration these antennas are placed in known location and basic AOA requires at least two of them to determine position on 2D plane or more of them to improve accuracy [6]. Requiring only two antennas is

an advantage over Trilateration which requires at least one more. Second advantage of this approach is no need for clock synchronization between devices.

There are also few disadvantages of this approach since it needs complex hardware setup due to the use of directional antennas. Other problem is with multipath locations since it can cause signal reflection and thus making it not useful for indoor localization. And final one to mention is the decrease of accuracy when mobile target moves further from the antennas [13, 14].

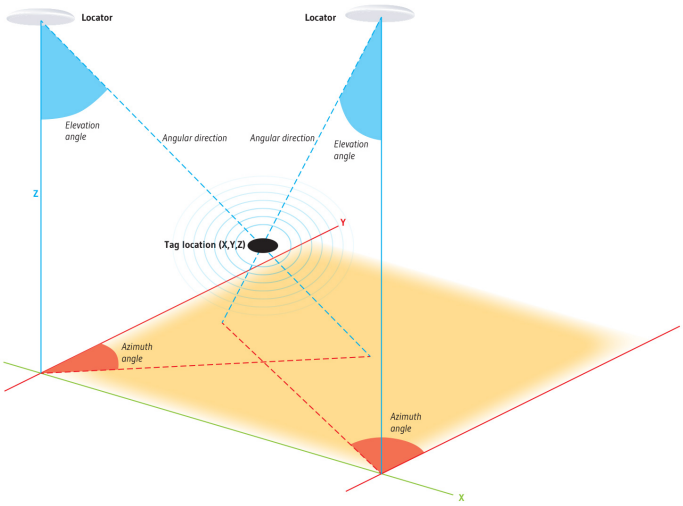

Figure 2.3: 3D location using AOA from Quuppa Intelligent Locating System (source: [15])

2.2 Fingerprinting

This method is a part of Signal Strength Fingerprint Maps (SSFM) type. Main point is using previously recorded data in the environment to figure out device location, hence the term fingerprint. There are multiple kinds of radio signal sources, such as Bluetooth, wireless or cellular devices that can be recorded.

Fingerprinting is implemented in two main phases. First, fingerprint maps construction also called offline phase. They are created by collecting Received Signal Strength (RSS) and optional extra features in known locations, the specific location is also recorded. All of these values are then saved in the database and that is called a fingerprint map. The second phase is localization itself, also known as online phase, where the device measures RSS values and compares them with fingerprint maps to approximate device position using suitable localization method [3, 16]. Commonly used algorithms and methods to approximate position are [9]

- probabilistic methods,
- k-Nearest Neighbors,
- neural networks,
- support vector machine,
- smallest M-vertex polygon.

There are multiple advantages of this approach and the most important is that it does not need any additional or specialized hardware. Next one is no need for time synchronization between the devices. Both of these advantages make it simple and cost effective method for localization. On the other hand building of maps is very time consuming and it needs heavy calibration. It is also susceptible to changes in the environment, such as people presence, object movement or relative humidity [9, 17].

2.3 Proximity

Proximity detection, also known as connectivity-based positioning, calculates only approximate location. Position is determined by cell of origin (COO) method with known position and limited range [6]. Device location is based on the cell of connected transmitting device ("associated access" point in Wi-Fi 802.11 systems) as shown in Figure 2.4 [18].

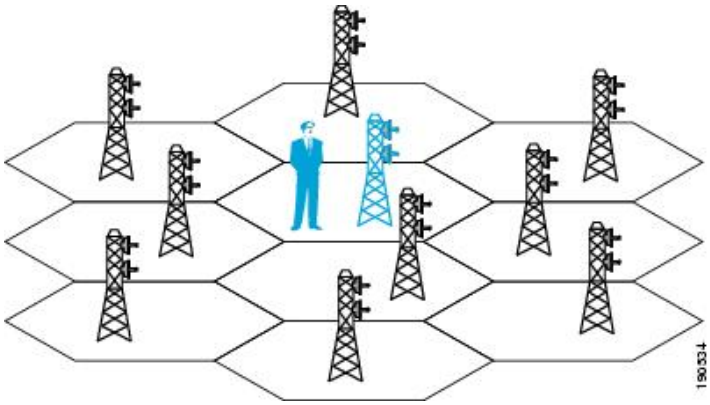

Figure 2.4: Cell of Origin (source: [18])

Main advantage of this approach is very easy implementation and no need for complicated algorithms, making calculations really fast. However, for various reasons, devices can be associated to cells that are not in close physical proximity. Such errors can happen, for example, in multi-floor buildings where floor cells overlap. There are additional methods that

can be used to improve this localization, such as using received signal strength indication (RSSI), manual method (human search) or connecting to device with highest signal strength [6, 18].

2.4 Other techniques

Scene analysis is a pattern recognition method that uses features of the scene observed from a particular vantage point to draw conclusions about the location of observer or objects in the scene [19]. This approach has been used in many cases, such as image and speech recognition as well as localization [20]. The advantage of this technique is that the location of objects can be inferred using passive observation and features. The disadvantage is the need for observer to have access to the features of environment against which they will compare its observed scene [19].

Dead Reckoning refers to a positioning solution that is implemented by measuring or deducing displacements from a known starting point in accordance with motion of the user [21]. Basically, calculating new position based on starting point, travel distance and angle of movement. New position calculations are dependent on previously calculated ones, creating the need for high accuracy of collected data since it makes errors cumulative [22].

3 Related Work

This is not a novel idea and there are already some completed solutions and papers written about RSS fingerprint collection using multiple devices. This chapter will describe few selected solutions close to this one as a comparison.

3.1 Improving Precision by Using Multiple Wearable Devices

This paper focuses on improving indoor localization using BLE-based fingerprinting with multiple devices [23]. Using combination of smartphone and wear should in this case prevent signal obstruction from human body and at least one of the devices should receive beacon signal. Unfortunately due to low BLE sensibility of wear devices, authors decided to supplement them with a second mobile device, in this case with Nexus 5 running Android version 4.4.

It proposes calculating medians from 800 millisecond tests where user can move only one meter from starting position at most. Average and variance is calculated for all medians with the same position, based on these values a normal distribution is used to model the potential variation of RSSI. Important thing to note is that fingerprint maps are also built based on facing direction. These maps are then tested using five main scenarios.

- P1: single device held in hand where body does not obstruct its line of sight (LOS) path to all beacons.
- P2: one device is placed in breast pocket where LOS may be obstructed to some beacons.
- P3: user holds smartphone in hand and wears a smartwatch on one wrist.
- P4: single device is placed in the breast pocket and the other is on one wrist.
- P5: one device is in the breast pocket, and two other devices, each on one wrist.

At first, four of these scenarios were tested in a 15x8 meters entrance hall with four deployed beacons in the corners and ten measurement positions. Using multiple devices in this location improved position precision and reduced error by 57%. Table 3.1 shows mean errors for previously mentioned cases in this location where using more devices improves localization. However there is one position where case P4 will result with higher error than P3 due to building's structure and signal obstruction for nearest beacons.

Scenario	Mean error (m)
P1	2.36
P2	1.71
P3	0.96
P4	0.41

Table 3.1: Mean errors at first location (sources: [23])

Second case, all of previously mentioned scenarios were tested in a conference room with unified ceiling and desks near the walls. This location is used to investigate the impact of beacon density to position precision. Three following combinations of beacons are used

- A: four beacons at the corners,
- B: combination A with one beacon at the center,
- C: combination B with four more beacons at the sides of the room.

	Mean error (m)		
Scenario	A	B	C
P1	2.05	2.05	1.76
P2	1.84	1.49	0.90
P3	1.36	1.09	0.63
P4	1.24	0.78	0.23
P5	0.80	0.39	0.07

Table 3.2: Mean errors at second location (sources: [23])

Using directional maps at this location resulted in increase of maximum localization error, which was not expected. This error can increase even more when using higher count of

beacons and occurs mostly when testing at the edges of the room. Mean error on the other hand shows an improvement, higher with more beacons used.

In summary, position error can be improved by three aspects: using more devices, using directional map or increasing the number of beacons. There are two main conclusions of this paper. First, confirmed precision degradation when testing near the edges of the room with obstructed signal to nearest beacons. Second, human body does not greatly change radio signal and device can receive reflected signals with sufficient strength.

3.2 SmartFix

Complete name of this paper is "An Indoor Locating Optimization Algorithm for Energy-Constrained Wearable Devices called SmartFix" [24]. The main goal was set to improve energy consumption efficiency for wearable-based indoor localization systems using WiFi fingerprinting. In the begging, single real-time experiment of energy consumption was run and split into two main parts: computation of location and collection of fingerprint. According to this experiment, energy consumption for data collection is 99% of localization algorithm.

This paper proposes novel indoor localization strategy, SmartFix, that can cooperate with an existing indoor localization technologies based on WiFi to decrease power consumption. It enhances accuracy of such algorithm with a little extra energy cost of calculation but large decrease of power consumption for signal collection. Aided with machine-learning algorithm, it obtains the relative features given by the trajectories of users in certain areas and modify the positioning results. SmartFix can save up to 70% of energy while achieving the same localization accuracy when compared to the original fingerprinting method.

To test this new system it was implemented with prototypes of TinyLoc [25], MoLoc [26] and basic WiFi fingerprinting method using K-Nearest Neighbors algorithm. TinyLoc is more focused on energy efficiency than location accuracy. In contrast, SmartFix analyses the history of people trajectory in given area to improve localization results by referring to user motion features. SmartFix then modifies positional results to achieve satisfying accuracy. MoLoc, same as SmartFix, also leverages user motion by collecting trajectory patterns using device built-in sensors to improve localization.

All previously mentioned, prototypes were deployed with and without SmartFix to test their power consumptions. This algorithm only needs a single real-time RSS signal in the locating phase to guarantee an excellent energy saving performance. Figure 3.1 shows power consumption of on-time locating on two specific devices: HTC one and Moto 360.

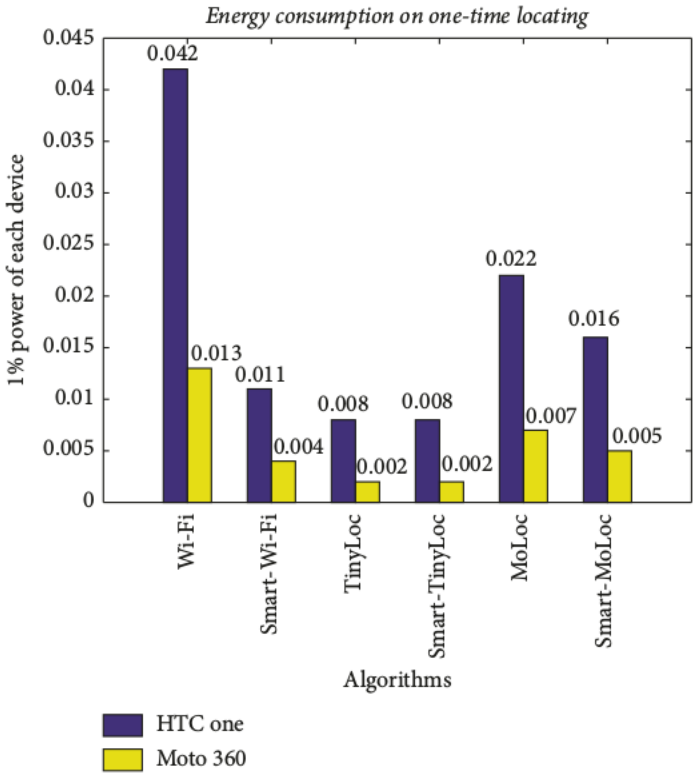

Figure 3.1: Energy consumption based on prototypes (source: [24])

This paper proposed an tested new localization technology, SmartFix, which main focus is to improve energy efficiency on wearable devices. According to the experiment, probability of error within 2 meters can be reached in 80% of cases. Meanwhile, energy consumption is 35% lower than that of MoLoc with the same accuracy. Results show that implementing SmartFix obtains the best accuracy with minimal energy cost.

3.3 Smartwatch vs. Smartphone

It is a comparative study about localization using smartwatch vs. smartphone [27] which presents that positioning accuracy using WiFi-based fingerprints implemented on smartwatch is sufficient for at least room-size locations. Average minimum room size is $10m^2$ or at least $3x3$ meters with maximum of five WiFi APs broadcasting using different channels.

Field study was conducted in six specific locations, such as farm, large room, house, two types of medium rooms and one small room. This study collected data using two Android-based devices, smartwatch MotoACTV was used and smartphone Samsung Galaxy S3 mini. To make fingerprint data more reliable the study also collects device orientation (horizontal, vertical) and data were taken in all four cardinal directions (north, east, south, west).

Test area	Floor size		Fingerprint size		Positioning accuracy SmartWatch (SmartPhone)		
	Dim.	Area	Dim.	Area	5AP	4AP	3AP
Farm	$30 \times 80m$	$2400m^2$	$10 \times 10m$	$100m^2$	82.5%(73.1%)	74.6%(69.8%)	56.2%(51.1%)
Large	$10 \times 20m$	$200m^2$	$2.5 \times 2.5m$	$6.25m^2$	41, 1%(40.3%)	36.9%(33.6%)	27.7%(28.1%)
House	$10 \times 16m$	$160m^2$	$2 \times 2m$	$4m^2$	63.9%(68.5%)	56.5%(66.6%)	47.1%(63.8%)
			$> 2 \times 2m$	$> 4m^2$	83.5%(87.5%)	79.3%(86.6%)	73.2%(81.1%)
Medium	$6 \times 6m$	$36m^2$	$2 \times 2m$	$4m^2$	67.1%(70.1%)	53.5%(67.9%)	35.0%(61.2%)
			$3 \times 3m$	$9m^2$	91.1%(96, 1%)	86.3%(95.2%)	75.7%(93.6%)
Small	$3.5 \times 3.5m$	$12.25m^2$	$1.75 \times 3.5m$	$6.13m^2$	98.1%(100%)	97.2%(99.5%)	91.3%(97.8%)

Figure 3.2: Summary table of results (source: [27])

Figure 3.2 shows the results of these experiments. Most important part of this summary is comparison of positioning accuracy between smartwatch and smartphone. In all tests the difference in classification error between the smartphone and smartwatch was at most 5% if all five APs were used. On the other hand it can be up to 25% worse when using small amount of APs in large environments. This study confirms that localization using smartwatch can be comparable to smartphone and sometimes even better for home environments. The usage of smartwatches is also preferable since it is tightly connected to a person.

4 Android

This chapter will provide information about Android and WearOS technology from the same company. Why it was developed and what are the differences between previous versions and other wear technologies.

Android is a Linux-based operating system for mobile and wear devices developed by Google. The main selling point of this system is being an open-source project, meaning everyone can access the code and modify it as they wish. Android was mainly developed for mobile platform but in time moved beyond and can be implemented into all kinds devices, such as wear, tablets, televisions and even refrigerators or cameras [47].

4.1 Android system structure

Android is created as a stack, meaning there are functional modules "stacked" on top of each other from Linux core over native libraries to applications as shown in Figure 4.1.

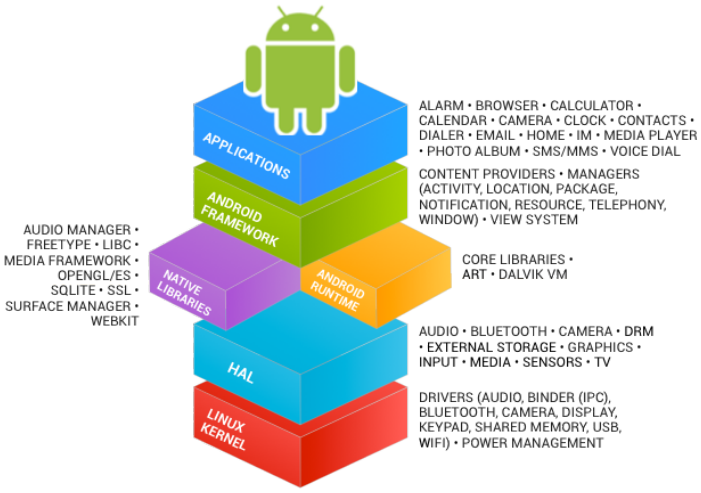

Figure 4.1: Android stack (source: [29])

Android maintains all implementations with complete software stacks to enable device creators to run and modify Android for their specific hardware. To support these modifications and testing every release has multiple "code lines" to separate stable versions from

experimental work [29]. There are multiple versions of Android system at this time and every single one has its own version information, code name and API level. Version codes are number identifications of a specific system build. Highest levels of these numbers are grouped into code names that are ordered alphabetically. For example, versions 8.0.0 and 8.1.0 have both the same code name called Oreo. Finally, API level is number identification for compatibility of specific application and is always compared to API level of device Android system [29, 30]. Devices with API levels lower than minimum level supported by the application are not able to run it.

The highest part of the Android system stack are applications which extend device functionality and are written primarily in Java or newly supported Kotlin programming language [34]. These application files are packaged into **.apk** file, which is a zip archive, containing all application files, such as Java classes, layouts, images and more. One of the very important files is **AndroidManifest.xml**, it contains all meta-data about the application, such as permissions, package name, used components, versions and so on. These application packages can be shared, for a nominal fee, via official Android market called Google Play. At the end of 2017 there were over three and a half million applications available in Google Play Store [34, 36, 37].

