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.

Thanks

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

1 Introduction **1**

1.1 Goals of this thesis . . . 2

1.2 Reason for selection of this topic . . . 2

2 Localization techniques **4**

2.1 Triangulation . . . 4

2.1.1 Lateration . . . 4

2.1.2 Angulation . . . 5

2.2 Fingerprinting . . . 6

2.3 Proximity . . . 7

2.4 Other techniques . . . 8

3 Related Work **9**

3.1 Improving Precision by Using Multiple Wearable Devices . . . 9

3.2 SmartFix . . . 11

3.3 SmartWatch vs. SmartPhone . . . 12

4 Android **14**

4.1 Android system structure . . . 14

4.2 Wear technologies . . . 16

4.2.1 Android Wear . . . 16

4.2.1.1 Standalone applications . . . 17

4.2.1.2 UI improvements . . . 18

4.2.1.3 Google Assistant . . . 19

4.3 Other wear technologies . . . 19

4.3.1 Tizen . . . 19

4.3.2 WatchOS . . . 20

5 Application design and implementation **22**

5.1 Hardware . . . 23
5.1.1 Smartphone and Smartwatch . . . 23
5.1.1.1 Phone . . . 23
5.1.1.2 Smartwatch . . . 23
5.1.2 Radio signal devices . . . 25
5.1.2.1 BLE beacons . . . 25
5.1.2.2 WiFi access-points . . . 26
5.2 Server . . . 26
5.3 Application Software . . . 27
5.3.1 AltBeacon Library . . . 27
5.3.2 Database . . . 28
5.3.2.1 SQLite database . . . 28
5.3.2.2 Couchbase database . . . 29
5.3.2.3 Comparison . . . 30
5.3.3 TileView . . . 30
5.4 Application implementation . . . 32
5.4.1 Scanner . . . 32
5.4.1.1 JobScheduler . . . 33
5.4.1.2 BroadcastReceiver . . . 34
5.4.2 Device communication . . . 35
5.4.2.1 Data Layer API . . . 35
5.4.3 Server communication . . . 36
5.4.3.1 Retrofit . . . 37
5.4.4 Application screens . . . 38
5.4.4.1 Map and scanner . . . 38
5.4.4.2 Devices . . . 40
5.4.4.3 Synchronization . . . 41
5.4.4.4 Wear scanner . . . 42
5.4.5 Interesting code examples . . . 43
5.4.5.1 Sending Fingerprint to Wear . . . 43
5.4.5.2 Parsing beacon information . . . 44
5.4.5.3 Map configuration . . . 45

6 Testing and data analysis 48

6.1 Testing environment . . . 48

6.2 Evaluation approach 49
6.2.1 WKNN 50
6.3 Data collection 51
6.3.1 First data collection 51
6.3.2 Second data collection 52
6.3.3 Third data collection 54
6.4 Evaluation 56
6.4.1 Decide K values for WKNN 56
6.4.2 Compare device technologies 57
6.4.3 Testing multiple fingerprints 59
6.4.4 Combining fingerprint data 60
6.4.5 Map comparison 62

7 Conclusion **64**
7.0.1 Future improvements 65

Literature **66**

Attachments **77**

List of figures

1.1 Comparison of Positioning Technologies (source: [4]) 2

2.1 2D and 3D Trilateration (source: [10]) 4
2.2 Multilateration (source: [11]) 5
2.3 3D location using AoA from Quuppa Intelligent Locating System (source: [15]) 6
2.4 Cell of Origin (source: [18]) 7

3.1 Energy consumption based on prototypes (source: [24]) 12
3.2 Summary table of results (source: [27]) 13

4.1 Android stack (source: [29]) 14
4.2 Smartwatch OS market share (source: [42]) 17
4.3 Wear design examples (source: [46]) 18
4.4 Wear watch faces (source: [43]) 18
4.5 Tizen showcase (source: [60]) 20
4.6 WatchOS showcase (source: [59]) 21

5.1 Application architecture (based on [9]) 22
5.2 Parts of Estimote beacon (source: [75]) 25
5.3 Tile pyramid for zoom levels (source: [87]) 31
5.4 Map screen with markers 38
5.5 Scan status 39
5.6 Maximum map zoom (left), List of fingerprints (right) 40
5.7 Bluetooth devices (left), Beacons (right) 41
5.8 Synchronization screen 42
5.9 Wear scanning screens 43

6.1 Map of deployed devices (based on [9]) 48
6.2 Map of errors for BLE in meters for all fingerprints 53

6.3 Map of errors for BLE (left) and WiFi (right) 55
6.4 Comparison of localization accuracy for $k = 2$ 57
6.5 Number of transmitters for all fingerprints 58
6.6 Comparison of errors based on device 59
6.7 Comparison of localization accuracy for testing multiple fingerprints 60
6.8 Comparison of localization accuracy for combining fingerprints 62
6.9 Maps of errors for all algorithms with last one for mobile error only 63

List of tables

3.1 Mean errors at first location (sources: [23]) 10
3.2 Mean errors at second location (sources: [23]) 11

5.1 Smartwatch comparison (sources: [66–70]) 24
5.2 Couchbase vs SQLite (sources: [66–70]) 30

6.1 Maximum errors for second data collection 52
6.2 Scanning information for wear (second scan) 54
6.3 Scanning information for wear (third scan) 55
6.4 List of errors for multiple K values 56
6.5 Device comparison: mean and max errors (in meters) 57
6.6 List of errors for testing multiple fingerprints 60
6.7 List of errors for fingerprint combination 61

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.

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]. Description of few algorithms can be found in Chapter 2.

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

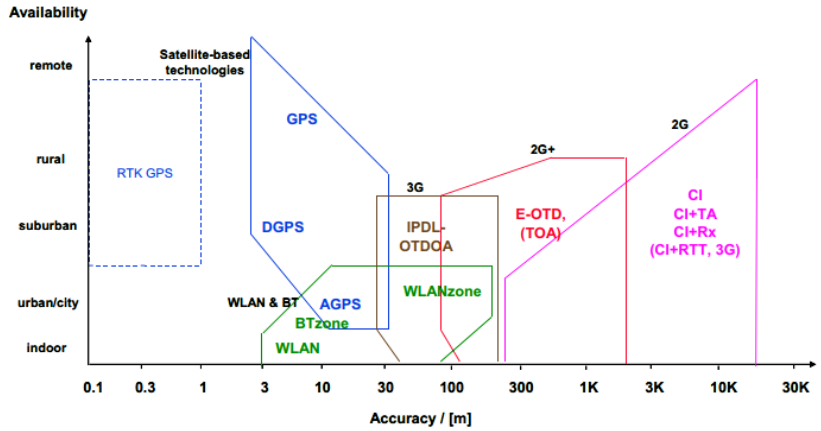

Figure 1.1: Comparison of Positioning Technologies (source: [4])

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 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 Android operation system called WearOS by Google (used to be Wear 2.0) and it is basically 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 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.