Application permissions maintain security for the system and users. Android requires that application declares the permissions it needs before it can use certain system data and features. Depending on how sensitive the area is, the system may grant the permission automatically, or it may ask the user to approve the request. Following example shows how to set required application permissions in AndroidManifest, Internet is granted automatically but location must be manually approved by device user.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Code example 4.1: Application permission settings

Android is a platform designed to be open-source and free which also makes it easy to create malicious applications. These application can bypass existing security and steal sensitive data, use telephone services or even gain control over the device [35]. Android has multiple ways to protect against such applications, two of the most notable ones are Android Permission Framework and Google Play Protect [34].

4.2 Wear technologies

Interactive wearable, as an example smartwatches, is a new part of mobile computers. Wear devices are categorically different from mobiles or tables in terms of usage, look and user interfaces (UI). According to the application design guidelines from major vendors users interact with wearable devices frequently throughout daily use. Each interaction is short, often less than 10 seconds, and dedicated to complete simple tasks [31].

Important thing to note is that there are multiple kinds of wear devices from watches, wristbands, cameras or even glasses [32]. Based on a report from Gartner technology research, conducted in 2017, most used wear devices were Bluetooth headsets, wristbands and smartwatches, in this order [33]. Thanks to their small size, wear devices are ideal to use for hands-free communication and health monitoring.

Two problems of these devices is their diversity in hardware and more importantly in software compatibility. Every device manufacturer can create their own operating system for specific wear device and it can be difficult to develop custom applications for them. To avoid such problems this thesis is focused only on smartwatch with WearOS operating system which is based on Android.

There are three main points to mention considering wear devices. First, small battery capacity that can be almost ten times smaller than of typical smartphone. Second, small display size with around forty-times less pixels which completely changes its properties. Finally, scaled down CPU with high efficiency [31]. Last two points are main parts of lowering power consumption of smartwatches but even with these cuts high-end watch devices can have really small battery life only in matter of few days or even hours.

4.2.1 WearOS by Google

WearOS by Google, formally known and Android Wear 2.0, is a version of Android operational system tailored to small-screen wearable devices. There are not too many in-system changes from smartphone but one of the main differences can be seen in UI since system had to be adjusted for watch size [38]. Due to scaled down processing power of watches can offload data wirelessly to mobile for heavy computing tasks, e.g. voice recognition [39].

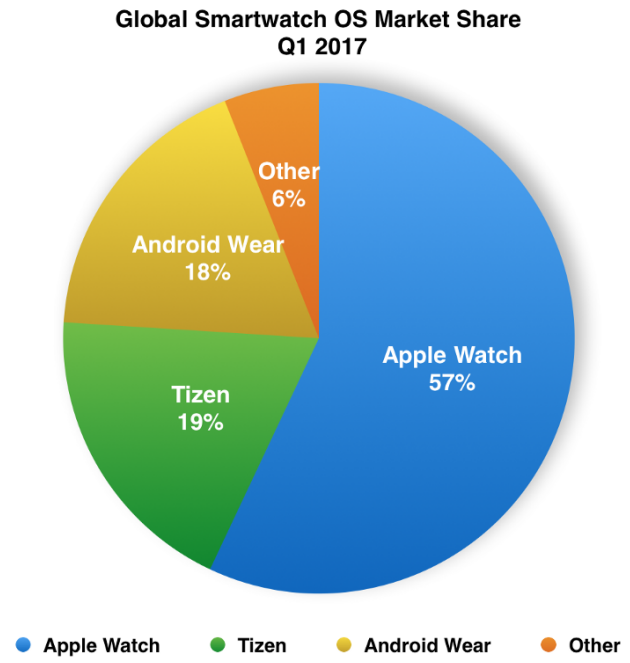

Figure 4.2: Smartwatch OS market share (source: [42])

WearOS is one of the three most popular smartwatch systems but it comes with its own set of problems. Most notable and annoying one is being unable to pair wear with specific mobile device. It can be caused by multiple reasons, such as system compatibility, custom hardware or mobile type and it is more common than it should be [40]. Having smartphone connected to WearOS can also cause faster battery drain on mobile device. Important thing to mention is that smartwatch can be connected to only single device and connecting to another mobile requires watch factory reset. Some other problems can be: software update issues, notifications not coming through to the watch, not being able to connect to WiFi, system crashes and few more [41].

Even with all these problems it is popular system and in early 2017 it got its biggest update yet. New version 2.0, later renamed to WearOS, which brought numerous improvements and features where few of the most notable ones will be described in this section [43–45].

4.2.1.1 Standalone applications

This feature is a crucial change and it enables smartwatch applications to run without the need for mobile device to be constantly connected. Before this version it was needed to have smartphone connected to Android Wear when application was supposed to be used. It also supported only connection to Android device and that proved as an obstacle for users without one since they could not use any applications on the watch [43, 44].

This change made applications work without mobile but that also created the need for them to be installed directly on wear without the need for controlling device. Thankfully part of this update is also standalone Google Play Store where users can browse applications that are designed specifically for the smartwatch [44]. Part of this feature is also enabling watch to use wireless and cellular networks on their own since most standalone applications require at least WiFi connection. Finally, as of this update, security for communicating between wear and mobile was reworked and improved. This communication is now done via Wearable Data Layer API that is used in almost all Google applications and it is also easy to use for a developer [43].

4.2.1.2 UI improvements

Part of new Android Wear version is implementation of Android's Material design guidelines [46]. It has much more "mature" look and darker design for reducing battery drain [44]. It is completely focused on wear devices and supports both round and square screens with new re-design of application launcher [43].

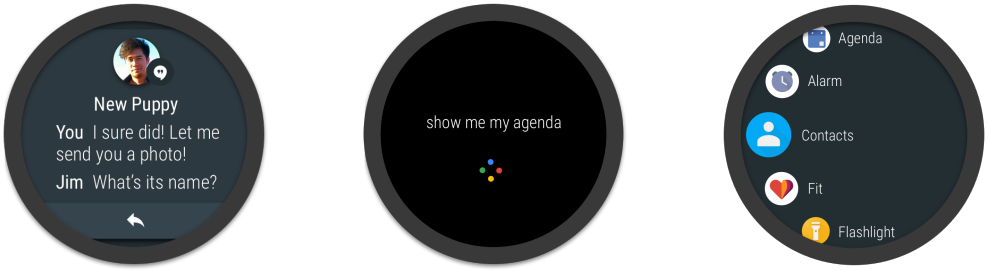

Figure 4.3: Wear design examples (source: [46])

Android is also trying to catch up with Apple's watchOS and that is why they remade default watch display, also called watch faces, and made it much more useful. Users can now add different widgets with data from any application to the watch faces [43]. This ensures quick access to the information user deems important [45]. All this data displays also match design of currently selected watch face and after clicking it user is directed right into the application [44].

Figure 4.4: Wear watch faces (source: [43])

4.2.1.3 Google Assistant

Google Assistant is basically voice controlled smart assistant, such as Amazon's Alexa, Apple's Siri or Microsoft's Cortana. There are multiple tech sites that run benchmarks of these systems [48–51] and they do not seem too much different, so there is no actual need to buy one over the other. These systems can pull information that user needs or wants and they can also track important information, such as place of work, sport interests, daily schedule, data related to health and much more [47]. With the update of WearOS this assistant is now also available on smartwatches [43, 44].

4.3 Other wear technologies

During the first years of smartmatches big amount of manufacturers created their own operating systems and flooded the market with them but that was a long time ago and situation has changed since then. Now there are only three main competitors as shown in the Figure 4.2 and those are Android (WearOS), Samsung (Tizen) and Apple (WatchOS). System from Android was already introduced so it is time for Tizen and WatchOS.

4.3.1 Tizen

Between year 2010 and 2012 there were multiple companies developing different wear systems, for instance: MeeGo by Nokia and Intel, Samsung Linux Platform (SLP) or Bada created by Samsung. Intel later decided to join forces with Samsung and they created Tizen organization and project which was based on SLP. This project either canceled support of previous systems or merged with them [52]. Some of the current members of Tizen organization are: Samsung, Intel, Huawei, Vodafone or SK Telecom [53].

Tizen system is built from the ground up on Linux platform and is a part of Linux Foundation. It is focused on the needs of all stakeholders of the mobile and connected device ecosystem based on the hardware on which it should run, such as mobile, tv and wearable. One of this systems main focuses is easy support for different manufacturers with changeable profiles to suit the needs of a specific manufacturer. Tizen also offers the power of native code development with the flexibility of unparalleled HTML5 support [54].

Figure 4.5: Tizen showcase (source: [60])

Application development for this system can be done in a custom made program Tizen Studio which supports developing native or Web applications where native apps are developed using the C programming language, but since Tizen's version 4.0 application can also be developed using .NET or Xamarin UI. This studio contains all required parts for developing an application from project management, writing and editing code to debugging and running the application. Newest version of Tizen Studio is 2.3 with the support of Windows, Ubuntu and macOS where the next version will remove support for 32-bit Windows and Ubuntu. Created applications can be deployed to Tizen store which is very similar to Google Play for Android system [55].

Even though wear version of Tizen OS is deployed only on Samsung products it overtook Android Wear in market shares in the beginning of 2017 but that might change with the release of Android Wear 2.0 (WearOS).

4.3.2 WatchOS

Apple started working on smartwatches around year 2002 inspired by sportswatches from Nike, with the actual results few years later (2014), when Apple Watch was revealed to be released in 2015. The release of this smartwatches introduced a new version of iOS system focused solely on wear devices called watchOS. Compared to previously mentioned systems

this one is the youngest but most popular of them [56]. WatchOS is currently developed and deployed only for Apple Watches, same as Tizen for Samsung devices, making Google's WearOS only one of them to be implemented by multiple wear manufacturers. First version of watchOS was based on the same principle as Android Wear because it required mobile device to use its applications but luckily for Apple, they quickly moved from this interaction in version 2.0. Next versions brought many improvements, such as running application in the background, improved design and support of completely mobile independent applications in the newest major version 4.0 [57].

Applications of this system consist of two related bundles: Watch app and WatchKit extension. The Watch app bundle contains the storyboards and resource files associated with application's user interfaces. The WatchKit extension bundle lives inside the Watch app and contains the code for managing those interfaces and for responding to user interactions [58].

Figure 4.6: WatchOS showcase (source: [59])

Application development for WatchOS can be done using Xcode which is a free program to develop applications for iOS platform created by Apple. Main programming language used in this program is Swift, also created by Apple and based on C and Object-C language. Swift is an open-source language focused on being easy to understand but still powerful and fast. Same as previous wear systems, created applications can be deployed using an App Store similar to Google Play or Tizen store.

5 Application design and implementation

This chapter describes all important information about created application. First, description of hardware devices used for developing this application. Second, server used to store data on. Third, software libraries used in the application. And finally, implementation of core parts with description of selected code examples.

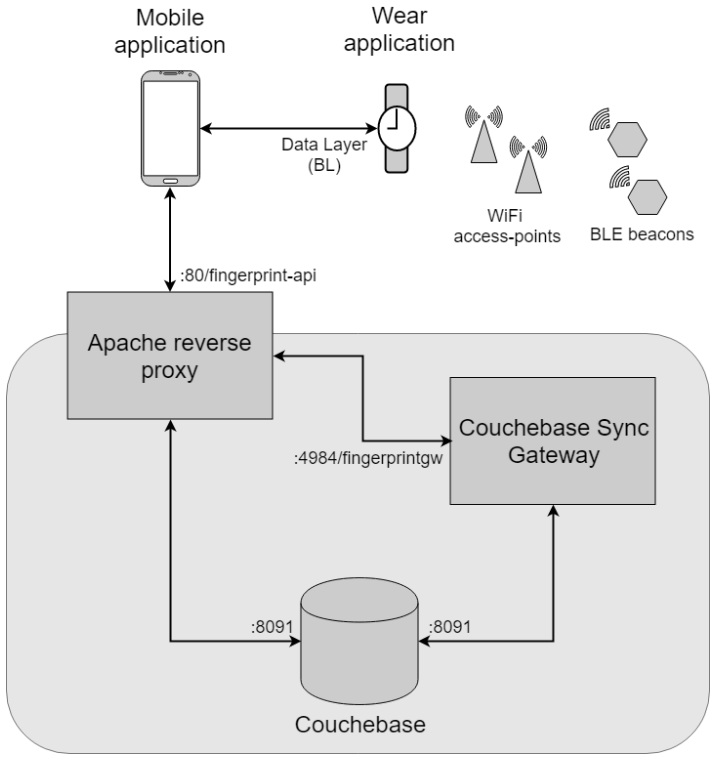

Figure 5.1: Application architecture (based on [9])

Figure 5.1 shows core devices and technologies used in this application. There are three main devices of this implementation, mobile, wear and the server. Mobile and wear parts implement scanning for radio signals and they are written in Java language for Android system. Server is also programmed in Java to keep implementations close to each other and make future adjustments easier.

5.1 Hardware

There are multiple kinds of hardware devices used in this application. All of them can be seen on previous Figure 5.1, where first and the most important are smartphone and smartwatch. These devices scan for radio signals from WiFi access-points and BLE beacons that are placed in the indoor environment. Final hardware part is the server containing scanned data from all devices and supporting synchronization between them.

5.1.1 Smartphone and Smartwatch

Both smart devices must support scanning of Bluetooth Low Energy (BLE), which is supported from Bluetooth 4.0, and WiFi signals. Secondary requirements are GSM and LTE modules to support more data types than BLE and WiFi.

5.1.1.1 Smartphone

Main part of the application is developed and tested on Redmi Note 4 from Chinese company Xiaomi. It is running customized version of Android 6.0 called MIUI. Even thought system was customized, in core it is still Android so there are no problems in that regard [62]. This phone has Bluetooth 4.1 with LE support and WiFi 802.11 a/b/g/n chip, thus main requirements for the hardware are met. Concerning secondary requirements, all them are also met with GSM and LTE support like most modern smartphones do [61].

One interesting thing about Xioami smartphones is their locked bootloader. It prevents users from any manual updates but most importantly from reseting the device to factory settings. For unlocking bootloader owner has to create an account on Xiaomi website and put a request to unlock it, which is usually processed within two weeks period and there is no actual guarantee of it being approved. After evaluation process is complete user will be notified via sms about the result of such request.

5.1.1.2 Smartwatch

Before development for wear device could begin it was required to find suitable smartwatches running Android Wear 2.0 system. This iteration was quite new at the time, which made it harder to select such device because there were not so many options to choose from. During selection process there were around twenty smartwatches with this system and only five of them were selected for a closer inspection based on few technical articles [63–65].

Watch	BLE / Wi-Fi	Czech Republic	Problems
LG W280 Sport	Yes / Yes	No	Battery life is one day or less. Too big in size.
LG W270 Titanium Style	Yes / Yes	Yes	Battery life is one day or less.
Huawei Watch 2	Yes / Yes	Yes	First update can take a long time. Slight Bluetooth pairing issues.
Polar M600	Yes / Yes	Yes	Polar support complains. Phone synchronization issues. GPS location malfunctions.
ASUS ZenWatch 3	Yes / Yes	No	Strap breaks fast. AW 2.0 update can break the watch. ASUS support complains.

Table 5.1: Smartwatch comparison (sources: [66–70])

Only one smartwatch could be selected out of the five displayed in Table 5.1. Selection was made based on hardware requirements, website articles and customer experience. First, wear device had to support BLE and WiFi, which all of them do. Second, it must have been sold in Czech Republic (CR) because it makes shipping and warranty dealings easier. Only three of five devices were sold in Czech Republic at that time so others were taken out of the consideration. Final decision was made based on extensive research of customer reviews in shopping sites, such as Amazon, CZC, Heureka, Alza, wear official websites [66–70] and other tech sites [63–65]. In the end, selected device was Huawei Watch 2 since requirements were met and there were not too many problems found in reviews.

Initial setup of the wear device was composed of two main parts. First, update wear system which took about one to two hours. Second, setup the watch and copy Google account information into it, this is where some problem were encountered. Copying of accounts from Redmi Note 4 to the watch never completed successfully. To fix this problem another smartphone (Huawei Y5 II) was used to copy the account and as it was already mentioned, only single device can be connected to smartwatch. Connecting to new smartphone forces a factory reset and removal of all data from wear device. Luckily, there is a workaround for this issue handled via Android Debug Bridge by following this [71] article. In short, it resets the information about connected mobile device and starts new Bluetooth connection search.

Since Google wants to focus also on iOS device technology, Android Wear 2.0 was rebranded to WearOs by Google to remove confusion in the future [72]. This forced an update on wear devices, updating it to the newest Android version which resulted in some changes during development process. First, an implementation bug in scanning library was introduced and had to be fixed. Second and bigger problem was the inability to deploy new

version of the application from Android Studio to the watch. This, for some reason, effected only the main computer used to develop this application and it was fixed by using another one.

5.1.2 Radio signal devices

Hardware devices used to send out radio signals are BLE beacons, WiFi access-points and Cellular towers. Only first two types of these devices can be placed inside of the building and are most commonly used for indoor localization. Signal from cellular towers is only a complimentary information that does not have to be used and cannot be influenced since they are placed outside by telecommunication companies.

5.1.2.1 BLE beacons

Beacons are small devices that can be easily placed in almost any environment. Only thing they do is send an information packets using Bluetooth and nothing else. They are commonly used in museums, airports and as of late for indoor localization [74]. Beacons have their own battery powering them which can last around a year or two without charging because of the new Bluetooth Low Energy (BLE) standard. This technology drastically reduces power consumption and introduces new configuration options regarding advertising interval and transmitter output power [73]. More in depth description of this technology and devices was written by Pavel Kriz et al. [9].

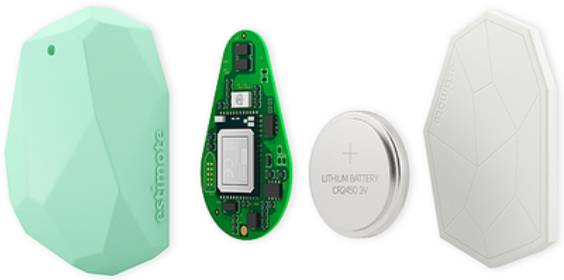

Figure 5.2: Parts of Estimote beacon (source: [76])

There are multiple beacon manufacturers, each with their own quality, signal strength, battery life and other hardware differences. Changes can also be found in software (packet) specifications for beacons, meaning data they send have different formats but it does not

mean other systems cannot see them. Some software changes are usually have to be made to detect different kinds of beacons but they are documented and not hard to make. Beacons are also usually platform independent, making them work with multiple systems, such as Android or iOS [73, 74]. Beacons used in this thesis are from a company called Estimote and they are most commonly used.

Another important thing to mention, beacons are not connected to the Internet and do not collect any data from devices around them. That is why all data have to be processed in another device, most commonly a smartphone, but it also makes beacon safe to use because there is no need to worry about the theft of user sensitive data [74].

5.1.2.2 WiFi access-points

WiFi signals are most common devices used in indoor localization due to their presence almost anywhere and high precision. In this case, several WiFi transmitters of the **eduroam** network made by Cisco were used. This network allows users (researchers, teachers, students, staff) from participating institutions to securely access the Internet from any eduroam-enabled institution [75]. Access-points are permanently deployed on every floor of the Campus building and cannot be influenced or have their settings changed, they are also usually at least two per location to enable transmitting in 2.4 GHz and 5 GHz band.

5.2 Server

This application does not necessarily need a server for its core functionality but it is common and faster to run an analysis of collected data and other heavy computational work on a different and usually faster device. Functionality of the server is to provide a backup of the data and also work as a synchronization point to enable other devices to download fingerprints and upload newly collected ones.

The server infrastructure was built in previous years [9] but it was modified for this solution. In its core, server uses NoSQL database to save fingerprints. This type of database does not have a fixed scheme, making it easy to save all kinds of data without any constraints. There are many implementations of NoSQL databases, currently over 225, to choose from. Selected database is called Couchbase, it saves data in JSON files with its very own query language similar to SQL. Part of this database can also replicate data between other devices which could be used in the application but it was deemed too slow and data heavy for use in this application. Not using this feature created the need for development of a middleman

API for communication between Android application and Couchbase database.

As Figure 5.1 shows, server has three main parts to work with. First, already mentioned Couchbase database to keep scanned data. Second, sync gateway that can synchronize data between devices, which is not used in this implementation but there is a connection to it for specific reason described later. As it was mentioned, these two parts were already implemented in previous years [9]. Final part is a newly created web application working as a REST API to send data between the server and mobile application.

This API is written in Java-based framework called Spring to keep the code similar to Android. Before implementing, it was documented using Swagger and it handles two main HTTP routes: **fingerprints** and **fingerprints-meta**. Mobile application checks meta data on the server before any tasks can be run. It is done by calling **fingerprints-meta** route which returns count of new fingerprints and last time current device saved new ones on the server. With this information the application calculates how many fingerprints to upload and download where both of these tasks run in parallel to speed up the process. To prevent device memory exhaustion and connection timeouts both of these tasks have specific limits, download can process 100 of fingerprints while upload only 20 per call. Server can also generate data files for analysis, split by technology, time, origin device and other filters. To simplify this generation it is similar to loading of all fingerprints, it does not create file with the data but prints the them directly into the connection stream.

One last thing to note in Figure 5.1 is the connection with sync gateway. Even though it is not used in this implementation for synchronization, it is used to save fingerprints into the database. When data is saved into the Couchbase directly, sync gateway will not be able to display them, that is why the upload goes through the gateway. It ensures all previous and next solutions will be able to use this feature without loosing any important data.

5.3 Application Software

Software part for smartphone and smartwatch is built on Android system which was already described in Chapter 3. This section will provide basic information about libraries, technologies and systems used in these applications.

5.3.1 AltBeacon Library

Android core does not allow scanning for BLE beacons, that is why this feature must be implemented in a library extending system features. There are multiple solutions that can

be used to scan for beacons such as Estimote SDK [77] which was already used in previous thesis [78]. To change things up BLE beacons are scanned using AltBeacon Library [79].

There was no open and inter-operable specification for proximity beacons at the time and Radius Networks decided to create the AltBeacon specification as a proposal to solve this issue. It is an open and free specification for BLE beacons with the focus to create an open and competitive market for implementation of these devices [80]. Basic configuration of this library can scan for beacons based on this standard but it also supports Eddystone beacons which is Google's open-source format. To support, already mentioned and used Estimote beacons, this library configuration must be altered to support their detection. Luckily, scanner function can be easily modified to support different kinds of beacons by adding just one line of the code [79, 81].

```
beaconManager.getBeaconParsers().add(
   new BeaconParser().setBeaconLayout("m:2-3=0215,i:4-19,
      i:20-21,i:22-23,p:24-24"));
```

Code example 5.1: Code to enable all beacon types

This library uses publish-subscribe design pattern, meaning one scan data is send to multiple clients if they listen for them. Using this approach has an advantage of eliminating the need to run scan per client. Update to Android 8 introduced a bug in this feature making it unable to confirm the registration of a client so the scanner had to be reworked to work around this bug, thus postponing the data collection and analysis.

5.3.2 Database

This solution makes use of two different types of databases to store all the fingerprint data for calculations. First, SQLite database implemented in Android mobile application to save fingerprints from smartphone and smartwatch. This database is a default implementation and most commonly used in Android. Second database type is Couchbase, implemented on the server **beacons.uhk.cz** to keep all fingerprints in one place and also enable synchronization plus data analysis.

5.3.2.1 SQLite database

SQL (Structured Query Language) is a standard language for storing, manipulating and retrieving data in databases. It is a type of relational database, meaning all data is saved into tables with specified rows and columns [87]. These tables usually have a set amount of rows with specific names that protect from adding wrong data, for example, you cannot add data

`Person(name, surname, eye color)` into table `Person(name, surname)` because there is no column named `eye color` in the table.

Structured data is one of the advantages for this database type and it makes calculation faster but as a drawback it uses more storage space. Other advantage is that the data can be saved only once because they can be connected to each other. It supports complex queries for creating, reading, updating and removing data (CRUD) and better security with user and table management. Some disadvantages of this system can be in complexity and inflexibility of database schema because it is hard to setup and does not allow saving of data not defined in the tables [86]. Altering schemes in the future can also be really complex since there is the need to parse current data into newly created tables which do not need to have the same schema as current ones.

Since SQL with all its features can consume a lot of hardware resources for a smartphone, Android decided to implement lite version of this database. SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional [87].

- Serverless = does not need second process for the server.
- Self-Contained = requires minimal support from operating system.
- Zero-configuration = no need for installation or any configuration.
- Transactional = data are protected against failed changes (application crashes, power failure, malformed data and more).

5.3.2.2 Couchbase database

SQL-based databases can be complex to implement, scale and usually require more data storage space. This created a motivation to develop so called NoSQL databases. Their most important and significant feature is not having a fixed schema, that makes them easy to scale and replicate between multiple devices. There are over 225 NoSQL databases at this time and selected Couchbase is one of them [83].

Couchbase is distributed, document-based database with its own querying language called N1QL. It is a database focused on simple server configuration and easy usage for clients, with built-in caching layer and distribution system it does not require any changes in the application. There can be either one server instance of Couchbase or multiple connected effectively creating a database cluster which holds all the data in multiple locations (nodes) [84]. This data is saved in JSON file format and usually in readable form without any encryption.

```
[
   { "id":"1", "name":"Joe", "lastName":"Doe", "address":{} },
   { "id":"2", "name":"James", "lastName":"Named", "address":{} }
]
```

Code example 5.2: JSON format example

Since SQL language is used for decades and it became the standard for working with data in databases, Couchbase embraces this approach and extends it for JSON files, creating a new language called N1QL. It has all the main features of the SQL with some minor improvements [85]. Currently this language can be used only on the server implementation and it is not supported in Android. Version of this database for mobile application is called Lite and so called **views** are used instead of N1QL. Views are objects containing selected data from database documents and their major problem is being very data heavy which is shown in the following section. This makes it one of two main reasons why mobile application uses an SQLite instead of Couchbase. Second point is to differentiate between previous solutions and test data consumption and loading speed of SQL database.

5.3.2.3 Comparison

Both of previously mentioned database solutions were tested in Android mobile application to figure out which one is faster and takes less data storage space. As a test, 315 fingerprint documents were loaded and displayed, all of them having more than 500 subdocuments which makes about 150 000 documents in total. As shown in the Table 5.2, SQLite takes less storage space and is almost three times faster in loading all the documents, that is why it was selected for this project. If Couchbase Lite would have faster query time or supported the use of N1QL it would be a preferable solution but it is not at this time.

Database type	Data size	Loading speed (315 documents)
SQLite	15MB	23 second
Couchbase without views	31MB	65 seconds
Couchbase with views	91MB	65 seconds

Table 5.2: Couchbase vs SQLite (sources: [66–70])

Selecting SQL comes with a disadvantage since it does not provide any data synchronization, due to that, a custom solution must have been created which was already described in the Server section.

5.3.3 TileView

There are multiple ways to display image map in Android, default solutions usually load the whole picture at once. This works for smaller pictures but it does not necessarily work for bigger ones because the application can run out of allocated memory. To solve this problem it is usually better to try external library or widget created specifically for displaying such images. One approach is to use 2D or 3D library to display image, these libraries usually work with the image as a whole and implement complex functionality to display it without memory exhaustion. The other approach, also used by Google Maps, is to cut the big image to smaller parts, called **tiles**, display them next to each other and recreate overall image. This approach has multiple advantages.

- Solves performance issues and enables cartographers to create aesthetically pleasing maps without the need to worry about performance impact.
- When user is panning, relevant pictures stay displayed while the others are loaded, thus improving user experience.
- Every zoom level has its own collection of images, this makes it easy to keep a good image quality for all zoom levels.
- When zoomed in, pictures out of the screen do not have to be kept in memory, this cannot be done with other solutions because image is not split into parts.
- Zoomed image can have more details than original one.

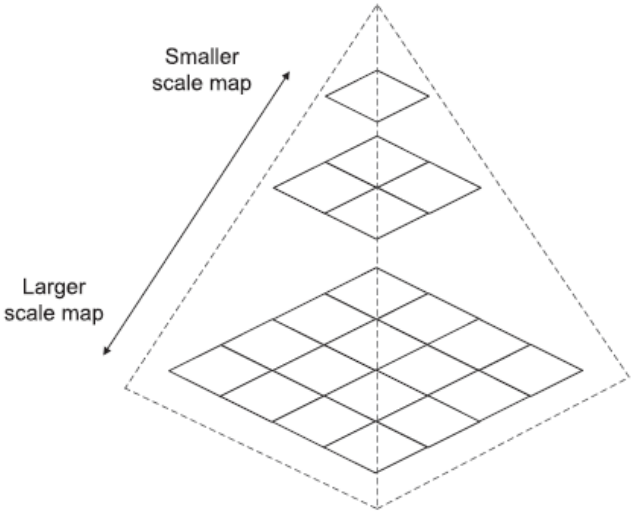

Figure 5.3: Tile pyramid for zoom levels (source: [88])

Tiles are images that have 265x265 pixels but that is not the requirement. There is one tile with the image displayed for the lowest zoom, level 0, which is usually not used and level 1 is used instead where each zoom increase splits every tile image into 4 more tiles. For practical purposes, maximum zoom level is usually around 21 with almost 4.5 trillion tiles.

Creating tiles by hand can be really time consuming job, even for low zoom levels, but it can also be done automatically using script or a program. For example, it is possible to create them by slice tool in PhotoShop, but this solution supports only low levels of zoom since it has a limit for amount of such created images. Author of this library provides a solution to this problem using a free tool called ImageMagick, which can resize, crop, cut and create images using a command line making it easy to develop a script for all zoom levels required [89].

```
C:\*path to imagemagick*\convert C:\*path to big image*.png
   -crop 256x256
   -set filename:tile "%%[fx:page.x/256]_%%[fx:page.y/256]" +repage +adjoin
   "C:\*path to tile folder*\tile-%%[filename:tile].png"
```

Code example 5.3: Creating tiles using ImageMagick

This code will split image.png into 256x256 tiles, and name them “tile-i-j.png”, where “i” is the column index and “j” is the row index. More information can be found in the TileView wiki [90].

5.4 Application implementation

Implementation of this application is split into two core parts, mobile and wear. Both of them have to implement two tasks in order to be able to create fingerprint maps. First, a scanner for radio signals which will record Bluetooth Low Energy beacons, WiFi access-points and Cellular towers. Second, communication between both implementations, mobile and wear, to allow sending fingerprint data between them. Those are two core functions for both applications but since the synchronization server is used, mobile part also has to communicate with the server.

Both of these applications are programmed in Java language using Android Studio and each of them has a separate configuration and is aimed to different system versions.

5.4.1 Scanner

Scans for radio signal and creates a fingerprint with specific position, device, radio signals and selected sensor information. This additional data, especially about the device, will help with analysis by splitting data into specific groups. There are multiple parts to this task and most of them are very computational heavy so it is better to implement it in a separate **Thread** to prevent application from freezing. Android provides multiple solutions for running tasks in separate threads.

- AsyncTask - simple solution, mainly focused on short running tasks (in seconds). This implementation is easily created since it handles all basic functions like running task, run code before and after task, publish progress to main application thread and many more.
- Service - mainly focused on long running tasks which are run immediately after it is started without checking device resources. Android wants to deprecate this approach and develop their new solution, JobScheduler.
- Custom thread - one of the hardest solutions to implement because developer has to handle all the tasks related to threads life-cycle and be careful to prevent creation of multiple unwanted threads that might freeze the application or even operating system.
- JobScheduler - One of the newest implementations, it is focused to intelligently schedule tasks and run them when the device has required resources ready.

Scanner is run using JobScheduler, which is a new technology and Android wants to continue developing this feature. Handling scanned data is implemented using publish-subscribe design pattern handled by **BroadcastReceiver**. There are two separate receivers, one for Bluetooth Low Energy and second for WiFi data, cellular scanner is connected to WiFi receiver and sensor scanning uses different implementations.

5.4.1.1 JobScheduler

This feature was introduced in Android 5.0 and it remains under active development with major changes in Android 7.0 version. It is a platform collecting information about jobs that need to run across all the applications. Some of the collected information about specific jobs are: required network access, is the job periodic, should the job be re-scheduled after device restart and many more. This information is used to schedule and run jobs at, or around,

the time they were scheduled. JobScheduler is intelligent about running jobs and it uses so called batching, which combines multiple jobs to reduce battery consumption [30, 91].

```
// Building job to run
JobInfo.Builder jobBuilder = new JobInfo.Builder(FingerprintScanner.JOB_ID, new
    ComponentName(getPackageName(), FingerprintScanner.class.getName()));
jobBuilder.setOverrideDeadline(1000);
jobBuilder.setPersisted(false);
jobBuilder.setExtras(bundle);
// Run created job
JobScheduler jobScheduler = (JobScheduler) getSystemService( Context.JOB_SCHEDULER_SERVICE
    );
jobScheduler.schedule(jobBuilder.build());
```

Code example 5.4: Schedule Firngerprint scanner job.

This code is used to schedule fingerprint scanner. Firstly, a job has to be created with specific identification and class to contain job code. Secondly, job information are provided for the scheduler: this task should start around a second after scheduling, it is not run after device restart and there are custom data send to it via `setExtras()` function. Finally, job scheduler is loaded, job constructed and scheduled to run. One thing to remember is to not create a new instance of JobScheduler class because it is a system service.

5.4.1.2 BroadcastReceiver

Broadcast messages are send from Android system and other Android applications, similar to the publish-subscribe design pattern. This messages are send when an event occurs, such as system boot up, start of device charging cycle or network connectivity change. In addition to system events, custom ones can be created in the application to inform other applications, including itself. When a broadcast is sent, the system automatically routes it to application that have subscribed to receive that particular type of broadcast [30]. Following code illustrates how easy it is to send a custom broadcast identified by specific action, which can be either system or custom. It sends BLE beacon data to be received and parsed into custom beacon entry in the scanner. Last line of the code sends broadcast to be received by other applications.

```
Intent intent = new Intent();
intent.setAction(ACTION_BEACONS_FOUND);
intent.putParcelableArrayListExtra(ACTION_BEACONS_DATA, foundBeacons);
sendBroadcast(intent);
```

Code example 5.5: Send broadcast with BLE beacons found.

Receiving broadcasts in application has two main steps. First, register the receiver implementation and specific broadcast message it should receive. This can be done via Android manifest file or dynamically in the code. If receiver is registered using manifest, application is also launched, only if it is not running already, when the broadcast is sent. Second, create an implementation of BroadcastReceiver class that will handle broadcast information and data using `onReceive()` function.

There are two receivers used in the scanner. First, used for WiFi scanning since it is a default Android solution. Second, BLE scanner was modified to use this approach because there are multiple parts of this application that can scan for surrounding beacons. Other two scanners use **Listeners** which are interfaces handling communication between system and application.

5.4.2 Device communication

In this application device communication is used to send fingerprint data between devices using Bluetooth. Mobile application sends information about newly created fingerprint such as location and scan duration, these values are the same for both devices. Wear application will, in return, send result data after scan was finished to save it into the mobile database.

First implementation of this feature was build upon Bluetooth chat example application from Google. This implementation was using three separate threads to make a connection, authorize it and then to send the data when devices were connected. Both of the devices had to accept the connection to send the data, which is normal, but there were some connection loss issues where one of the devices was considered connected and sending data while the second device was disconnected and not receiving anything. That is the main reason why this solution was removed at early stage of the development and substituted with Google's Data Layer API.