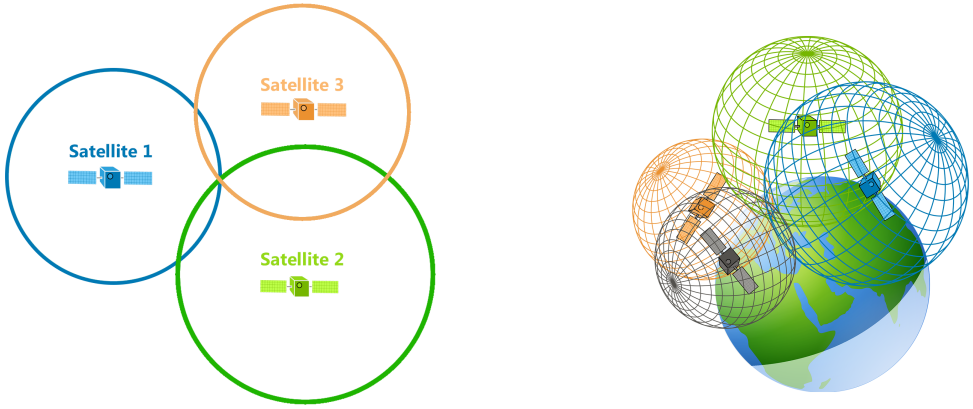

Figure 2.1: 2D and 3D Trilateration (source: [10])

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.

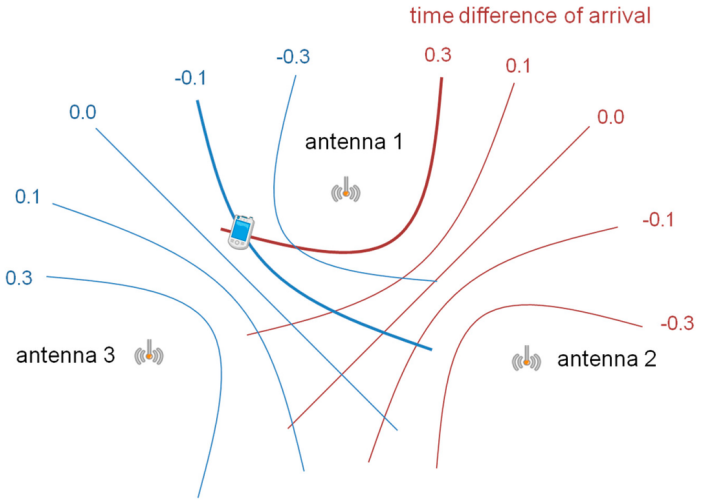

Figure 2.2: Multilateration (source: [11])

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 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].

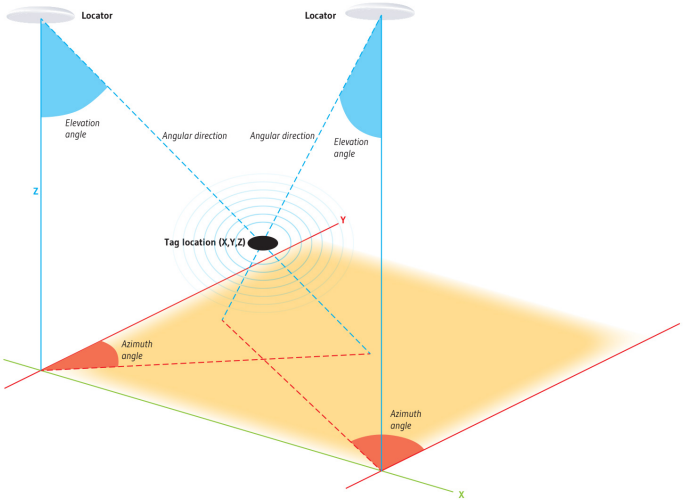

Figure 2.3: 3D location using AoA from Quuppa Intelligent Locating System (source: [15])

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 used 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 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 stations. 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].

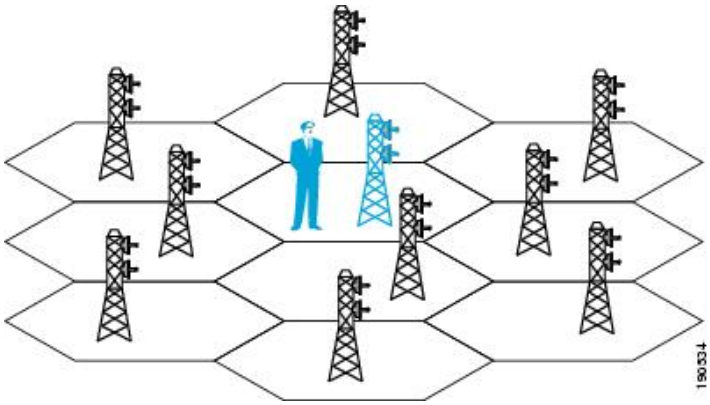

Figure 2.4: Cell of Origin (source: [18])

Main advantage of this approach is very easy implementation and no need for complicated algorithms, thus 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 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 applications, such as image and speech recognition as well as location [20]. The advantage 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 it will compare its observed scenes [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, calculate 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 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

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 second mobile device, in this case with Nexus 5 running Android 4.4.

This paper proposes calculating medians from 800 millisecond tests where user can move one meter 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. One thing to note is that fingerprint maps are also built based on facing direction. There are five main scenarios tested

- 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 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.

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 edge 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.

	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])

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 of this paper is to improve energy consumption efficiency for wearable-based indoor localization systems using WiFi fingerprinting. At the begging single real-time experiment of energy consumption was run and split into two main parts: computation of location and collection of fingerprints. 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 a large decrease of power consumption for signal collection. Aided with machine-learning algorithm, it obtains the relative features given 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 fingerprint 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 excellent energy saving performance. Figure 3.1 shows power consumption of on-time locating on two specific devices: HTC one and Moto 360.

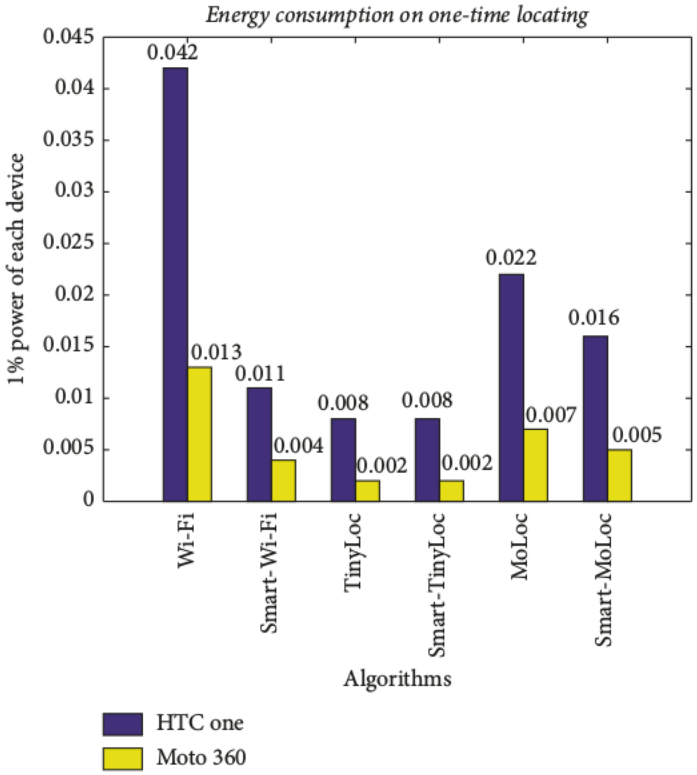

Figure 3.1: Energy consumption based on prototypes (source: [24])

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 meter 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 result 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 penson.

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. 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].