To limit power consumption of this application on wear it does not run constantly. Before each scan is started, mobile informs wear to start the application, after startup, confirmation is send back to mobile to initiate sending of fingerprint data. After wear receives the data it starts a new scan for radio signals and later returns the fingerprint back to mobile right after scanning finishes.

5.4.2.1 Data Layer API

Data Layer is commonly used by Google to send data between their applications. There is an implementation built completely on this feature called Google Tag Manager which sends

data from websites to other Google-based tools like Analytics or Adwords. In this case it is a JavaScript object or variable creating virtual "layer" of the website application which contains various data points, making its name, Data Layer [92].

Since it proved as a useful solution it was also implemented for Android and is currently a part of Google Play services. The Data Layer API allows to store and retrieve data between different devices and also kinds of devices, it this case between mobile phone and wear. There are multiple ways to send data depending on what type of data should be sent [30].

- Data Item - provides a data storage that is synchronized between devices. Whenever data changes, all of the devices using this item are then informed about the change. All data are identified by its own specific Uri containing creator device and path.
- Asset - used to send binary blobs of data, such as images. It takes care of the data transfer automatically using caching to avoid sending the same data multiple times.
- Message - good for sending small amounts of data or remote procedure calls, such as controlling mobile media player from wear. If the device receiving data is connected, sender receives a result code confirming successful data send. Although the result code does not guarantee delivery of the message and if the application requires data reliability it should use Data Items.
- Channel - transfers large data entities, such as music and movie files. It saves the disk space over Data Item or Asset because it does not create a copy of the message on local device before synchronization. It can also be used to transfer streamed data, such as music pulled directly from the network.

This application uses message system and data items, where messages are used to send information for wear application to startup and confirm if it was started. Fingerprint data are then send via data items. Since application may be either in the foreground or background there is a need for two solutions to receive the data. First, service that runs when the application is not started or in the background. Second, any activity can implement an interface that will get received data in the foreground.

One important thing to mention is, to send the data successfully between the devices they must have the same APK signatures, meaning they must be signed by the same key. For example, debug versions of two applications developed on two different computers will not be able to send data due to this restriction.

5.4.3 Server communication

Main task of communicating with the server is to synchronize data on the mobile, downloading previous fingerprints and upload newly scanned ones. This feature uses server API to transfer data and since it is a very data and computational heavy, it is implemented in a separate thread, same as a scanner using JobScheduler. It was already mentioned that server API is documented in Swagger, this tool can also generate Android code to communicate with the server, based on defined functions in the documentation. This will generate working code without the requirement of using secondary library but the code is unnecessary complex and so this application uses external library called Retrofit.

Even though the new version of Google's WearOS makes this feature easier to implement on the wear device, it is still used only in the mobile application for one simple reason: to lower power consumption.

5.4.3.1 Retrofit

Retrofit is a library turning HTTP API into a Java interface making the implementation easy and simple. This interface contains calls with parameters based on the API documentation, it can return either raw response or automatically convert the data into Java classes based on the configuration. Creating the interface is only one of two parts to make this library working. Second is to create an instance of this library's main class with a configuration specific to the API and use it to initiate calls.

```
public interface ApiConnection {
   @GET("fingerprints")
   Call<List<Fingerprint>> getFingerprints(@Header("deviceId") String deviceId,
      @Query("timestamp") long timestamp,
      @Query("limit") int limit,
      @Query("offset") long offset);
}
```

Code example 5.6: Retrofit interface example

Previous code contains a small example of Retrofit's interface class with a function for getting fingerprints. First, Retrofit must be informed which type of HTTP call it should use, such as GET, POST, PUT or DELETE. Part of this information can also be URL path added to the call, **fingerprints** in this case. Secondly, all of the parameters with information where they should be posted, in header, body or query.

```
Retrofit retrofit = new Retrofit.Builder()
   .baseUrl("http://beacon.uhk.cz/fingerprint-api/")
```

```
.addConverterFactory(JacksonConverterFactory.create())
.callbackExecutor(Executors.newSingleThreadExecutor())
.build();
```

Code example 5.7: Retrofit configuration example

There is only one required configuration to make Retrofit working and that is defining the base URL so the library knows where to send the calls. Some other configurations might be adding data converter which converts the data from Json to Java classes, run calls in different threads or create custom HTTP client.

5.4.4 Application screens

Mobile application has three screens, map and scanner, surrounding devices, synchronization screen and wear application has only one containing information about current scan. All of these screens are made to as simple for use as possible.

5.4.4.1 Map and scanner

This screen is the core of mobile application and it shows the map of a building floor with fingerprint positions as markers. Map is implemented using TileView library previously described in this chapter. It is used to display locations of fingerprints and enable to create new ones, provide data about existing ones and enable to delete them. Deleting was not supposed to be part of this implementation but it proved useful when one or both devices collected "broken" fingerprints, which are usually created by failure in WiFi scanning or failing of sending data from mobile to wear device. Deleting removes last created fingerprint group at specific spot, meaning it deletes one fingerprint per device type. This is possible because fingerprints have their specific scan id connecting them together.

Figure 5.4 shows two screens of the map with markers, which are split by two colors and images. Color combination signifies if the latest fingerprint is from current device (green) or from another one (blue). This is based on fingerprint origin device, which is always mobile, meaning if wear displays as green then fingerprint originated on wear connected to the current mobile phone. Image combination distinguishes between device types, mobile and wear in this case.

Each spot on the map can be clicked, including the markers, to display overlay with fingerprint actions. There are two different displays for this overlay. First, when user clicks on existing marker there are options to display information about that specific spot, containing list of the fingerprints, enable user to create new fingerprint on that spot, delete last group

Figure 5.4: Map screen with markers

or hide this window. Second, displayed when clicked spot does not have previously created fingerprints. This overlay displays only option to add new fingerprint or hide the window. Both of these options are displayed in Figure 5.4.

Adding new fingerprints will create all necessary data for the scan and run it via already mentioned JobScheduler. This scan has always priority, meaning if there is a non-priority scan running it is canceled and this one is run instead. It can happen when devices screen initiated a scan which did not finish before scheduling a scan using this screen. When scan is running all application screens (Activities) display scan information as shown in the Figure 5.5, thus informing about scan status in all parts of the application.

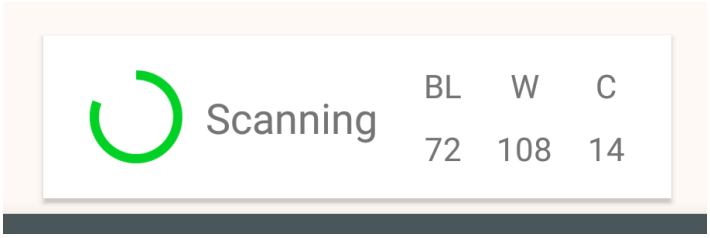

Figure 5.5: Scan status

This information window overlays the screen just above the application menu. It informs user about scan status, progress and how many device records were saved until now. Informations displayed are BLE (B), WiFi (W) and Cellular (C) data collected but it does not show Sensor (S) information to save screen space.

Original map image has 3000x2200 pixels but with a low data size around 230 KB, so it could be displayed as a single image but for already mentioned reasons it is not. The image is cut into four zoom levels (1 to 4) with first having 4 images and the final one with 144. Left side of Figure 5.6 shows maximum zoom on the mobile screen and right side displays information about fingerprints in a specific spot.

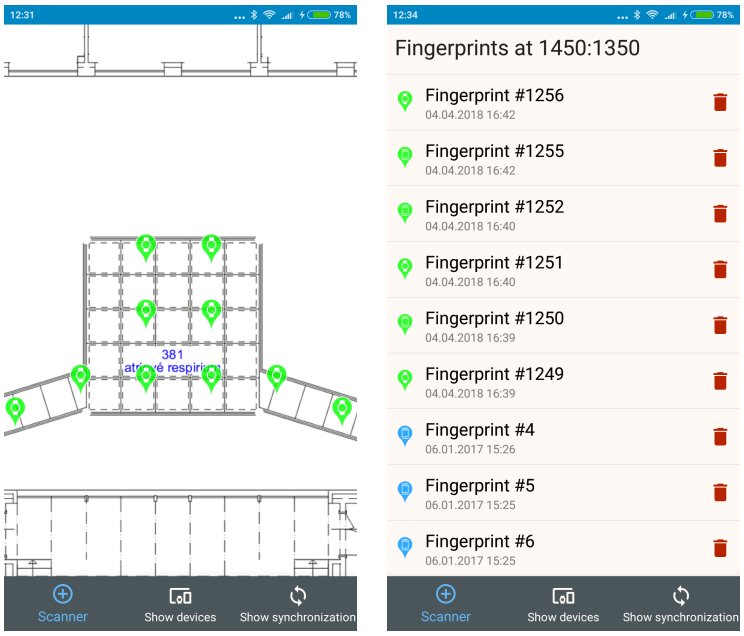

Figure 5.6: Maximum map zoom (left), List of fingerprints (right)

List of fingerprints for a specific spot displays information about origin device by icon, time it was created and enables to delete specific ones from the mobile. It is planned to extend this feature to display all important information about fingerprint, such as ids, device, counts of measurements and even RSSI averages. This would enable to assess fingerprint viability before uploading it on the server.

Important thing to note is that deleting fingerprints will not be reported to the server, therefore deletion of fingerprints is advised before uploading to the server. Deleting of already uploaded fingerprint on the phone will remove it only from that device and to delete it from the server is only enabled on the server itself.

5.4.4.2 Devices

This screen was supposed to be used for establishing Bluetooth connection between mobile and wearable device which is now done via WearOS application. Connection over Bluetooth is also not necessarily required since Data Layer API can now also send data between devices via WiFi. Even though it has no real use it was kept to display surrounding devices and

mainly BLE beacons.

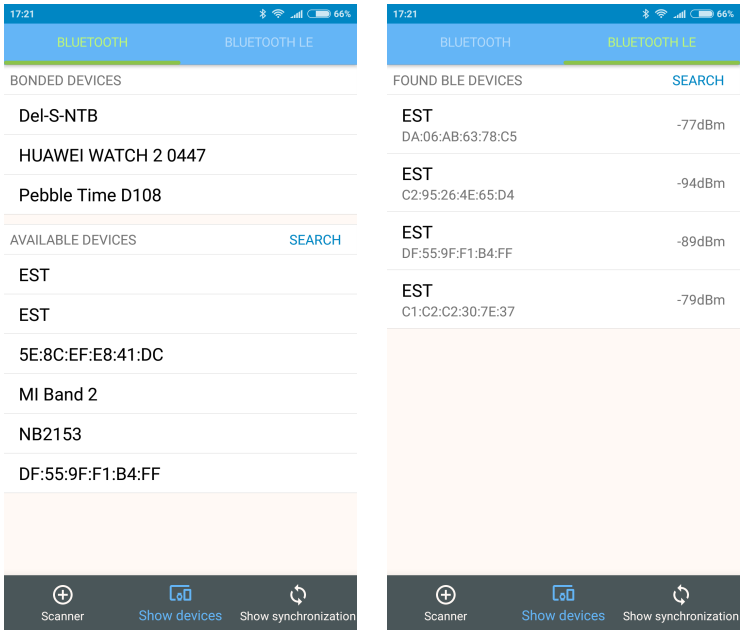

Figure 5.7: Bluetooth devices (left), Beacons (right)

Left screen of Figure 5.6 displays all the Bluetooth devices recorded during last scan, which is 15 seconds long. This scanning feature is not continuous and to display new devices it is needed to start a new scan by pressing "Search" button, which will also clear the available devices list. There is no need to scan for bonded Bluetooth devices since Android keeps them in a separate list, which is good but it does not check if the devices are in range or not, this means it can also display the device that is not available.

Right screen displays only devices broadcasting using Bluetooth Low Energy, those are mainly beacons but it can also be wristbands, some TVs and few more devices. It displays basic information like name, mac address and last signal strength received. Same as Bluetooth scanning it is not continuous and initiated by pressing "Search" button where length of such scan is set to 30 seconds in this case.

There is a possibility, for both screens, of device not having the name set and if that is the case, mac address is displayed instead of the missing name. To make this screen more viable it could display more information about devices, plus it could be extended to show WiFi access-points and Cellular towers in range.

5.4.4.3 Synchronization

As the name suggests this screen is used to synchronize data between mobile and the server. It shows current fingerprint differences and enables data synchronization based on

used input. It does not download or upload the data on its own since it is very data consuming, that is why it needs to be triggered by the user. Contrary to scanning, it does not display information using an overlay with the status but it updates counts on the screen after every successful upload or download call is done. It also shows an animation of synchronization button on top of the screen and displays a message if synchronization finished successfully or failed to inform user about the status of this task.

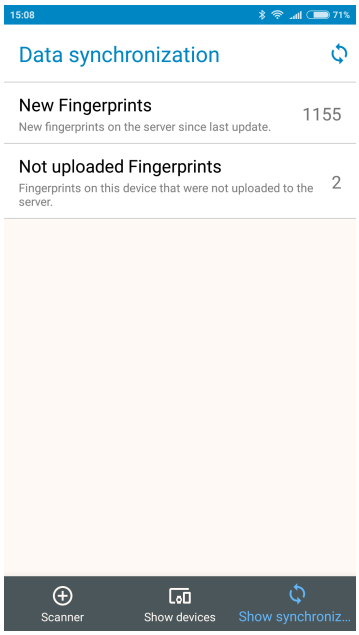

Figure 5.8: Synchronization screen

It is planned to expand this screen for the use of configuration to make application more flexible and able to be used in different environments than the current one. There are many changes considered which could influence the scanning results and application use.

- Be able to change scanning properties like maximum length, time periods for each scan or make scanning periodic. There could also be a separate settings for wearable device.
- Enable to change server API address and download/upload limits. The API would have to implement same endpoints as current one in this case.
- Implement map tiles uploading for different floors or buildings with the possibility to change between them and download fingerprints based on such selected location.
- User verification for the application and API with the possibility to change them on this screen.

5.4.4.4 Wear scanner

Wearable application is meant to be as simple as possible. It has only single screen since all the other settings are done on mobile application. This screen has two display states and holds the device wake lock preventing screen dimming or turning off when there is a scan running. Only reason to keep this wake lock is to display scan information without the need to click on wear screen. This feature could be also implemented as optional with the possibility to change in the phone application settings (currently synchronization) screen.

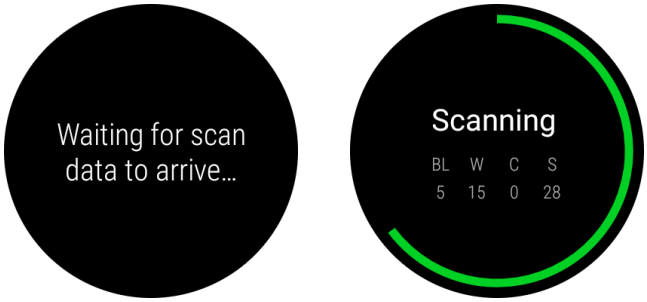

Figure 5.9: Wear scanning screens

First screen shows information message about receiving fingerprint data from mobile, when no data is received for 10 seconds it is closed to reduce battery drain. Second screen displays scan status with the progress bar and count of scanned records. When scan completes this informations stays displayed for tree seconds, after this time the screen is reset to the first one and application is closed. Since the scan can be run with wear at a position where user is not able to check the data, it informs about scan start and end with a short device vibration.

5.4.5 Interesting code examples

This application is a collection of complex code and functions where some interesting parts were selected and described in more detail.

5.4.5.1 Sending Fingerprint to Wear

Core functionality of communication between both devices. This code can be found in a WearDataSender class of mobile application and it sends fingerprint data to the wear using Data Layer API right after wear application is started.

```
if(mFingerprint != null) {
   mFingerprint.setDeviceEntry(null);
```

```
    PutDataMapRequest dataMapRequest =
        PutDataMapRequest.create(DataLayerListenerService.SCAN_PATH);
    DataMap dataMap = dataMapRequest.getDataMap();
    ParcelableUtils.putParcelable(dataMap, DataLayerListenerService.SCAN_DATA, mFingerprint);

    PutDataRequest request = dataMapRequest.asPutDataRequest();
    request.setUrgent();
    Wearable.getDataClient(mContext).putDataItem(request);

    mFingerprint = null;
}
```

Code example 5.8: Sending Fingerprint to Wear

This class handles starting of wear application and also sending fingerprint data, that is why it has a global variable for fingerprint data. Fist line of the code checks if this variable is not empty so it could be send to wear device. Following part of the code does only one change to the fingerprint before it is send, it removes origin device information to be filled in by the wear. When fingerprint is prepared it must be converted into Parcelable DataMap which is an object that is shared via Data Layer. This data is identified by a string key so application can save multiple objects into this map. Next part of the code creates a request using this data map and sets its mode to urgent to ensure fast delivery to all devices that listen for it. Request is completed at this point and can be send via DataClient which is loaded from the class Wearable. Final line of code takes care of clearing the global variable containing fingerprint to prevent sending the same data multiple times.

Data Layer identifies all the data by a specific Uri and when the data is changed it sends an information to all devices, meaning each fingerprint must be different for sending notification about the data change to other devices. Luckily, every fingerprint has its own specific id and scan id which is always different so this feature does not pose any problem to the data sending.

5.4.5.2 Parsing beacon information

None of the scanner parts use default classes to contain scanned information and each of them must be parsed into custom classes of this application. Following code, taken from FingerprintScanner class, shows an example how to parse beacon data.

```
String bssid = beacon.getBluetoothAddress();

long scanTime = currentMillis - mStartTime;
if(scanTime == currentMillis) {
    scanTime = 0;
```

```
}
long scanDifference = calculateBeaconScanDifference(scanTime, bssid);

BeaconEntry newBeacon = new BeaconEntry();
newBeacon.setBssid(bssid);
newBeacon.setDistance((float) beacon.getDistance());
newBeacon.setRssi(beacon.getRssi());
newBeacon.setTimestamp(currentMillis);
newBeacon.setScanTime(scanTime);
newBeacon.setScanDifference(scanDifference);

return newBeacon;
```

Code example 5.9: Parsing beacon information

First loaded information is device mac address because it is used to calculate time differences between last time this device was recorded and current time which is done in following part. After all time values are calculated BeaconEntry can be created containing all required data and immediately returned. Setting `scanTime` to 0 works as a protection against having it the same as `currentMillis` because start time of the scan might not have been set just yet. This is needed mainly for WiFi scanning because these parsing classes have to be set just before running the scanner code which leaves few milliseconds where data can be recorded before the scan is actually started, considering WiFi scan cannot be controlled. It could be also solved by not recording this data but since there is only a few millisecond difference it was decided that environment will not change enough to make the data irrelevant and so they are recorded too.

5.4.5.3 Map configuration

Map is the main part of mobile application and it handles multiple functions thus requiring a lot of settings to work as intended. It also supports many functions which can be configured or disabled based on the implementation required. This map and the code is implemented in MapFragment class and displayed using MapActivity.

```
mMap.setSize(MAP_WIDTH, MAP_HEIGHT);
mMap.addDetailLevel(1.000f, "tiles/j3np/1000/j3np-%d_%d.png");
mMap.addDetailLevel(0.500f, "tiles/j3np/500/j3np-%d_%d.png");
mMap.addDetailLevel(0.250f, "tiles/j3np/250/j3np-%d_%d.png");
mMap.addDetailLevel(0.125f, "tiles/j3np/125/j3np-%d_%d.png");

mMap.setScaleLimits(0, 2);
mMap.setScale(0.50F);
```

```
... Setup markers and hotspots ...

frameTo(MAP_WIDTH / 2, MAP_HEIGHT / 2);

mMap.setShouldRenderWhilePanning(true);
mMap.setShouldLoopScale(false);
```

Code example 5.10: Map configuration

First part contains setting map size which is 3000x3000 pixels and configure zoom levels with their specific tile images based on given path. Setting zoom levels one by one makes it highly modifiable with the possibility to have different designs for each zoom level. Next part sets zoom values, such as allowed zoom levels and initial zoom of the map. Following part is setting up markers and hotspots which will be described later. Calling function `frameTo()` centers the map with a delay because if it is done immediately it does not work. Final part of this code enables loading when map is panning and disables returning to maximum zoom level after double clicking while being zoomed in, on zoom level 2 in this case.

```
mMap.setMarkerTapListener(mMarkerTapListener);
mMap.defineBounds(0, 0, MAP_WIDTH, MAP_HEIGHT);
mMap.setMarkerAnchorPoints(-0.5f, -0.5f);

HotSpot hotSpot = new HotSpot();
hotSpot.set(0, 0, MAP_WIDTH, MAP_HEIGHT);
mMap.addHotSpot(hotSpot);
mMap.setHotSpotTapListener(mHotspotTapListener);
```

Code example 5.11: Setup markers and hotspots

This code example sets variables for map markers, such as displaying control window after clicking, define bounds and anchor point for the marker to center it horizontally and vertically. Defining bounds to the markers is very important information because it enables to calculate real pixel location from map's absolute position since this location changes based on how much the map is zoomed in. Scanner needs the real marker location to create fingerprint on the right spot and enabling the localization evaluation. This code does not actually display any markers on the map, that is done later in the code.

Second part of this example works similarly to previous one but it does not set the click function only to markers but creates an invisible overlay over the whole map, effectively enabling to click on any part of the map to display control window for fingerprints.

6 Testing and data analysis

Goal of this chapter is to describe application testing, data collection and evaluation of collected data.

6.1 Testing environment

Test site of this application is a third floor of the Campus building of the Faculty of Informatics and Management, University of Hradec Kralove (FIM UHK). The main walk-through corridors are in a rectangular arrangement. Classrooms and offices are situated inwards and outwards in relation to the corridors. There is a roofed atrium in the center of the building. Experiments have been conducted in a 52x43 meter area.

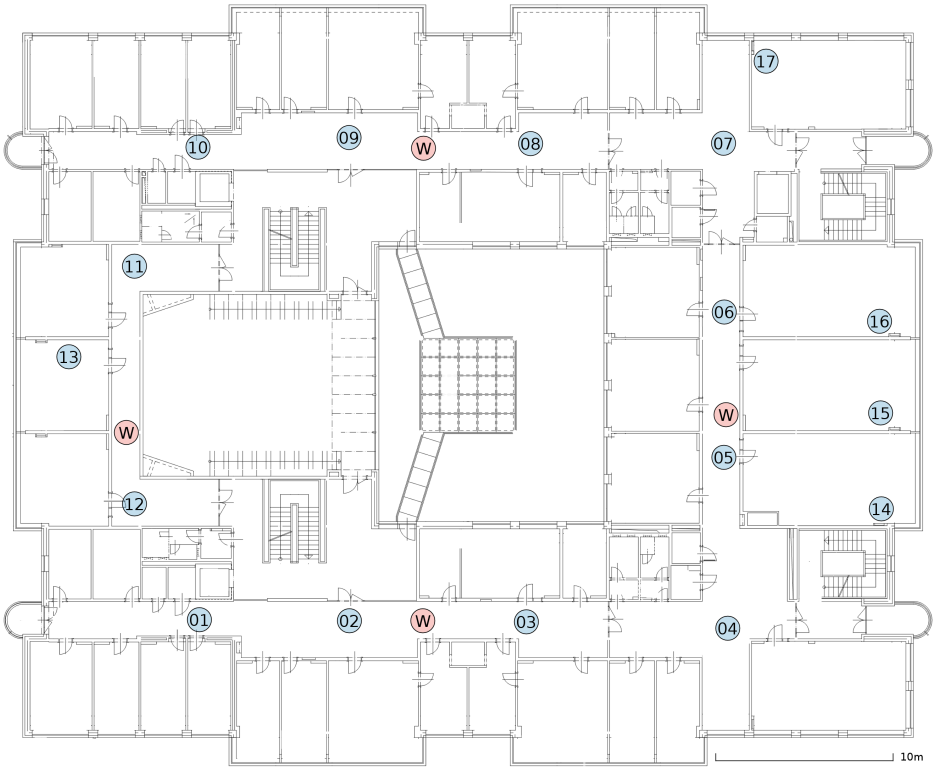

Figure 6.1: Map of deployed devices (based on [9])

Figure 6.1 shows environment and positions of deployed devices. There are four WiFi transmitters for **eduroam** network made by Cisco (marked as W) on this floor. They are permanently placed on the ceiling and their settings could not be altered. Each marked place has usually more than one of them, typically at least two to enable broadcasting in 2.4 GHz and 5 GHz band. Their TX and power is automatically adjusted to help mitigate interference and signal coverage problems.

The other devices marked by numbers from 1 to 17 are Bluetooth Low Energy beacons placed evenly in the corridors and classrooms on the floor. Their broadcast parameters were set to advertising interval of 100 ms and the TX power of 0 dBm. Corridor beacons are placed in positions about 10 meters apart from others [9]. Since the beacons were placed two years ago there was a possibility of battery depletion and malfunction. After multiple tests it was confirmed that two beacons had their battery depleted (1, 4), one was reconfigured (14) and one was misplaced and had to be changed (5).

6.2 Evaluation approach

Basic approach for evaluation is the same as in previous years, using solution created by Pavel Kriz in [9]. This approach is based on WKNN algorithm used to compare fingerprints against each other by selecting one and calculating its position using other fingerprints. Basic premise of this evaluation is to pick one fingerprint, forget its position and try to calculate it via specific amount of closest neighbors. This calculated value is then compared to the real one and difference between these values is the localization error in meters. Advantage of this approach is that it can be done in only one phase instead of two, since online phase is supplemented by one selected fingerprint. Second advantage is an easy implementation and calculation of error difference because both calculated and real positions are known.

This evaluation is implemented to be used only for one fingerprint and compare it to the others. It evaluates BLE and WiFi signals separate or together so it can provide an insight on which technology is better and if combining them improves the precision. This evaluation program also enables filtering out data not used for analysis, which are records of devices other than mentioned BLE beacons and WiFi access-points. It is developed to not differentiate between technologies of recording devices and it had to be expanded to support this feature. Few following implementations were added to this evaluation approach.

- Filtering data by the device type, which is an important part to enable testing only for a specific device type. These fingerprints are tested only against its own technology

and results can be used to compare precisions between different technologies.

- Selecting fingerprint group, it selects multiple fingerprints to be tested at the same time. This selection is done using scan id information, meaning both scan originated at the same time and place on different devices. Both fingerprints are then tested individually but the results are combined together with the aim to improve overall localization accuracy. This approach has two testing types, first is running the test against fingerprint's own technology and the other against all of them.
- Combining data from multiple fingerprints based on device technologies and consider it as one. Fingerprints are selected based on their scan id and all scanned data are cloned into mobile fingerprint combining them together. It can copy data only from selected radio technologies or all of them to provide the best overall localization.

All of these different approaches will enable to compare technologies and to assess which approach has the lowest localization error so it could be used in the future.

6.2.1 WKNN

This approach is an extension of algorithm **K-Nearest Neighbors** (KNN) with **Weights** making it **WKNN**. KNN is based on the idea that unclassified individual should belong to the same class as its nearest neighbor in the data set [93]. It calculates the position based on the locations of selected K-nearest neighbors, where K change influences the accuracy of this algorithm. Extended with weights, WKNN can make some neighbors or data more valuable than other ones. In this case, closer fingerprints have higher weight and will be used to calculate position by Euclidean distance formula. The equation for fingerprint $f = (f_1, f_2, ..., f_n)$ from the ith fingerprint $S_i = (s_{i1}, s_{i2}, ..., s_{in})$ can be expressed by following formula [9, 93]:

$$D_i = \sqrt{\sum_{j=1}^{N} (f_j - f_{ij})^2},$$

where N is a number of unique transmitters in the measurement.

These distance values are sorted and based on them, first k fingerprints are chosen to calculate weighted estimate of position P of measured fingerprint. It can be achieved by using their known positions $P_i[x_i, y_i]$ in a following formula:

$$P = \frac{\sum_{i=1}^{k} P_i Q_i}{\sum_{i=1}^{k} Q_i}, \; where \; Q_i = \frac{1}{D_i}.$$

WKNN algorithm was chosen because it is easy to implement and result data are not difficult to interpret.

6.3 Data collection

Data was collected at 105 evenly-spaced (2m) positions of single floor of Campus building. Each position was measured four times, in two different headings (north, south) and with two device orientations to prevent signal obstruction. Using both devices, 105 positions and 4 scans per position resulted in 840 different measurements, consisting of 466 411 BLE and 73 757 WiFi individual RSSI samples making it 540 168 in total.

Each of the scans was run for 30 seconds on both devices at the same time to provide equal environment. It is not really feasible to run such a long scan in one run, so the decision was made to split it into sub-scans with smaller time intervals. BLE beacons have advertising interval set to 100 ms and scanner length could be set to this number but it is set to 200 ms to prevent packet loss between the scans. Localization using WiFi usually does not need a big amount of data to provide good precision, thus scanner is set to run every 5 seconds. One problem with this scanner is that Android does not allow to set scan length or cancel running WiFi scan, only thing that can be canceled is listening for new WiFi records but scan could still be running in the background. Due to this limitation it can happen that current scan receives data from previous one, this is not prevented in the application but such data can be filtered out during evaluation by their scan time, which is usually set to 0. Cellular and sensor scanner are set to the same length as WiFi since this data is used only as complimentary information and it also helps to reduce fingerprint data size.

Three data collections were run during application development and data from the last one were used for evaluation. To consider fingerprint viable it must contain at least 20 records for Bluetooth Low Energy and WiFi, otherwise it needs to be deleted and re-scanned.

6.3.1 First data collection

This first collection was meant as a test run in real environment and it consisted of scanning ten spots, each was scanned three times for 20 seconds to check if the scanner is reliable. It revealed two main problems with the current scanner implementation.

First, WiFi scanner on wear device was not able to record any data. It was highly likely that there was a problem with the scanner and also scan length, since quick 30 seconds test showed data added at the last second. This was fixed by two main changes, increasing scan

length to 30 seconds and more importantly wear device now starts a "dummy" WiFi scan right after the application is displayed, just before an actual scanning is triggered. These changes helped to improve this scanner and it does not create any problems since there is little time span between this "dummy" scan and the actual scan, usual time difference is in milliseconds. This issue was also recorded only when wear was not connected to any WiFi network.

Second, not enough data was received by BLE scanner, usually less than by WiFi scanning. Data from previous years showed that each scan usually has hundreds of BLE records but this scanner was able to only collect around 50 of them. It was later discovered that sub-scan times were not properly set for AltBeacon library and scanner was run for 1 second instead of 200 milliseconds with only one device record per scan.

After both of these issues were fixed, second data collection was done to collect the data for evaluation. Final thing figured out during this collection is that wear device was not able to record any Cellular information even though specification shows it should support LTE technology, but that was probably because there were no LTE towers in range.

6.3.2 Second data collection

This collection was meant to be the final one and the data were supposed to be used for evaluation, but there were few more problems revealed after all data was collected. Data seemed fine at the first look but evaluation showed problems in BLE precision compared to the results from previous years. For a start, data analysis resulted with around 30 fingerprints removed from the testing due to no floor BLE beacons found and most of these fingerprints originated on wear device.

Max error (m)	BLE	WiFi	Combined
All devices	27	14	11
Mobile	30	14	11
Wear	31	11	11

Table 6.1: Maximum errors for second data collection

Table 6.1 shows a summary of maximum errors in meters for collected data, it shows that BLE errors can be two or three times higher than those of WiFi, reaching over 30 meters error which would place the device on the other side of the building. Mean and median

errors showed the same trend, so next step was to test if some beacons are missing in the data and display all fingerprint errors on the map to find the worst locations.

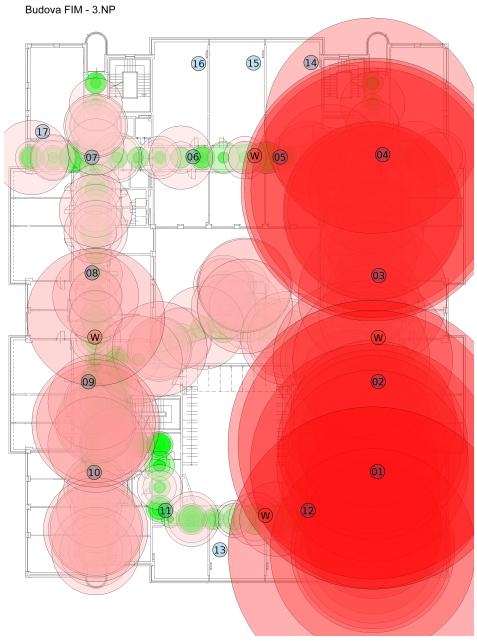

Figure 6.2: Map of errors for BLE in meters for all fingerprints

Figure 6.2 shows errors for BLE scanning on the map with two indications of error value. First, circle size is calculated from error number, higher errors are displayed using bigger circles. Second, fingerprints with error under three meters have green color and those above have color based on their error in the red spectrum, higher error results in more saturated red. Judging based on these information, evaluation seems to be the worst near beacons 1 and 4, which were later found out to be out of power. Next evaluation using differences in advertising interval revealed beacon number 14 reconfigured to different settings. And final tests, by visual checks if all beacons are in place, confirmed that beacon number 5 was moved and had to be replaced. All of these tests resulted in four malfunctioning beacons all over the floor and deeming the data not viable for further evaluation, hence a new round of data collection was required.

Since this was a complete data collection it also served as a test for wear device and how long it can stay operational while scanning. Multiple data was collected to measure this, such as start and end time, number of places and fingerprints scanned and of course power status. Wear device is never used under 15% because this is the threshold when power saving is turned on and device shuts down all non-essential functions.

Table 6.2 shows a summary of all important information for wear scanning evaluation. First, comparing the percentages of power with time shows that wear device cannot usually

	% Battery power			
Time	Start	End	Places	Fingerprints
2:50	100%	15%	30	240
2:40	100%	17%	37	296
1:55	100%	35%	12	96
1:20	100%	70%	18	144
0:45	30%	15%	8	64

Table 6.2: Scanning information for wear (second scan)

last for more than three hours of scanning, which is no enough, since all the scans together took about 8 hours to complete. The two longest scans show that wear can scan for around 35 places without the need to re-charge, which is confirmed by the shortest scan where scanning 8 places takes around 45 minutes and 15% of the device battery. There are also some exceptions, for instance, the third scan which was done in the middle of Campus building where there are no beacons and it was needed to re-scan multiple places many times to achieve set requirements for fingerprints mentioned in previous chapter.

6.3.3 Third data collection

Third and final data collection was conducted after all beacons were checked and fixed. Scanning showed only one single problem and that was again with the WiFi scanning on the wear, where no records were found after around 10 spots (80 fingerprints) collected. This was not a huge issue because it has two easy fixes. First, restarting the device which will enable to scan for 10 more spots without problems. Second, turn off and on the WiFi receiver on the wear device but this fix only works for next two or three scans, so it is not ideal. Both of these solutions were used based on the situation but both devices were always turned on and off after 15 successfully scanned spots.