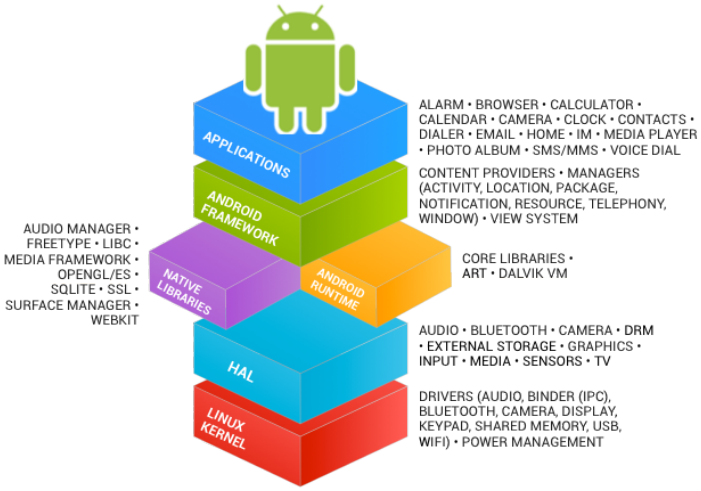

Figure 4.1: Android stack (source: [29])

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 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 new Kotlin programming language [34]. These application are packaged into `.apk` files, which is a zip archive, containing all application files like 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.xml` , Internet is granted automatically but location must be manually granted by the 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 platform designed to be open source and free which makes it easy to create malicious applications. These application can bypass existing security and steal sensitive data, use telephony services or even gain control over the device [35]. Android has multiple ways to protect against such applications one 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 phones or tables in term of usage, design and user interfaces (UI). According to the app design guidelines by major vendors, users interact with wearable devices frequently throughout daily use. Each interaction is short, often less than 10 seconds, and is dedicated to simple tasks [31].

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

One problem with this diversity is hardware and software compatibility. Every device creator can create their own operating system for specific wear device and it can be difficult to develop custom applications for it. To avoid such problems this thesis is focused only on smartwatch with Android Wear operating system.

There are three main points to note with watch devices. First, small battery capacity that can be almost ten times smaller then in typical smartphone. Second, point is display with around forty-times less pixels which completely changes properties. Final, 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 just hours.

4.2.1 Android Wear

Android Wear is a version of Android OS tailored to small-screen wearable devices. There are not too many changes from system for 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 Android Wear wirelessly offloads data to the smartphone for heavy computing tasks, e.g., voice recognition [39].

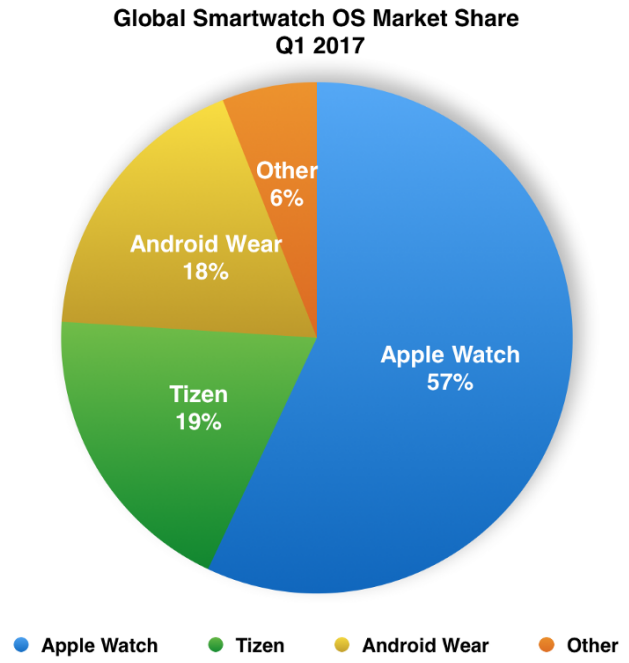

Figure 4.2: Smartwatch OS market share (source: [42])

Android Wear is one of the most popular smartwatch systems but comes with it is own set of problems. Most notable and annoying one is being unable to pair wear with specific mobile device. It can be caused by combination of things such as system compatibility, custom hardware or phone type but it is more common then it should be [40]. Having phone connected to Android Wear can also cause phone to drain battery way faster. One thing to note is that smartwatch can be connected to only single device and connecting to another requires factory reset. Few other problems can be update issues, notifications not coming through to the watch, not being able to connect to Wi-Fi and system crashes [41].

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

4.2.1.1 Standalone applications

This feature is crucial change and it means that watch applications do not need mobile phone to function. Before this version it was needed to have connected phone with Android Wear support to use applications. Being forced to have Android phone proved as an obstacle for users without one since they could not use any applications on the watch [43, 44].

Since application can now work without phone there should be a way to install them directly. Thankfully part of this update is also standalone Google Play Store where users can browse apps that are designed specifically for the watch [44]. Part of this feature is also enabling watch to use wireless and cellular networks on their own since most standalone applications require this feature. And final part of this update was improving and securing communication with phone. This is now done via Wearable Data Layer API that is used in almost all Google applications and it is also pretty easy to use as 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].

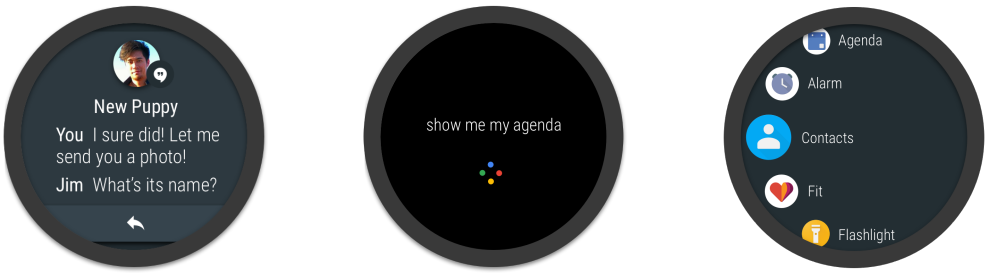

Figure 4.3: Wear design examples (source: [46])

Android is also trying to catch up with Apple's watchOS and make default watch display, also called watch faces, much more useful. Users can add different widgets of any containing data of any application to the watch faces [43]. This ensures quick access to the data user deems important [45]. All this data displays also match design of currently selected watch face and after clicking it will direct 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 same as for example 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 to be that different so there is no need to buy one over the other. These systems can pull information that you need or want and they track where you work, sports you like, your schedule, stuff that might interest you and much more [47]. With the update of Wear 2.0 this feature is now available on smartwatches [43, 44].

4.3 Other wear technologies

During the first years of smartmatches big amount of manufacturers created their own system and flooded the market but that was a long time ago. Now there are only three main competitors as shown in the Figure 4.2 and those are Android (WearOS), Samsung (Tizen) and Apple (WatchOS). Android system 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 like MeeGo by Nokia and Intel, Samsung Linux Platform (SLP) or Bada, both created by Samsung. In the end Intel joined forces with Samsung and created Tizen organization and project which was based on SLP and either canceled support of previous systems or merge 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 part of Linux Foundation. It is focused on the needs of all stakeholders of the mobile and connected device ecosystem and it has multiple profiles based on the hardware on which it should run, such as mobile, tv, and wearable. One of this system focuses is easy support for manufacturers with easily changeable profiles to suit the needs of a specific manufacturer. Tizen also offers the power of native application development with the flexibility of unparalleled HTML5 support [54].

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

Developing application for this system can be done in a custom made program Tizen Studio which supports developing native or Web applications where native are developed using the C programming language, but since Tizen version 4.0 application can also be developed using .NET or Xamarin UI. The studio contains all required part for developing an application from project management, writing and editing code, debugging and running the application. Newest version of Tizen Studio is 2.3 with the support of Windows, Ubuntu and macOS where next version will remove support for 32 bit Windows and Ubuntu. These applications created can be deployed to Taizen store similar to Google Play [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 Wear 2.0.

4.3.2 WatchOS

Apple started working on smartwatches around year 2002 inspired by sportswatches from Nike with actual results in 2014 when Apple Watch was revealed to be released in 2015. With its release it introduced its own system watchOS, based on iOS, which is the youngest of all three mentioned but the most popular [56]. watchOS is currently developed and deployed only for Apple Watches same as Tizen for Samssung making Google's WearOS only system

to be implemented by multiple wear manufacturers. First version of watchOS was based on the same principal as watchOS because it required mobile device to use its applications but Apple quickly moved from this interaction in version 2.0. Next versions brought many improvements, such as running application in background, improved design and support completely mobile independent application from version 4.0 [57].

Applications of this system consist of two related bundles: a Watch app and a WatchKit extension. The Watch app bundle contains the storyboards and resource files associated with application 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 created by Apple. Main programming language used in this program is Swift which is also created by Apple and based on C and Object-C language, which can be also used to program iOS applications. Swift is an open source language focused on being easy to understand and still powerful and fast. Same as previous wear systems created applications can be deployed using App Store similar to Google Play or Tizen store.

5 Application design and implementation

This chapter describes all important information about created application. First is hardware and software used for developing and testing of the application. Second is structure and description of core parts used in the application.

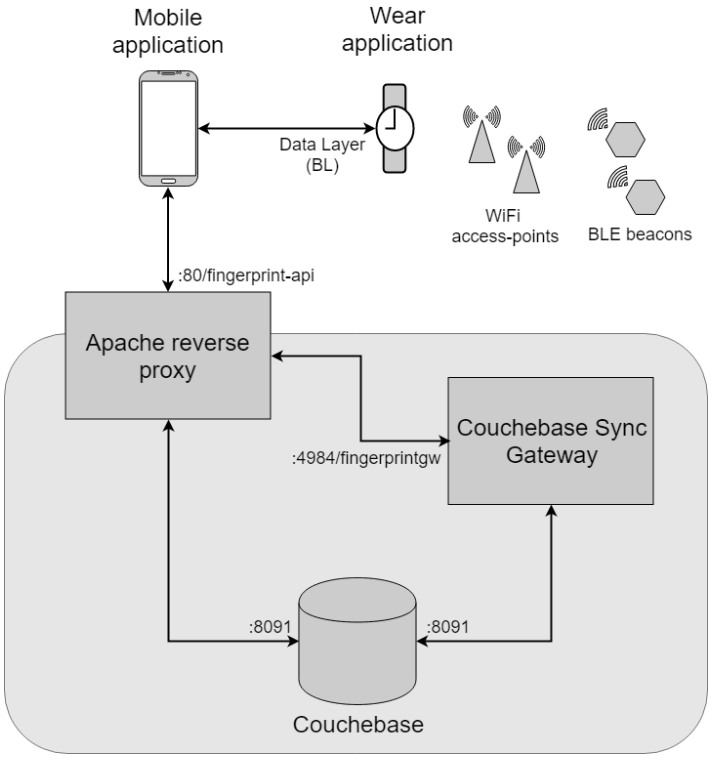

Figure 5.1: Application architecture (based on [9])

Figure 5.1 shows basic devices and technologies used in this application. There are three main part of this implementation, mobile, wear and server application. Mobile and wear parts implement scanning for radio signals and 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. Main goal of the server is to store scanned data.

5.1 Hardware

There are multiple hardware devices used in this application. All of them can be seen on Figure 5.1, where first and the most obvious are smartphone and smartwatch. These devices scan radio signals from WiFi access-points and BLE beacons that are placed in the indoor environment. And a final hardware part is a server holding scanned data from all devices.

5.1.1 Smartphone and Smartwatch

Both smart devices must support scanning of Bluetooth Low Energy beacons that can be done with Bluetooth 4.0 and higher. Secondary requirements are Wi-Fi, GSM and LTE modules to support more data types than just BLE beacons.

5.1.1.1 Phone

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 so main requirement for the hardware is met, with all the secondary requirements also met with Wi-Fi 802.11 a/b/g/n, GSM, and LTE support like most modern smartphones would [61].

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

5.1.1.2 Smartwatch

First goal of this thesis was to select smartwatches running on Android Wear 2.0 technology. This iteration is quite new currently, which makes it harder to select proper wear device since there are not so many options. During selection process there were around twenty of watches with this system and only five of them were selected to closer inspection based on few articles [63–65].

Only one smartwatch could be selected out of five displayed in Table 5.1. Firstly, wear device had to support BLE and Wi-Fi which all have. Secondly, it must have been sold in

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])

Czech Republic (CR) since it makes shipping and warranty 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]. Finally, selected device was Huawei Watch 2 since there were not too many problems in reviews and other requirements were met.

Initial setup of the wear device was composed of two main parts. First, update wear system which took about one to two hours. Second task was to setup the watch and copy Google account, this is where some problem were discovered. Copying of accounts from Redmi Note 4 to the watch never completed. To fix this problem another smartphone (Huawei Y5 II) was used to copy the account, it was already mentioned that only single device can be connected to smartwatch and connecting to a new one requires factory reset removing all the data. It was needed to pair wear with phone without factory reset which was handled via Android Debug Bridge following this [71] article.

Since Google wants to focus also on iOS devices technology Android Wear 2.0 was re-branded to Wear Os by Google to remove confusion in the future [72]. This updated was forced on wear devices, updating it to the newest Android version which forced some changes in the development of this solution. First, implementation bug in scanning library was introduced and had to be fixed. Second and bigger problem was inability to deploy new version of the application from Android Studio to the watch. This, for some reason, effects the main computer used to develop this application and it was fixed by using another computer.

5.1.2 Radio signal devices

As hardware devices are used BLE beacons, WiFi access-points and Cellular towers. Only first two types of these devices can be placed inside a building and are most commonly used in indoor localization. Signal from cellular towers is only a complimentary data that does not have to be used and cannot be influenced since they are placed 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 in 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 the advertising interval and the transmitter output power [73]. More in depth description of this technology and devices was written by Pavel Kriz et al. [9].

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

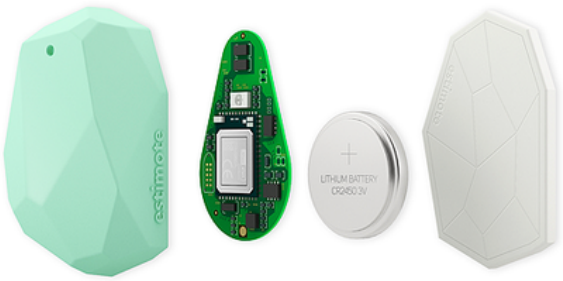

Figure 5.2: Parts of Estimote beacon (source: [75])

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

5.1.2.2 WiFi access-points

WiFi signals are commonly used in indoor localization due to their presence almost anywhere. In this case several WiFi transmitters of the `eduroam` network made by Cisco were used. They are permanently deployed on every floor of a test building and cannot be influenced or changed.