For this scan, only one specific change was made and that was starting a "dummy" scan on a mobile device. Before this collection it was implemented only on wear device, which was changed to ensure that scanning code for both devices is completely the same and thus provide equal environment. Collecting of all fingerprints took about 7 hours and 15 minutes, in this case, which is almost an hour less than the length of previous scanning which took 8 hours and 10 minutes. Battery information in Table 6.3 confirms the results from previous

	% Battery power			
Time	Start	End	Places	Fingerprints
2:40	100%	15%	37	296
2:00	100%	33%	30	240
1:10	100%	58%	18	144
0:45	50%	22%	11	88
0:40	40%	15%	9	72

Table 6.3: Scanning information for wear (third scan)

scanning and shows that wear device cannot scan more than three hours at the time before reaching power saving mode threshold. This feature can now be disabled after the new update of wear device system but it was not used since there is no guarantee it will disable all parts of power saving and provide equal environment.

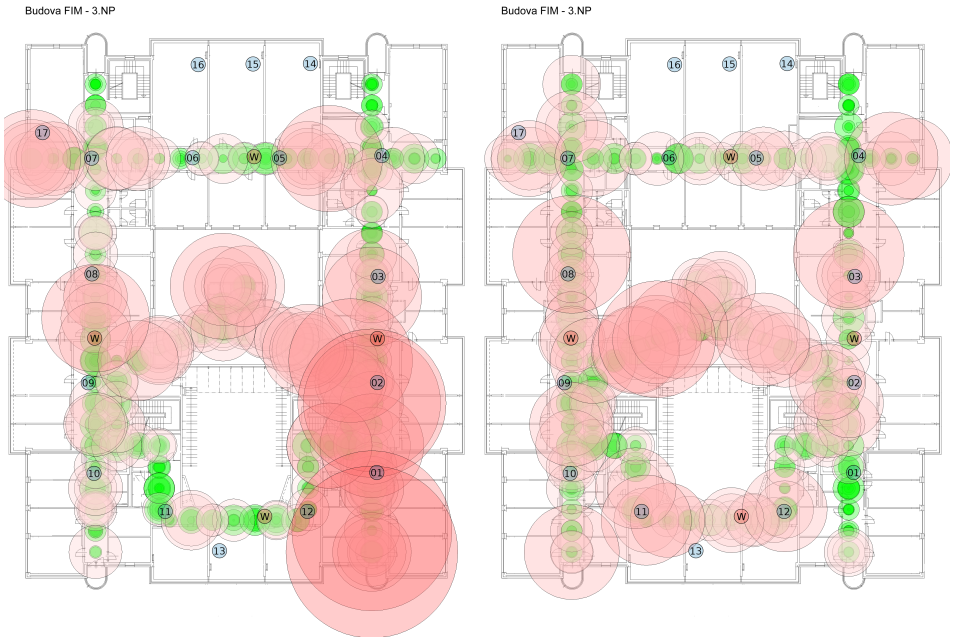

Figure 6.3: Map of errors for BLE (left) and WiFi (right)

Figure 6.3 shows two images, one per radio technology, of all fingerprint errors on the map, creating eight overlapping circles on each spot. WiFi seems more precise in comparison with BLE and there might still be some issues around beacon 1. Places with one or two light red circles will usually be improved by other fingerprints on that spot but the others could increase overall localization error. On the other hand, comparison of this figure and Figure 6.2 shows a decrease of BLE errors for this scanning. Looking at the differences between WiFi

and BLE suggests that WiFi should have less errors and better accuracy but there might be issues for both technologies in the middle of the building.

6.4 Evaluation

Evaluating of all the data has multiple approaches and combinations since in needs to consider information, such as algorithm implemented, multiple device types and radio signal technologies. During the second data collection it was figured out that wear device is not able to scan for Cellular records so these radio signals will not be tested in the evaluation. First thing that needs to be decided is which k value to use in following evaluations using WKNN algorithm and after that multiple tests can be run to compare device types and how fingerprints can be used to improve each other.

6.4.1 Decide K values for WKNN

To select proper k value all data was tested without their device information, thus putting all of them together and test them against each other. Only $k \in \{2, 3, 4\}$ were used in the selection process because using $k = 1$ defeats the purpose of WKNN algorithm and setting $k > 4$ results with too big errors. Following table shows a summary of mean, median and maximum errors in meters for all tested k values.

	K = 2			K = 3			K = 4		
	BLE	WiFi	Combined	BLE	WiFi	Combined	BLE	WiFi	Combined
Mean	2.25	1.77	1.52	2.20	1.76	1.52	2.15	1.80	1.56
Median	1.93	1.06	1.02	1.76	1.30	1.22	1.61	1.32	1.06
Maximum	12.90	11.11	10	12.59	11.38	9.67	13.28	10.60	9.39

Table 6.4: List of errors for multiple K values

Mean and median information are the main factors used to decide which k to use and maximum is meant only as a complimentary information. Considering values by technology it seems $k = 2$ is best for WiFi and $k = 4$ for BLE so it will be decided between these two. Combining technologies together shows better mean and median for $k = 2$ and maximum for $k = 4$ and because main values are better when using $k = 2$ it was selected for following evaluations. It also achieves better accuracy with WiFi, which is more precise than BLE technology, and this value was also already tested and used in previous years [9].

Figure 6.4 shows comparison for both radio technologies used for evaluation and their combinations. It correlates with Table 6.4 showing WiFi as a better technology and their

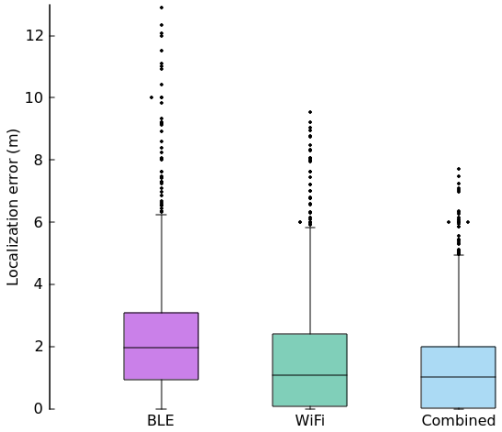

Figure 6.4: Comparison of localization accuracy for $k = 2$

combination improved over using only single one of these two technologies. This graph is also used to compare this evaluation method with future iteration of WKNN algorithm which consider technology of recording device when evaluating fingerprints.

6.4.2 Compare device technologies

After it is known which parameters to use, evaluation of the device technologies can be started. For this part minimum of data filtering was used and all the fingerprints were tested based on their device technology. To compare overall precision all data were also put together and tested against each other without taking device technology into consideration, meaning position of mobile fingerprints could be calculated using wear fingerprints and vice versa. This is not the best approach since it defeats the purpose of using different device types but it is easily implemented for the first round of evaluation.

	Mean error (m)			Median error (m)			Max error (m)		
	BLE	WiFi	Combined	BLE	WiFi	Combined	BLE	WiFi	Combined
Wear	2.27	2.67	2.21	2.00	2.10	2.00	12.90	11.11	10
Mobile	2.05	0.88	0.85	1.76	0.80	0.88	11.50	8	6.06
All devices	2.25	1.77	1.52	1.93	1.06	1.02	12.90	11.11	10

Table 6.5: Device comparison: mean and max errors (in meters)

Table 6.5 shows the comparison of device fingerprints and their calculated mean and max error values. BLE technology does not show much of a difference in devices, just about 0.2 meters which makes them comparable. The WiFi, on the other hand, shows a big difference between them and making mobile device a better solution with over two times lower mean error. Fingerprint data were checked to find precise reason behind this difference.

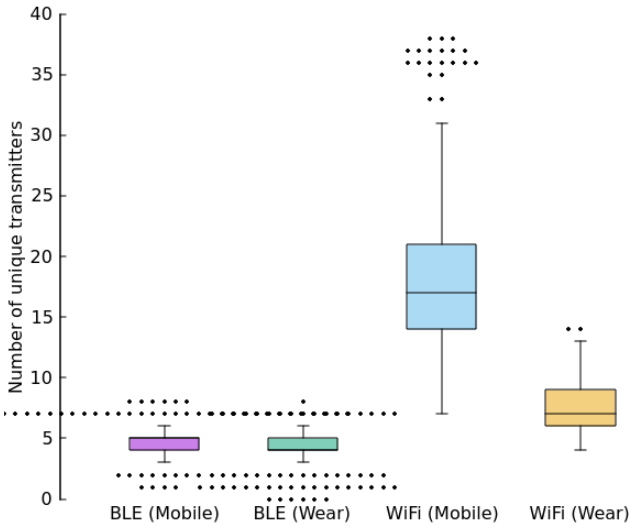

Figure 6.5: Number of transmitters for all fingerprints

Figure 6.5 shows the number of transmitters for all fingerprints based on radio technology and device. It shows close numbers for BLE but there seems to be a lower amount of WiFi transmitters recorded by wear device. It seemed that wear was not able to record signals from some of them and after comparing the data from wear to mobile it was discovered that missing transmitters use 5 GHz technology. Mobile fingerprints contain over 7 300 WiFi records originating on access-points using 5 GHz and wear has 0 of them recorded, which means that wear device does not contain 5 GHz WiFi receiver. Records from 5 GHz are more precise than those of 2.4 GHz making this a reason why wear precision is not as good when comparing it to the mobile device. Every manufacturer is in control of hardware for its devices and Huawei decided not to implement it due to high power usage.

Combining data from both wireless technologies together and comparing devices shows wear error almost three times higher than the one of mobile device, thus increasing the error when combining all data together. It increases mean error from mobile only 0.85 to 1.52 combined and max from 6.06 to 10. Using only mobile device would result in better precision instead of combining them in this case. It is connected to already mentioned WiFi precision of wear device.

To compare both technologies more precisely all fingerprint errors were sorted based on location and displayed in Figure 6.6. This image confirms the information that using wear device fingerprints will result in higher error than mobile but there are few places where using wear is better, unfortunately not a lot of them.

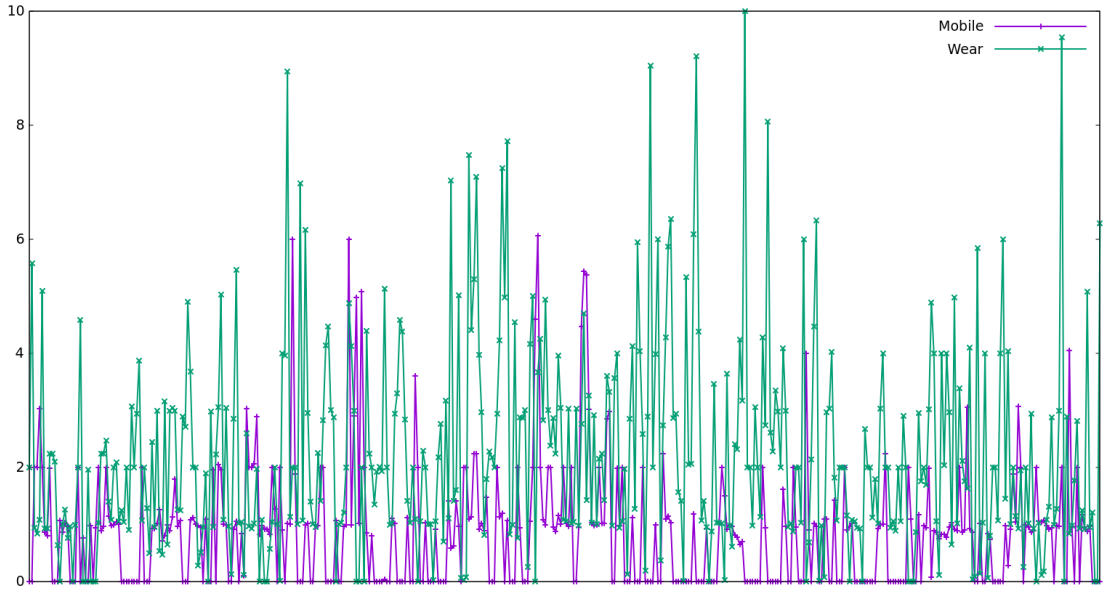

Figure 6.6: Comparison of errors based on device

6.4.3 Testing multiple fingerprints

Previous method is testing only single fingerprint against all other ones without the device type information. Improving upon previous design this evaluation selects multiple, two in this case, fingerprints with different origin device and runs the WKNN algorithm on them. These fingerprints have to be from the same location and also from the same scan group, which is identified by their scan id. All of these fingerprints have their location calculated and this information is then averaged to increase the location precision. This algorithm supports multiple technologies and it could be improved to support weights where specific technology, such as mobile, could have higher weight to reduce influence of technologies with worse precision.

There are two main implementations of this evaluation. First, calculates position for both fingerprints while ignoring their device technology, each fingerprint location is calculated using all fingerprints, same as previous evaluation. Second, each of the fingerprint location is calculated only using its own device technology which is used to check the calculation influence based on the device, thus figuring out if testing fingerprints against their own technology can improve overall precision or make it worse.

Comparing this Table 6.6 with Table 6.5 shows improvement in BLE precision but median values have gotten worse for WiFi making combined error higher from 1.02 to 1.33 meters compared to previous evaluation, which was expected. On the other hand there is a big decrease of maximum errors for both technologies decreasing combined error from 10 to 5.78 meters. From these values it is not easy to compare if precision got better or not but it can be

	Compared to own technology			Compared to all		
	BLE	WiFi	Combined	BLE	WiFi	Combined
Mean	2.18	1.77	1.53	2.30	1.77	1.52
Median	1.87	1.50	1.33	1.99	1.50	1.33
Maximum	10.41	6.21	5.78	10.59	6.21	5.78

Table 6.6: List of errors for testing multiple fingerprints

used to compare if there is any influence of comparing fingerprints with its own technology. It shows slightly better values for BLE but those do not influence combined results which are almost completely the same, meaning there is no reason to compare fingerprint only with its own technology in this approach.

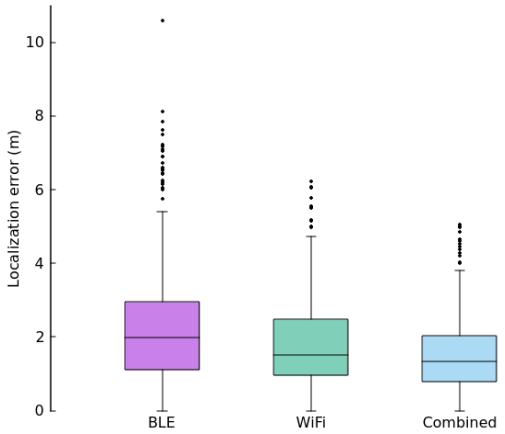

Figure 6.7: Comparison of localization accuracy for testing multiple fingerprints

When comparing this figure with Figure 6.4 it shows a decrease of median accuracy mainly for WiFi which is also reflected on combined evaluation. Overall accuracy improves because errors are more clustered together and more evenly spread, which is confirmed by the decrease of maximum error from 10 to 5.78 making it more viable in complex environments and problematic places. This approach is not better then mobile only localization with these errors.

6.4.4 Combining fingerprint data

Last evaluation did not prove tangible improvement of localization accuracy using wear technology in comparison with mobile only. For this reason next evaluation takes a different approach and combines data from multiple fingerprints, based on the device, into one. As

it was already mentioned, fingerprints are grouped based on scan id, which identifies them as being taken at the same place and time using different devices. Data of such fingerprints are combined into single one, reducing their count for evaluation but hopefully increasing overall accuracy. Since there are multiple fingerprints combined together, the information about origin device is lost and there is no way to compare the technologies against each other.

There are two possible cases which were implemented for this evaluation. First, combine all data from a fingerprint group into single fingerprint. Second, combine only BLE data and keep WiFi from mobile, because it is known that wear WiFi precision is not sufficient and decreases the localization accuracy.

	WiFi with BLE			BLE only		
	BLE	WiFi	Combined	BLE	WiFi	Combined
Mean	2.12	0.93	0.88	2.12	0.88	0.83
Median	1.73	0.87	0.86	1.73	0.80	0.83
Maximum	13.99	7.19	6.99	13.99	8.00	7.04

Table 6.7: List of errors for fingerprint combination

This approach as a whole seems to be a big overall improvement in accuracy but with the increase of maximum error for BLE technology as a drawback. Copying all data together not only improves the precision, compared to previous evaluations, but also makes it comparable to using only mobile. In numbers from Table 6.5, mobile errors are 0.85m mean, 0.88m median and 6.06m maximum compared to 0.88m mean, 0.86m median and 6.99m maximum for this evaluation. Improvement is only in median but other two values increase in comparison.

Second part of this approach, copying only BLE records to mobile fingerprints while keeping its current WiFi precision. With BLE error decreased this finally proved to be a better solution than using only mobile localization. Comparing the data from Table 6.7 shows decrease of error for mean value from 0.85 to 0.83 meters which is around 2.4% and median from 0.88 to 0.83 meters, reaching up to 6% improvement over mobile only localization. One value that goes up is maximum error from 6.06 to 7.04 resulting in 16% increase of this error.