5.2 Server

Implemented application do not necessarily need a server for its core functionality but it is common and faster to run an analysis of the data and other heavy computational work on a different, faster device. Functionality of the server is to serve as a backup of the data and also as a synchronization point to enable other devices to download fingerprints and upload new ones.

The server infrastructure was built in previous years [9] but it was improved for this solution. It uses NoSQL database to save fingerprints. This type of database does not have fixed scheme, making it easy to save all kinds of data. There are many NoSQL databases, around 250 at this time, to choose from. Selected database is called Couchbase, it saves data in JSON files with its own query language similar to SQL. Part of this database can also replicate data between other devices which could be used in the application part. In the end it was deemed too slow and data heavy for usage in the application, creating the need for creation of a middleman API between Android application and Couchbase database.

As Figure 5.1 shows server has three main parts to work with. First, already mentioned is Couchbase database to keep scanned data. Second, sync gateway that can synchronize data between devices, not used in this implementation. Both of these part were already implemented in previous years [9]. Final part is 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. It was documented using Swagger and it handles two main HTTP routes: `fingerprints` and `fingerprints-meta`. Before any tasks are issued, mobile application checks data on

the server using `fingerprints-meta` route which returns count of new fingerprints and last time specific device saved fingerprints on the server. With this information the application knows how many fingerprints to upload and download, in this specific order, to ensure data is stored first. To prevent device memory exhaustion and connection timeout both of these tasks have specific limits, download can process 100 of fingerprints while upload only 20. Server can also generate data files for analysis, split by technology, time, origin device and other filters. To simplify this generation it is similar loading all fingerprints, meaning it does not create file but prints the data 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 Couchbase directly sync gateway will not display them, that is why the upload must go through the gateway. This is to ensure all previous and next solution can use this feature without loosing any important data.

5.3 Application Software

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

5.3.1 AltBeacon Library

Since Android core does not allow scanning for BLE beacons 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 [76] which was already used in previous thesis [77]. To change things up BLE beacons are found via AltBeacon Library [78].

Since there was no open and inter-operable specification for proximity beacons, Radius Networks has created the AltBeacon specification as a proposal to solve this issue. It is an open and free specification for BLE beacons with focus to create an open, competitive market for implementation of these devices [79]. Basic configuration of this library can scan for beacons based on this standard and it also supports Eddystone beacons which is Google's open source format. To support already mentioned Estimote beacons this library configuration must be altered to support their detection. Luckily this function can be easily modified to support different kinds of beacons by just one following line of code [78, 80].

```
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 a scan per client. Update to Android 8 introduced a bug in this feature making it unable to confirm the registration of client so the scanner has to be reworked 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, is SQLite database implemented in Android mobile application to save Fingerprints from smartphone and smartwatch. This database is default implementation and most commonly used in Android applications. Second database type is Couchbase implemented on the server `beacons.uhk.cz` to keep all Fingerprint data in one place and this enable synchronization and 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 [86]. These tables usually have 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 of this database type and it makes calculation faster but usually uses more storage space. Other advantage is that data can be only saved once since 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 with complexity and inflexibility of database scheme because it is hard to setup and does not allow other data then is defined in the tables [85]. Altering schemes in the future can be really complex since there is the need to parse all previous data to the new tables which does not need to have the same scheme as previous 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 [86].

- 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, ...).

5.3.2.2 Couchbase database

Since SQL based databases can be complex to implement, scale and usually require more data storage space there was a motivation to create so called NoSQL databases. Their most important and significant feature is not having a fixed scheme, that makes them easy to scale and replicate between multiple devices. There are around 225 NoSQL databases at this time and selected Couchbase is one of them [82].

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 to create a database cluster which holds all the data in multiple locations (nodes) [83]. 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 is used for decades and it became standard for working with data in databases, Couchbase embraces this approach and extends it for JSON files, this language is called N1QL. It has all the main features of the SQL with some minor improvements [84]. Currently this language can be used only on the server implementation, meaning Android cannot use this feature. Version for mobile application is called Lite and instead of N1QL so called

`views` are used, they are objects containing all the selected data from the documents. Major problem of `views` is being very data heavy which is shown in Comparison section. That is one of two main points why mobile and wear applications use an SQLite instead of Couchbase. Second point is to differentiate between previous solutions and test data consumption and load speed of SQL database.

5.3.2.3 Comparison

Both of these database solutions were tested on Android mobile application to figure out which one is faster and takes less data storage space. As a test 315 fingerprint documents will be loaded and displayed, all of them have more than 500 sub-documents which makes about 150 000 documents. As Table 5.2 shows SQLite takes less 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 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 there must be custom solution created which was already described in 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 have to work for bigger ones because the device can run out of memory. To solve this problem it is usually better to try custom 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 (`tiles`) and display them next to each other. This approach has multiple advantages.

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

Tiles are usually 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, where each zoom increase splits this image into 4 tiles. For practical purposes max zoom level is usually around 21 with almost 4,5 trillion tiles.

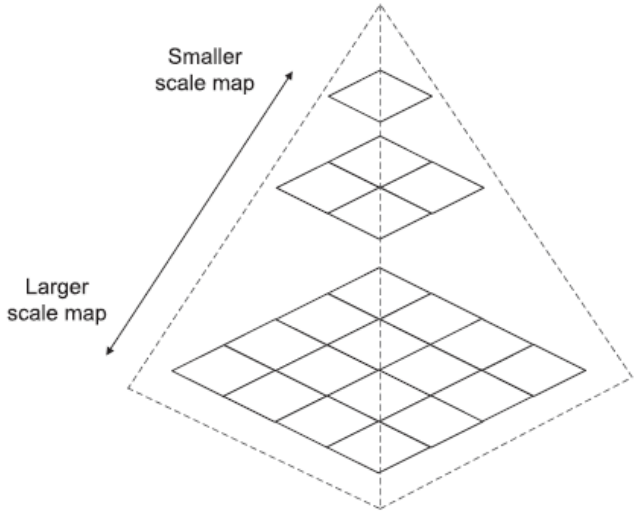

Figure 5.3: Tile pyramid for zoom levels (source: [87])

Creating tiles by hand can be really time consuming job even for low zoom levels it can be done automatically by script or program. For example it is possible to create them by slice tool in PhotoShop but this solution support only low level 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 create a script for all zoom levels required [88].

```
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 [89].

5.4 Application implementation

Implementation of this application is split into two parts, mobile and wear. Both of them have to implement two tasks in order to be able to create fingerprint maps. First, scanner for radio signals which will record Bluetooth Low Energy beacons, WiFi access-points and Cellular towers. Second, communication between both implementation, mobile and wear, to allow sending fingerprint information 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, 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 and selected sensor information. This additional data, especially about the device, will help with analysis by splitting data into 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. This implementation is easily created since it handles basic functions like running task, run code before and after task, publish progress to the main Thread and many more.

- Service - Mainly focused on long running tasks and it is run immediately after it is started.
- Custom Thread - One of the hardest solutions because developer has to handle all the task related to Threads life-cycle and be careful to prevent creation of multiple unwanted Threads.
- JobScheduler - One of the newest implementations, it is focused to schedule task and run them when the device has required resources ready and not immediately.

Scanner is run using JobScheduler, it is a new technology and Android wants to develop this feature. Handling scanned data is implemented using publish-subscribe design pattern handled by `BroadcastReceiver`. There are three separate Receivers for each type of signal collected and one more for collecting data from sensors.

5.4.1.1 JobScheduler

This feature was introduced in Android 5.0 and it remains under active development with major changes in Android version 7.0. This platform collects information about jobs that need to run across all the applications. Some information collected about specific jobs are: required network access, is the job periodic, should the job be scheduled after device restart and more. This information is used to schedule jobs to run at, or around, the time they were scheduled. JobScheduler is intelligent about running jobs and it uses so called batching, which is combining multiple jobs to reduce battery consumption [30, 90].

```
// 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 ID and class to contain job code. Secondly, job information are provided: 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 and this job is run. One thing to remember is to not create a new instance of JobScheduler class since 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 or network connectivity change. In addition to system events, custom ones can be created in the application to inform other applications. 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 is to send a custom broadcast identified by specific action, which can be system or custom. BLE beacon data is added to be received and parse into Fingerprint. Last line sends the actual 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 steps. First, register the receiver implementation and specific broadcast message it should receive, this can be done via manifest or dynamically in code. If receiver is registered using manifest, application is launched (if it is not running already) when the broadcast is sent. Second, create an implementation of BroadcastReceiver class that will handle information and data using `onReceive()` function.

There are two BroadcastReceiver 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 receive information about found 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 the fingerprints such as location and scan duration, these values are 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 Thread to make connection, authorize it and then to send the data when device was connected. Both devices had to accept the connection to send the data, which is normal, but there were some connection loss issues where one devices was considered connected and sending data while the second device was disconnected and not receiving data. That is why this solution was removed at early development stages and substituted by Google's Data Layer API.

To limit data consumption of this application on wear it does not run constantly. Before any scan is started mobile informs wear to start the application, after this is done mobile sends fingerprint data to wear part of the application which is returned after the scan is done with acquired data.

5.4.2.1 Data Layer API

Data Layer is commonly used by Google to send data between their applications. There is an implementation using this feature called Google Tag manager which is sends data from websites to other Google tools like Analytics or Adwords. In this case is a JavaScript object or variable creating virtual `layer` of website application which contains various `data` points, making its name, `Data Layer` [91].

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

- Data Item - It provides a data storage that is synchronized between mobile and wearable device. Whenever data changes all devices using this item are then informed about the change and it is identifies by Uri containing creator 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 wearable. If the device receiving data is connected sender receives a result code confirming successful data send. Although if the device is disconnected right after receiving the result code this information might not be 100% precise.
- 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 copy of the message on local device before synchronization. It can be also used to transfer streamed data, such as music pulled from a network.

This application uses message system and data items, where messages are send to start wear application and confirm its startup and later Fingerprint data are send via data items. Since application may be either in foreground or background there need to be two solutions to receive the data. First, a 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 foreground.

One important thing to note 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 this is a very data and computational heavy task it is run in a separate thread same as a scanner, also using a 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 simple library called Retrofit.

Even though the new version of Android Wear OS 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 simple. Such interface contains calls with parameters based on the api documentation, that can return either raw response or automatically convert the data into Java classes based on configuration. Creating the interface is only one of two parts to make this library working. Second is create an instance of this library's main class with a configuration specific to the api.

```
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.

Small example of Retrofit interface class with a function for getting fingerprints. Firstly, Retrofit must be informed which 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 must inform 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 work and that is defining base URL so the library knows where to post 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 the scan.

5.4.4.1 Map and scanner

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

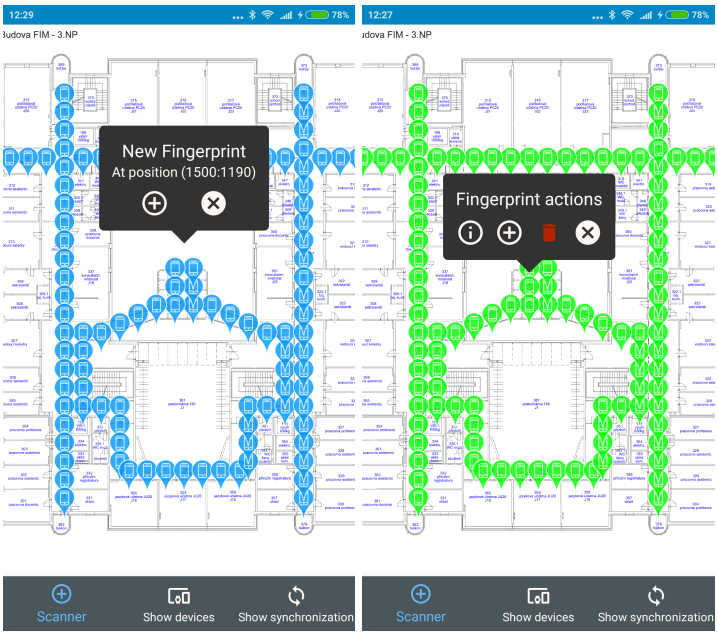

Figure 5.4: Map screen with markers

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 current mobile phone. Images combination distinguishes between device type, mobile or wear in this case.

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

Adding new fingerprints will create all necessary data for a scan and runs 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. This is the case when Devices screen initiated a scan which did not finish just yet. When this 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.

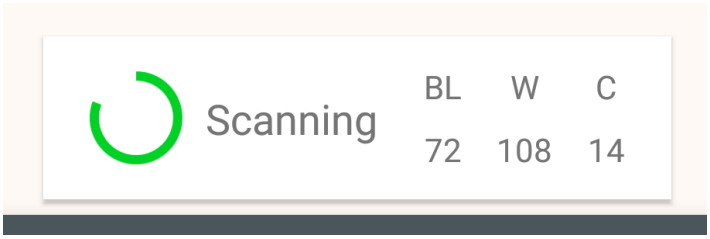

Figure 5.5: Scan status

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. It displays information about BLE (B), WiFi (W) and Cellular (C) data collected but it does not show Sensor (S) information to save space.

Original map image has 3000 x 2200 pixels but with a low 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 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 fingerprint in a specific spot.

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.

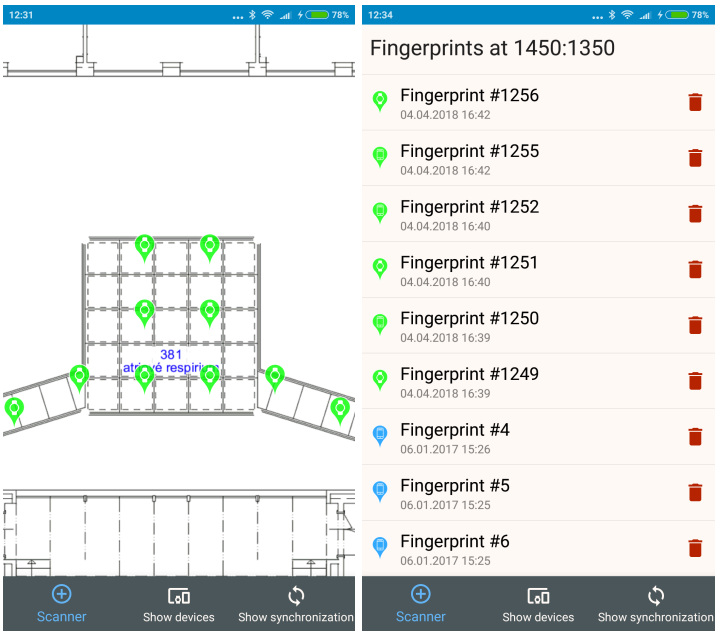

Figure 5.6: Maximum map zoom (left), List of fingerprints (right)

Important thing to note is deleting fingerprints will not be reported to the server, therefore deleting fingerprints is advised before uploading to the server. Deletion 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

It was supposed to be used for establishing connection between mobile and wearable device but this is done via Wear OS application and not necessarily required since Data Layer API and it also can send data between devices via WiFi. Even though it has no use it was kept to display surrounding devices and mainly BLE beacons.