Figure 6.8 shows visualization of errors for both parts of this approach. BLE errors are the same for both cases and they seem to be somewhere between classic approach and testing of multiple fingerprints, since it has more clustered values compared to classic one but not as much as multiple fingerprints testing approach. However, WiFi shows the best localization precision compared to all previous iterations. Both of these approaches seems to be compa-

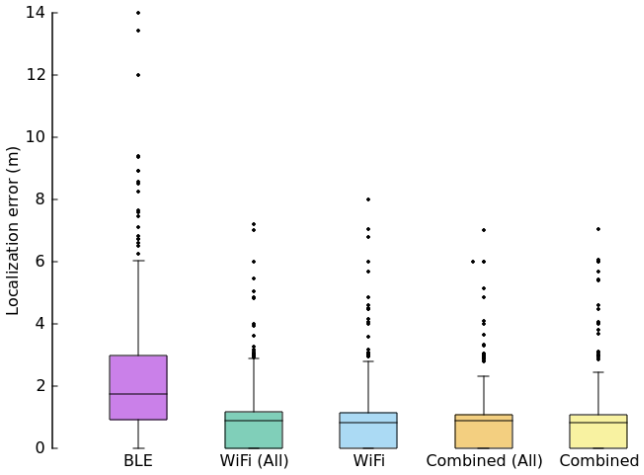

Figure 6.8: Comparison of localization accuracy for combining fingerprints

rable and each one has a different merit to it. Combination of all data seems to increase mean and median but lowers the maximum errors, making it more reliable in problematic places, where the other approach seems to have the opposite result and being better overall.

6.4.5 Map comparison

To confirm previous results, all errors were displayed on the map to compare them and also to find problematic spots of the evaluation. These maps contain only combined errors of radio technologies to make it easier to read and not complicate it with differences between BLE and WiFi. There is also one other slight change to displayed data where errors under 200 centimeters are set to display as a circle with this precise error to make them more visible. This only influences last two images of Figure 6.9 because there are multiple positions with 0 meters error fingerprints.

Figure 6.9 shows map images for all previously described and evaluated algorithms with last one displaying mobile only evaluation. Top left image shows errors for classic evaluation that ignores device of origin. Top right is for the next round of evaluation which takes multiple fingerprints with different device origins, calculates their position and averages it to improve accuracy. Bottom left picture is for the last evaluation which combines multiple fingerprints into one. Finally, picture on the bottom right displays errors for mobile only localization used as a comparison with previous algorithms.

Classic evaluation shows a big amount of orange circles, which means that error of a specific fingerprint is over three meters, and most of them seem to be in the middle of Campus building where there are no beacons placed associated with this floor. The other problematic spots seems to be in the south part of the building (bottom part of the map). Rest of the

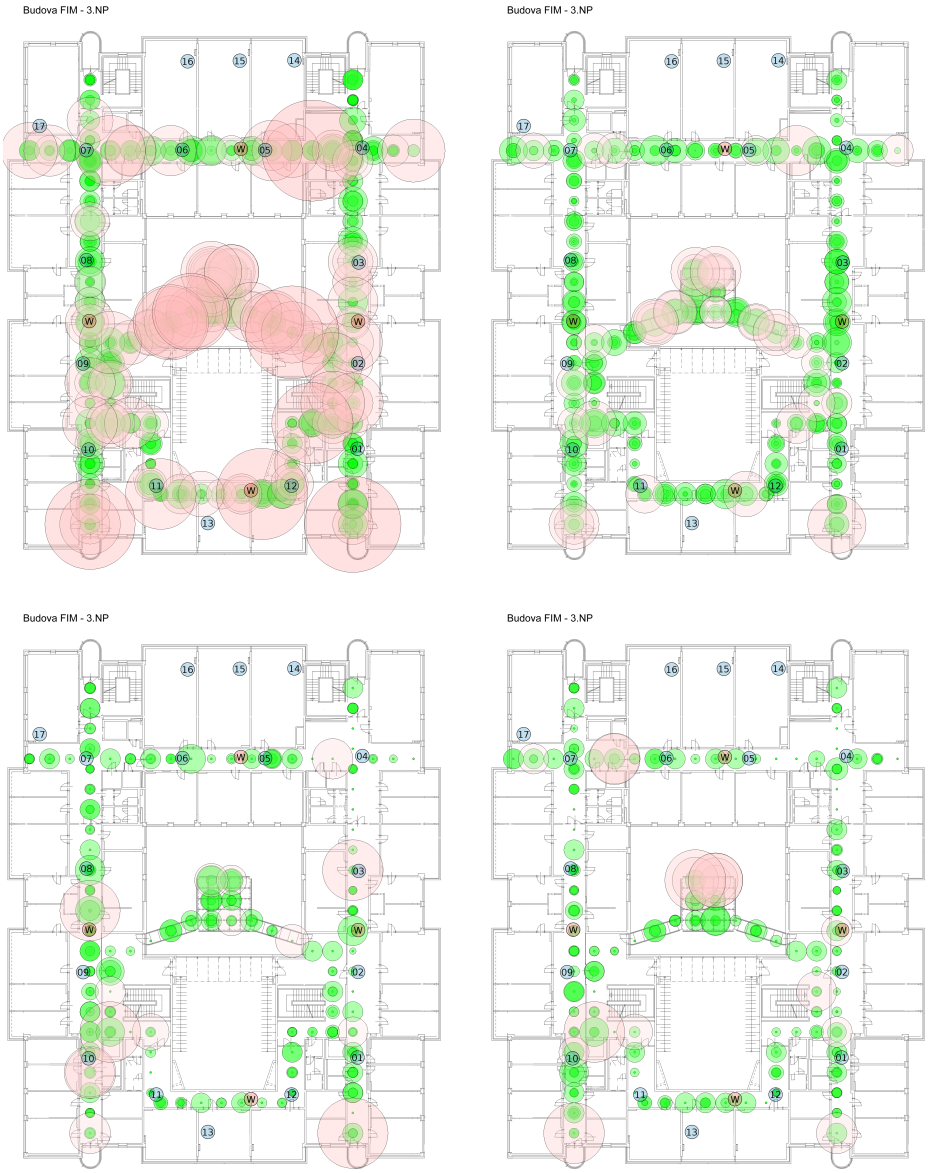

Figure 6.9: Maps of errors for all algorithms with last one for mobile error only

places are not considered as problematic since they usually have only one fingerprint with error over three meters. There are 119 fingerprints with error higher than three meters which is around 14%.

Moving to the second image, it shows a big improvement of the localization for all positions and filtering out which places may actually be problematic. Bringing the count of fingerprints with error higher than three meters from 119 to 35, which is just 4% of all fingerprints. And as for the final algorithm image (bottom left), it illustrates the best precision of all algorithms where 82 fingerprints have zero meters error and just 15 with error above three meters, bringing it down just under 2%.

7 Conclusion

This thesis has introduced a new way to collect radio fingerprints on mobile in combination with wear device to improve indoor stationary localization. This solution also included changes to the data distribution between devices and the server. Created system consists of a server, mobile and wear devices with the Android operating system and WearOS for wear, both of these Android devices with Bluetooth Low Energy support. This system is designed to enable creation of radio fingerprint maps and update them anytime. Evaluation of this system was based on the Weighted K-Nearest Neighbors algorithm. This evaluation works only with specifically defined beacons and WiFi access-points to maintain equal environment for all fingerprints. Based on the acquired data in a real world scenario, the results of the localization were evaluated using WiFi, BLE and their combination with addition to using data from mobile, wear and their combination.

Evaluation was composed of three main algorithm implementations to figure out how to use data from multiple device types to improve overall localization accuracy. First test, using data from all device types separately showed lower accuracy of WiFi localization on wear, this was traced back to wear device not possessing the ability to scan for 5 GHz WiFi signals. This actually makes overall localization less accurate when combining the data together, where mean error increased from 0.85 to 1.52 meters, making it almost two times worse. Second test used one fingerprint per device type and averaged their location resulting in improvement of overall accuracy compared to previous evaluation but it is still not as precise and using single mobile device. Third and final evaluation combined data from multiple fingerprints at the same position and created using different device types, data from wear are cloned into data from mobile fingerprints. Best results were reached when combining fingerprint data from both devices into a single one with up to 6% improvement of overall accuracy when combining only BLE records and keeping WiFi from mobile. Accuracy improvement could be even higher in locations which are not using 5 GHz WiFi transmitters or by using multiple devices supporting 5 GHz WiFi. When this network type is used and scanning devices do not support it, better localization accuracy can be reached by not including WiFi results from

these device in the evaluation.

It was proved that wear device can improve localization and its precision when used with the mobile but it is important to keep in mind that at this time smartwatches do not possess high battery life and cannot be used to scan for a long periods of time. Higher battery life could mean the possibility for manufacturers to implement 5 GHz WiFi receiver and effectively improve localization even more, bringing accuracy to the level of mobile device. Important thing to mention is that using smartwatches can sometimes be preferred since it is tightly connected to a person.

7.0.1 Future improvements

This application is meant to be used in different environments and the first improvement should be to enable uploading of new maps and differentiating between locations, make it possible to change between them and load different fingerprints based on a specific location. Another likely improvement is making scanning more viable by supporting changes in its settings, such as length of the scan or lengths of all sub-scans. Final and very important improvement could be implementation of location calculation and display, where scanned data would be send to the server which would return calculated location to the application. This part would be implemented at the end of development process or it could be implemented in a separate solution to prevent over-compilation of existing application.

Literature

[1] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger and Elmar Wasle. *GNSS – Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Springer Science & Business Media, 2007 [cited 2018-01-10], ISBN 9783211730171.

[2] AviationChief. *Global Navigation Satellite System (GNSS) Global Positioning Satellite (GPS) System* [online]. AviationChief.Com, 2017 [cited 2018-01-15]. Available at: `http://www.aviationchief.com/gps-system.html`

[3] Xinglin Piao, Yong Zhang, Tingshu Li, Yongli Hu, Hao Liu, Ke Zhang and Yun Ge. *RSS Fingerprint Based Indoor Localization Using Sparse Representation with Spatio-Temporal Constraint* [online]. National Center for Biotechnology Information, 2016 [cited 2018-01-14], Available at: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5134504/`

[4] Stéphane Beauregard and Harald Haas. *Pedestrian Dead Reckoning: Basis for Personal Positioning* [online]. School of Engineering and Science International University Bremen, 2006, Available at: `http://ave.dee.isep.ipp.pt/~lbf/PINSFUSION/BeHa06.pdf`

[5] Gabriel Deak, Kevin Curran and Joan Condell. *A survey of active and passive indoor localisation systems*. In: *Computer Communications*. Elsevier, 2012 [cited 2018-01-11], Volume 35, Issue 16, ISSN: 0140-3664.

[6] Zahid Farid, Rosdiadee Nordin, and Mahamod Ismail. *Recent Advances in Wireless Indoor Localization Techniques and System* [online]. School of Electrical, Electronics & System Engineering, University Kebangsaan Malaysia (UKM), 2013 [cited 2018-01-15], Available at: `http://downloads.hindawi.com/journals/jcnc/2013/185138.pdf`

[7] Shweta Singh, Ravi Shakya and Yaduvir Singh. *Localization techniques in wireless sensor networks* [online]. Department of Computer Science, Ideal

Institute of Technology, Ghaziabad, 2015 [cited 2018-01-15], ISSN: 0975-9646, Available at: https://pdfs.semanticscholar.org/6299/85defbf9cc1a937a1b88c9c2a893552e3d89.pdf

[8] Paweł Kułakowski, Javier Vales-Alonso, Esteban Egea-López, Wiesław Ludwin and Joan García-Haro. *Angle-of-arrival localization based on antenna arrays for wireless sensor networks* [online]. In: *Computers & Electrical Engineering*. Elsevier, 2010 [cited 2018-01-15], Volume 36, Issue 6, Pages 1181-1186. Available at: http://ai2-s2-pdfs.s3.amazonaws.com/17c6/0e17c4e72cc3fd821e12169c1c2ca7736bd4.pdf

[9] Pavel Kriz, Filip Maly, and Tomas Kozel. *Improving Indoor Localization Using Bluetooth Low Energy Beacons* [online]. In: *Mobile Information Systems*. Hindawi Publishing Corporation, 2016 [cited 2018-01-15], Volume 2016, Article ID 2083094. Available at: https://www.hindawi.com/journals/misy/2016/2083094/abs/

[10] GISGeography. *Trilateration vs Triangulation – How GPS Receivers Work* [online]. GISGeography.com, 2018 [cited 2018-01-15]. Available at: http://gisgeography.com/trilateration-triangulation-gps/

[11] Kenjirou Fujii, Yoshihiro Sakamoto, Wei Wang, Hiroaki Arie, Alexander Schmitz and Shigeki Sugano. *Hyperbolic Positioning with Antenna Arrays and Multi-Channel Pseudolite for Indoor Localization* [online]. MDPI AG, Basel, 2015 [cited 2018-01-15]. Available at: http://www.mdpi.com/1424-8220/15/10/25157/htm

[12] David Munoz, Frantz Bouchereau Lara, Cesar Vargas and Rogerio Enriquez-Caldera. *Position Location Techniques and Applications*. Elsevier Science Publishing Co Inc, 2009 [cited 2018-01-15], ISBN: 9780080921938. Available at: http://www.mdpi.com/1424-8220/15/10/25157/htm

[13] Group 891: Wireless Location. *ANGULATION: AOA (Angle Of Arrival)* [online]. DEPARTMENT OF ELECTRONIC SYSTEMS, Aalborg University, 2010 [cited 2018-01-15]. Available at: http://kom.aau.dk/group/10gr891/methods/Triangulation/Angulation/ANGULATION.pdf

[14] Jais, M. I., Ehkan, P., Ahmad, R. B., Ismail, I., Sabapathy, T., and Jusoh, M. *Review of angle of arrival (AOA) estimations through received signal strength indication (RSSI) for wireless sensors network (WSN)* [online]. In: Computer,

Communications, and Control Technology (I4CT), 2015 International Conference on. IEEE, 2015, [cited 2018-01-16], p. 354-359. Available at: https://www.researchgate.net/profile/Phaklen_Ehkan/publication/283476641_Review_of_angle_of_arrival_AOA_estimations_through_received_signal_strength_indication_RSSI_for_wireless_sensors_network_WSN/links/564106b008aebaaea1f6d6e5/Review-of-angle-of-arrival-AOA-estimations-through-received-signal-strength-indication-RSSI-for-wireless-sensors-network-WSN.pdf

[15] Quuppa Oy. *Quuppa Intelligent Locating System* [online]. 2018 [cited 2018-01-16]. Available at: http://quuppa.com/technology/

[16] Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N. Padmanabhan. *Indoor Localization Without the Pain* [online]. In: Proceedings of the sixteenth annual international conference on Mobile computing and networking, 2010 [cited 2018-01-16], Available at: http://dl.acm.org/citation.cfm?id=1860016

[17] Xiaoyang Wen, Wenyuan Tao, Chung-Ming Own, and Zhenjiang Pan. *On the Dynamic RSS Feedbacks of Indoor Fingerprinting Databases for Localization Reliability Improvement* [online]. Sensors, 2016 [cited 2018-01-16], Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5017443/

[18] Cisco. *Wi-Fi Location-Based Services 4.1 Design Guide - Location Tracking Approaches* [online]. Cisco, 2018 [cited 2018-01-16], Available at: https://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Mobility/WiFiLBS-DG/wifich2.html

[19] Jeffrey Hightower and Gaetano Borriello. *Location systems for ubiquitous computing* [online]. Computer, 2001 [cited 2018-01-17], 34.8: 57-66. Available at: http://www.csd.uoc.gr/~hy439/lectures11/hightower2001survey.pdf

[20] COOK, B., et al. *Location by scene analysis of wi-fi characteristics* [online]. Relation, 2009 [cited 2018-01-17], 10.1.119: 6216. Available at: http://www.ee.ucl.ac.uk/lcs/previous/LCS2006/2.pdf

[21] Levi, R.W. and Judd, T. *Dead reckoning navigational system using accelerometer to measure foot impacts* [online]. Google Patents, 1996 [cited 2018-01-17]. Available at: https://www.google.com/patents/US5583776

[22] Z. Zhou, T. Chen, L. Xu. *An Improved Dead Reckoning Algorithm for Indoor Positioning Based on Inertial Sensors* [online]. In: Advances in Engineering Research, 2015 [cited 2018-01-17]. ISBN: 978-94-62520-71-4. Available at: `https://www.atlantis-press.com/proceedings/eame-15/22314`

[23] NAKAJIMA, Naoki, et al. *Improving Precision of BLE-based Indoor Positioning by Using Multiple Wearable Devices* [online]. In: Adjunct Proceedings of the 13th International Conference on Mobile and Ubiquitous Systems: Computing Networking and Services. ACM, 2016 [cited 2018-03-26]. p. 118-123. Available at: `https://dl.acm.org/citation.cfm?id=3004041`

[24] WANG, Xiaoliang; XU, Ke; LI, Ziwei. *SmartFix: Indoor Locating Optimization Algorithm for Energy-Constrained Wearable Devices.* In: Wireless Communications and Mobile Computing, 2017 [cited 2018-03-26]. Available at: `https://dl.acm.org/citation.cfm?id=3004041`

[25] W. Xiaoliang, X. Ke, Y. Zheng, and Z. Ge. *Tinyloc: Indoor localization for energy-constrained wearable devices* [online]. In: Chinese Journal of Computers, 2016 (Chinese), [cited 2018-03-26]. Available at: `http://www.cnki.net/kcms/detail/11.1826.TP.20161106.1649.002.html`

[26] SUN, Wei, et al. *MoLoc: On distinguishing fingerprint twins* [online]. In: Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on. IEEE, 2013 [cited 2018-03-26]. p. 226-235. Available at: `http://ieeexplore.ieee.org/abstract/document/6681592/`

[27] HOELZL, Gerold, et al. *Size does matter-positioning on the wrist a comparative study: Smartwatch vs. smartphone* [online]. In: Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on. IEEE, 2017 [cited 2018-03-30]. p. 703-708.