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 the scan by pressing "Search" button, this 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 device is in range or not, this it can 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 some TVs or wristbands. It displays basic information like name, mac address and last signal strength received. Same as Bluetooth list scanning is not continuous and is initiated by pressing "Search" button, length of such scan is set for 30

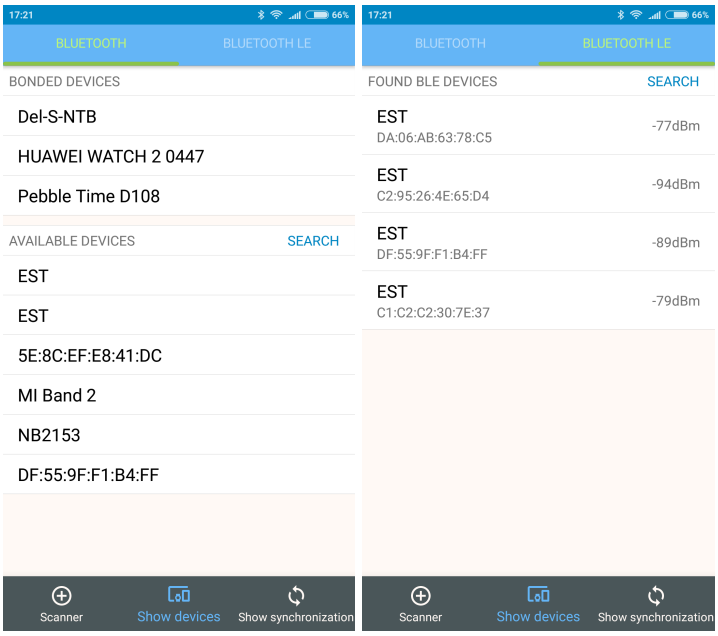

Figure 5.7: Bluetooth devices (left), Beacons (right)

seconds in this case.

There is a possibility for both screens where device does not have a set name, in this case MAC address is displayed instead of 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, this it needs to be triggered by the user. Contrary to scanning information it does not display an overlay with the status, instead of that, after every successful upload or download is done it updates counts on the screen. It also shows an animation of top synchronization button and displays a message if it finished successful or failed.

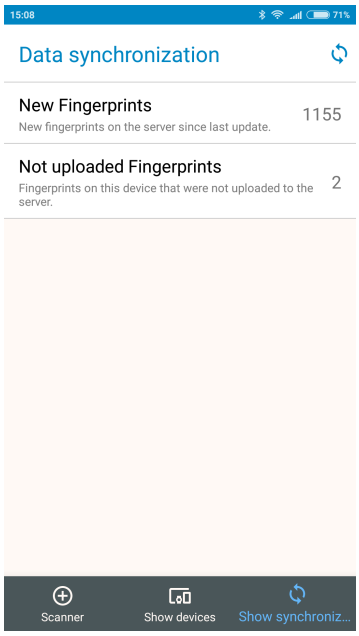

Figure 5.8: Synchronization screen

It is planned to expand this screen for the use of application configuration to make it 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.

- Change scanning properties like maximum length, time periods for each scan or even make scanning periodic. There could also be different 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.
- Add map tiles uploading for different floors or buildings with the possibility to change between them and download fingerprints based on such location.
- User verifications 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 a single screen since all the other settings are done on mobile application. This screen has two display states and holds device wake lock prevent 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.

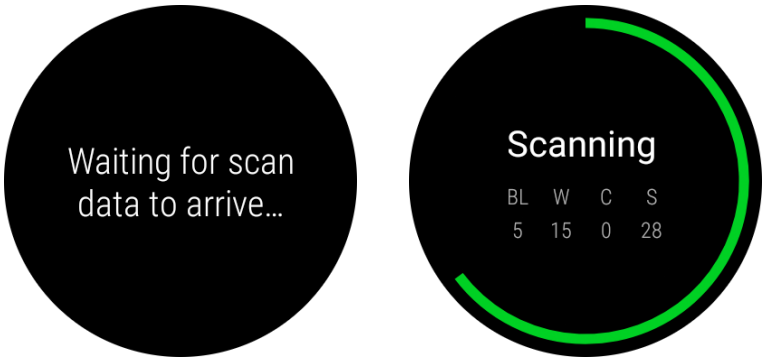

Figure 5.9: Wear scanning screens

First screen shows information message about receiving scan data from mobile, when no data is received for 10 seconds it is closed to reduce battery drain. Second screen displays scan status with a progress bar and count of scanned records. When a scan is completed the informations about it 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 collection of complex code and functions and some interesting parts were selected to describe 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