[28] Marziah Karch. *What Is Google Android?* [online]. Lifewire, 2017 [cited 2018-01-17]. Available at: `https://www.lifewire.com/what-is-google-android-1616887`

[29] Android. *Android Open Source Code* [online]. Android.com, 2018 [cited 2018-01-17]. Available at: `https://source.android.com/`

[30] Android. *Android Developers* [online]. Android.com, 2018 [cited 2018-01-17]. Available at: https://developer.android.com/index.html

[31] Renju Liu and Felix Xiaozhu Lin. *Understanding the Characteristics of Android Wear OS* [online]. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 2016. p. 151-164. Available at: https://athena.smu.edu.sg/mobisys/backend/mobisys/assets/paper_list/pdf_version/paper_12.pdf

[32] Samuel Gibbs. *10 most influential wearable devices* [online]. Guardian News, 2017 [cited 2018-01-18]. Available at: https://www.theguardian.com/technology/2017/mar/03/10-most-influential-wearable-devices

[33] Gartner, Inc. *Gartner Says Worldwide Wearable Device Sales to Grow 17 Percent in 2017* [online]. Gartner, Inc., 2017 [cited 2018-01-18]. Available at: https://www.gartner.com/newsroom/id/3790965

[34] Bahman Rashidi and Carol Fung. *A Survey of Android Security Threats and Defenses* [online]. JoWUA, 2015, [cited 2018-01-19]. Available at: https://www.researchgate.net/profile/Bahman_Rashidi2/publication/282365848_A_Survey_of_Android_Security_Threats_and_Defenses/links/560ec06908ae6b29b499a51f/A-Survey-of-Android-Security-Threats-and-Defenses.pdf

[35] Parvez Faruki, Ammar Bharmal, Vijay Laxmi, Vijay Ganmoor, Manoj Singh Gaur and Mauro Conti. *Android Security: A Survey of Issues, Malware Penetration and Defenses* [online]. IEEE Communications Surveys and Tutorials, 17(2), pp. 998-1022, 2015, [cited 2018-01-19]. Available at: http://dx.doi.org/10.1109/COMST.2014.2386139

[36] Statista. *Number of available applications in the Google Play Store from December 2009 to December 2017* [online]. Statista, 2018, [cited 2018-01-19]. Available at: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[37] AppBrain. *Number of Android applications* [online]. AppBrain, 2018, [cited 2018-01-19]. Available at: https://www.appbrain.com/stats/number-of-android-apps

[38] LIU, Xing, et al. *Characterizing Smartwatch Usage In The Wild* [online]. In: Proceedings of the 15th Annual International Conference on Mo-

bile Systems, Applications, and Services. ACM, 2017 [cited 2018-01-19]. p. 385-398. Available at: https://pdfs.semanticscholar.org/0cc2/4bccc3067ed688e576603bc6bab0e5e1b1db.pdf

[39] Liu, Renju, and Felix Xiaozhu Lin. *Understanding the Characteristics of Android Wear OS* [online]. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services. ACM, 2016 [cited 2018-01-19], p. 151-164. Available at: https://athena.smu.edu.sg/mobisys/backend/mobisys/assets/paper_list/pdf_version/paper_12.pdf

[40] Sandra Henshaw. *Android Wear Problems (and Solutions!)* [online]. Tiger Mobiles Limited, 2016 [cited 2018-01-20]. Available at: https://www.tigermobiles.com/2016/01/android-wear-problems-and-solutions/

[41] Simon Hill. *10 of the worst Android Wear problems, and how to fix them* [online]. Designtechnica Corporation, 2017 [cited 2018-01-20]. Available at: https://www.digitaltrends.com/wearables/android-wear-problems/

[42] ADNAN F. *Tizen overtakes Android Wear in smartwatch market share* [online]. SamMobile, 2017 [cited 2018-01-20]. Available at: https://www.sammobile.com/2017/05/11/tizen-overtakes-android-wear-in-smartwatch-market-share/

[43] Paul Lamkin. *Android Wear 2.0: Ultimate guide to the major smartwatch update* [online]. Wareable, 2017 [cited 2018-01-20]. Available at: https://www.wareable.com/android-wear/android-wear-update-everything-you-need-to-know-2735

[44] Elyse Betters and Chris Hall. *Android Wear 2.0: What's new in the major software update for watches?* [online]. Pocket-lint Limited, 2017 [cited 2018-01-20]. Available at: https://www.pocket-lint.com/smartwatches/news/google/139007-android-wear-2-0-what-s-new-in-the-major-software-update-for-watches

[45] Chris Martin. *Android Wear 2.0 news: release date and features* [online]. Tech Advisor, 2017 [cited 2018-01-20]. Available at: https://www.techadvisor.co.uk/new-product/google-android/android-wear-2-3640616/

[46] Android Developers. *Designing for Android Wear* [online]. Android, 2018 [cited 2018-01-20]. Available at: https://developer.android.com/design/wear/index.html

[47] Elyse Betters. *What is Google Assistant, how does it work, and which devices offer it?* [online]. Pocket-lint Limited, 2018 [cited 2018-01-20]. Available at: https://www.pocket-lint.com/apps/news/google/137722-what-is-google-assistant-how-does-it-work-and-which-devices-offer-it

[48] DAN MOREN. *Alexa vs. Siri vs. Google Assistant: Which Smart Assistant Wins?* [online]. Tom's Guide, 2017 [cited 2018-01-20]. Available at: https://www.tomsguide.com/us/alexa-vs-siri-vs-google,review-4772.html

[49] Digital Trends Staff. *Virtual assistant comparison: Cortana, Google Assistant, Siri, Alexa, Bixby* [online]. Digital Trends, 2017 [cited 2018-01-20]. Available at: https://www.digitaltrends.com/computing/cortana-vs-siri-vs-google-now/

[50] Brian Heater. *Comparing Alexa, Google Assistant, Cortana and Siri smart speakers* [online]. TechCrunch, 2017 [cited 2018-01-20]. Available at: https://techcrunch.com/2017/10/08/comparing-alexa-google-assistant-cortana-and-siri-smart-speakers/

[51] Joe Hindy. *Google Assistant vs Siri vs Bixby vs Amazon Alexa vs Cortana – Best virtual assistant showdown!* [online]. Android Authority, 2017 [cited 2018-01-20]. Available at: https://www.androidauthority.com/google-assistant-vs-siri-vs-bixby-vs-amazon-alexa-vs-cortana-best-virtual-assistant-showdown-796205/

[52] Haroon Q. Raja. *Tizen OS: Brief History, Roots, and Current Status* [online]. xda-developers, 2013 [cited 2018-04-16]. Available at: https://www.xda-developers.com/tizen-os-brief-history-roots-and-current-status/

[53] *Tizen Members* [online]. Tizen Association, 2014 [cited 2018-04-16]. Available at: https://www.tizenassociation.org/members/

[54] *Tizen* [online]. Tizen Project, 2012 [cited 2018-04-16]. Available at: https://www.tizen.org/about

[55] *Tizen Developers* [online]. Tizen Project, 2012 [cited 2018-04-16]. Available at: https://developer.tizen.org/development/tizen-studio

[56] Rhiannon Williams. *A timeline of how the Apple Watch was created* [online]. The Telegraph, 2015 [cited 2018-04-17]. Available at: http://www.businessinsider.com/a-timeline-of-how-the-apple-watch-was-created-2015-3

[57] *watchOS* [online]. 9to5Mac, 2018 [cited 2018-04-17]. Available at: https://9to5mac.com/guides/watchos/

[58] *Apple Developer* [online]. Apple, 2018 [cited 2018-04-17]. Available at: https://developer.apple.com/

[59] Jason Fitzpatrick. *How to Find and Install Apps on Your Apple Watch* [online]. How-To Geek, 2015 [cited 2018-04-17]. Available at: https://www.howtogeek.com/230224/how-to-find-and-install-apps-on-your-apple-watch/

[60] Mishaal Rahman. *No future for Android Wear in Samsung's Watches* [online]. xda-developers, 2016 [cited 2018-04-16]. Available at: https://www.xda-developers.com/report-samsung-is-done-with-android-wear-os/

[61] GSMArena. *Xiaomi Redmi Note 4 - Full phone specifications* [online]. GSMArena, 2018 [cited 2018-01-21]. Available at: https://www.gsmarena.com/xiaomi_redmi_note_4-8531.php

[62] XiaomiMobile. *Xiaomi Redmi Note 4 LTE* [online]. XiaomiMobile, 2018 [cited 2018-01-21]. Available at: https://xiaomimobile.cz/xiaomi-redmi-note-4-pro-lte-global.html?search_query=note+4&results=16

[63] Android Authority Team. *Best Android Wear watches (old version)* [online]. Android Authority, 2017 [cited 2018-01-21]. Available at: https://www.androidauthority.com/best-android-watches-572773/

[64] James Peckham. *Best Android Wear watch 2018: our list of the top Google OS smartwatches* [online]. TechRadar, 2017 [cited 2018-01-21]. Available at: http://www.techradar.com/news/wearables/every-android-wear-smartwatch-in-the-world-today-1288283

[65] Michael Simon. *Best Android Wear watches of 2017* [online]. PCWorld, 2017 [cited 2018-01-21]. Available at: https://www.pcworld.com/article/3209668/android/best-android-wear-watches-of-2017.html

[66] LG Electronics. *LG Watch Sport™ - AT&T* [online]. LG Electronics, 2018 [cited 2018-01-22]. Available at: http://www.lg.com/us/smart-watches/lg-W280A-sport

[67] LG Electronics. *LG Watch Style* [online]. LG Electronics, 2018 [cited 2018-01-22]. Available at: http://www.lg.com/us/smart-watches/lg-W270-Titanium-style

[68] HUAWEI Technologies Co. *HUAWEI WATCH 2* [online]. HUAWEI Technologies Co., 2018 [cited 2018-01-22]. Available at: http://consumer.huawei.com/en/wearables/watch2/specs/

[69] Polar Electro. *Polar M600* [online]. Polar Electro, 2018 [cited 2018-01-22]. Available at: https://support.polar.com/e_manuals/M600/Polar_M600_user_manual_English/Content/technical-specifications.htm

[70] ASUSTeK Computer Inc. *ASUS ZenWatch 3* [online]. ASUSTeK Computer Inc., 2018 [cited 2018-01-22]. Available at: https://www.asus.com/us/ZenWatch/ASUS-ZenWatch-3-WI503Q/specifications/

[71] Adam Conway. *How to Pair Android Wear Watches to New Phones without Factory Resetting* [online]. xda-developers, 2017 [cited 2018-01-22]. Available at: https://www.xda-developers.com/pair-android-wear-without-factory-reset/

[72] Dennis Troper. *Android Wear, it's time for a new name* [online]. Google, 2018 [cited 2018-04-02]. Available at: https://www.blog.google/products/wear-os/android-wear-its-time-new-name/

[73] Locatify. *Indoor Positioning Systems based on BLE Beacons – Basics* [online]. Locatify, 2015 [cited 2018-01-27]. Available at: https://locatify.com/blog/indoor-positioning-systems-ble-beacons/

[74] Patrick Leddy. *10 Things About Bluetooth Beacons You Need to Know* [online]. pulsate, 2015 [cited 2018-01-27]. Available at: http://academy.pulsatehq.com/bluetooth-beacons

[75] *Eduroam* [online]. eduroam, 2018 [cited 2018-04-19]. Available at: https://www.eduroam.org

[76] The Estimote Team Blog. *Reality matters — Preorder for Estimote Beacons available, shipping this summer* [online]. Estimote, Inc., 2013 [cited 2018-01-27]. Avail-

able at: http://blog.estimote.com/post/57087851702/preorder-for-estimote-beacons-available-shipping

[77] *Estimote SDK for Android* [online]. Estimote, 2018 [cited 2018-01-22]. Available at: https://github.com/Estimote/Android-SDK

[78] Radek Brůha. *Pokročilé metody rádiové indoor lokalizace* [online]. Univerzita Hradec Králové, 2017 [cited 2018-01-22]. Available at: https://theses.cz/id/jss047

[79] *Android Beacon Library* [online]. Radius Networks, 2018 [cited 2018-01-22]. Available at: https://altbeacon.github.io/android-beacon-library/

[80] *AltBeacon* [online]. AltBeacon, 2018 [cited 2018-01-22]. Available at: http://altbeacon.org/

[81] *Eddystone format* [online]. Google Developers, 2018 [cited 2018-01-22]. Available at: https://developers.google.com/beacons/eddystone

[82] *NOSQL Databases* [online]. NoSQL, 2018 [cited 2018-01-27]. Available at: http://nosql-database.org/

[83] *NOSQL Databases* [online]. NoSQL, 2018 [cited 2018-01-27]. Available at: http://nosql-database.org/

[84] Brown, Martin C. *Getting Started with Couchbase Server: Extreme Scalability at Your Fingertips* [online]. O'Reilly Media, Inc., 2012 [cited 2018-01-27]. Available at: https://books.google.cz/books?hl=cs&lr=&id=5xu33G9LGkMC&oi=fnd&pg=PR5&dq=Couchbase&ots=nrw7O3HiVh&sig=6qmpVJLxwdOK9RRZsICHUfsD2wI&redir_esc=y#v=onepage&q&f=false

[85] *What is N1QL?* [online]. Couchbase, 2018 [cited 2018-01-27]. Available at: https://www.couchbase.com/products/n1ql

[86] *Explain Relational Database Management System (RDBMS)* [online]. W3Schools, 2016 [cited 2018-01-23]. Available at: http://whatisdbms.com/explain-relational-database-management-system-rdbms/

[87] *What Is SQLite* [online]. SQLite Tutorial, 2018 [cited 2018-01-23]. Available at: http://www.sqlitetutorial.net/what-is-sqlite/

[88] Sterling Quinn, John A. Dutto. *Why tiled maps?* [online]. e-Education Institute, College of Earth and Mineral Sciences, The Pennsylvania State University, 2018 [cited 2018-04-08]. Available at: `https://www.e-education.psu.edu/geog585/node/706`

[89] Mike Dunn. *TileView* [online]. Mike Dunn, 2016 [cited 2018-04-11]. Available at: `https://github.com/moagrius/TileView`

[90] Mike Dunn. *Creating Tiles* [online]. Mike Dunn, 2016 [cited 2018-04-11]. Available at: `https://github.com/moagrius/TileView/wiki/Creating-Tiles`

[91] *Scheduling of tasks with the Android JobScheduler - Tutorial* [online]. vogella, 2017 [cited 2018-04-06]. Available at: `http://www.vogella.com/tutorials/AndroidTaskScheduling/article.html`

[92] *Google Tag Manager DataLayer Explained* [online]. Analytics Mania, 2017 [cited 2018-04-08]. Available at: `https://www.analyticsmania.com/post/what-is-data-layer-in-google-tag-manager/`

[93] KELLY JR, James D.; DAVIS, Lawrence. *A Hybrid Genetic Algorithm for Classification*. In: IJCAI. 1991 [cited 2018-04-13]. p. 645-650. Available at: `https://www.ijcai.org/Proceedings/91-2/Papers/006.pdf`

Attachments

1. CD containing following content
    a) /applications - contains data for all applications created
        i. /mobile_wear - source code for android applications
        ii. /server - source code for server application
    b) /evaluation - results of the analysis and data used for it
        i. /gnuplot_scripts - gnuplot scripts used to draw data using graphs
        ii. /images - images created during evaluation
        iii. /raw_data - all data from the evaluation in raw format
    c) /tex - thesis text in tex format
2. Public Github repository with Android applications (mobile and wear) and thesis text:
   `https://github.com/Del-S/WearFingerprint`
3. Public Github repository with server application:
   `https://github.com/Del-S/WearFingerprintServer`
4. Public Github repository with evaluation code in a separate branch for this project:
   `https://github.com/pavkriz/radio-localization-eval/tree/sucharda`

Univerzita Hradec Králové
Fakulta informatiky a managementu
Akademický rok: 2017/2018

Studijní program: Aplikovaná informatika
Forma: Prezenční
Obor/komb.: Aplikovaná informatika (ai2-p)

Podklad pro zadání DIPLOMOVÉ práce studenta

PŘEDKLÁDÁ:	ADRESA	OSOBNÍ ČÍSLO
Bc. Sucharda David	Masarykovo náměstí 3, Nová Paka	I1500697

TÉMA ČESKY:

Sběr rádiových fingerprintů pomocí chytrých hodinek

TÉMA ANGLICKY:

Radio Fingerprint Acquisition Using a SmartWatch

VEDOUCÍ PRÁCE:

Ing. Pavel Kříž, Ph.D. - KIKM

ZÁSADY PRO VYPRACOVÁNÍ:

Cíl: Navrhnout a implementovat aplikaci pro mobilní telefon a chytré hodinky na platformě Android Wear 2.0, pomocí které bude možné změřit rádiové fingerprinty souběžně pomocí mobilního telefonu a hodinek. Vyhodnotit a porovnat přesnost indoor lokalizace s použitím existujících algoritmů využívajících rádiové fingeprinty.

Osnova:
1. Úvod
2. Indoor lokalizace s pomocí rádiových fingerprintů
3. Platforma Android Wear 2.0
4. Analýza a návrh
5. Implementace
6. Testování, zhodnocení výsledků
7. Závěr

SEZNAM DOPORUČENÉ LITERATURY:

https://www.hindawi.com/journals/misy/2016/2083094/
https://www.microsoft.com/en-us/research/project/radar/
https://developer.android.com/wear/index.html
http://www.sciencedirect.com/science/article/pii/S1877050912005170
https://link.springer.com/article/10.1007%2Fs13218-017-0496-6
http://www.mdpi.com/1424-8220/17/6/1299
http://www.mdpi.com/1424-8220/17/6/1339
http://www.mdpi.com/1424-8220/17/8/1789

Podpis studenta: ..... **Datum:** 15. 11. 17

Podpis vedoucího práce: ..... **Datum:** 15.11.17