Since this class handles starting wear application and also sending the data it has Fingerprints saved as a global variable and to be send it must not be empty. Only one change is done to the fingerprint before it is send, which is removing of origin device information to be filled in the wear. When fingerprint is prepared it must be converted into Parcelable DataMap which is an object that is saved into the Data Layer. This data is identified by a String key so application can save multiple objects into this map. Next part of code creates a request using this data and sets its mode to urgent to ensure fast delivery to all devices that listen to this information. Request is completed at this point and can be send via DataClient which can be loaded from the class Wearable wit final line of code clearing the variable containing fingerprint to prevent its sending 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 data change to the 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 uses default classes to contain scanned information. With each class custom data must be parsed from default ones during the scanning and following code, taken from FingerprintScanner class, shows 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 the last time 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 scanner code which leaves few milliseconds where data can be started before scan is actually stared considering WiFi scan cannot be controlled. It could be also solved by not recording this data but since it is only a milliseconds space 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 the application and handles multiple functions and mode thus required a lot of the settings to work properly. It also has many functions which can be configured or disabled based on the implementation required. This map and the code is implemented in MapFragment class.

```
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. Next part is setting up markers and hotspots which will be described later. Calling function frameTo centers the map with a delay because if is done directly it will not work. Final part of this code enables loading when map is panning and disables returning to maximum zoom after double clicking while being zoomed in, be 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

First part of the code sets variables for map markers, such as displaying control window

after clicking, define bounds and anchor point for marker centering in horizontally and vertically. Defining bounds to the markers is very important information because it enables to calculate real pixel location from map absolute one since location on the map changes based on how much the map is zoomed but for scanning the real fingerprint location is needed. This code does not actually display any markers on the map that is done later in the code.

Second part 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, thus effectively enabling to click on any part of the map to display control window for fingerprints.

6 Testing and data analysis

This chapter goal is to show application testing, data collection and analysis.

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 52 m x 43 m area.

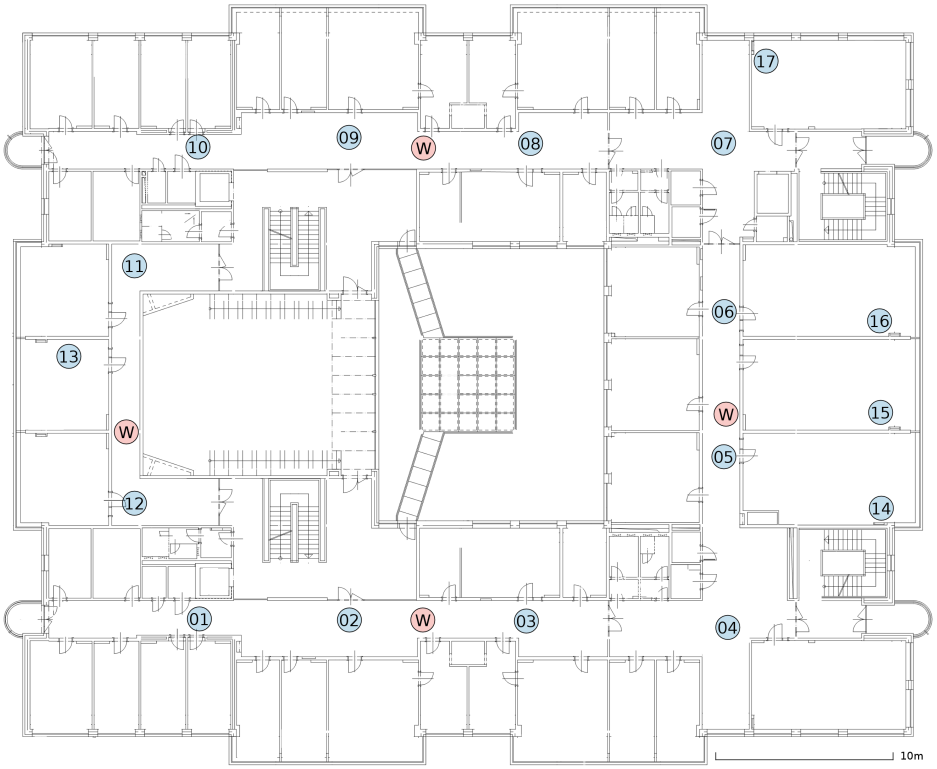

Figure 6.1: Map of deployed devices (based on [9])

Figure 6.1 shows 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. One of them broadcasting in a 2.4 GHz and the other in a 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 on the floor. They broadcast parameters were set to the advertising interval of 100 ms and the TX power of 0 dBm. Corridor beacons are placed in positions about 10 m apart from others [9]. Since the beacons were placed two years ago there was a possibility of battery depletion and malfunction. By multiple tests we figured that two beacons had their battery depleted (1, 4), one was reconfigured and one was missing completely (5).

6.2 Evaluation approach

Basics of evaluation are same as in previous year using solution created by Pavel Kriz in [9]. This approach is using WKNN algorithm to compare fingerprints against each other by selecting one and comparing it to all other fingerprints. Basic premise of this evaluation is to pick one fingerprint, forget its position and try to calculate it via specific amount of neighbors. This calculated value is then compared to real one and difference between these values is the 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 easy implementation and calculation of error difference since both calculated and real positions are known.

This evaluation it implemented to used on one fingerprint and compare it to others, implementing testing of BLE and WiFi signals separate or together 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. Since this approach does not differentiate between technologies of recording devices it had to be expanded to support this.

- Filtering data by the device type, which is an important part to enable test only for a specific device type. These fingerprints are tested only against its own technology and results can be used to compare precisions between technologies.

- Selecting fingerprint group, in this case multiple fingerprints are picked an tested. This selection is done via scan id, meaning both scan originated at the same time and place on different devices. Both fingerprints are tested individually but the results are combined together aiming to improve total localization accuracy. This approach enables two testing types, run the test against its own technology or all fingerprints.
- Combine data from multiple 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.

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

6.2.1 WKNN

This approach is extends 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 [92]. 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 a 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, 92]:

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

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. We can achieve this 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 Campus building floor. 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 in total 540 168.

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 long scans in one run, so the decision was made to split sub-scans into smaller time intervals. BLE beacons have advertising interval set to 100 ms and single scan length could be set to this number but it is set to 200 ms to prevent packet loss between the scans. Since localization using WiFi usually does not need big amount of data, 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 WiFi scanning, only thing that can be canceled is listening for new WiFi records but scan could still be running. The result of this may be that current scan receives data from previous one, this is not prevented in the application but it can be filtered out for evaluation. Cellular and sensor scanner are set to same length as WiFi since this data is used only as complimentary and it helps to reduce fingerprint data size.

Three data collection sessions 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. This session revealed two main problems with the scanner.

First, with WiFi scanning on wear device where the scanner did not find any data. It was probable that there was a problem with 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 WiFi scan right after the activity is displayed and before actual scanning is triggered. These changes helped to improve this scanner and it does not create any problems since there is little time between this “dummy” scan and the actual scan, usual time difference is in milliseconds.

Second, not enough data was received by BLE scanner. Data from previous years showed that each scan usually has hundreds of BLE records but this scanner was able to collect around 50 of them. It was later discovered that 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 it second data collection was run to collect the data for evaluation. And there is one final thing to note, which is that wear device was not able to scan any Cellular records even though specification shows it should support LTE technology.

6.3.2 Second data collection

This collection was meant to be the final one and the data should have been used for evaluation, but there were problems revealed all the data was collected. All data seemed fine but evaluation showed problems with BLE beacons. For a start, analysis data informed that around 30 fingerprints were removed from testing due to no specific BLE beacons found. 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 table of maximum errors in meters for collected data, as it shows BLE errors can be two or three times higher then of WiFi ones reaching almost 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 errors on the map to find the worst locations.

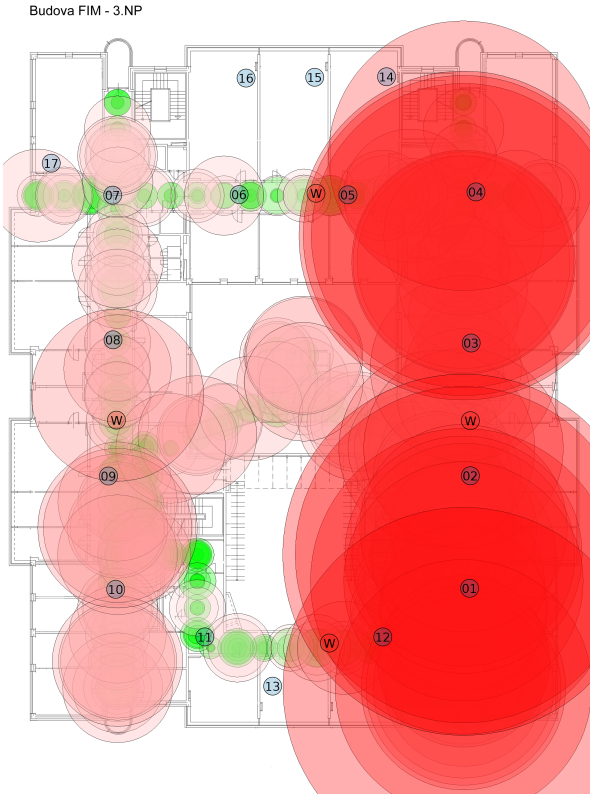

Figure 6.2: Map of errors for BLE in meters for all fingerprints

Figure 6.2 shows errors on BLE scanning on the map with two indications of error size. First, circle size is calculated from error number, the higher error = bigger the circle. Second, fingerprints with error under three meter have green color and those above have color based on their error in red spectrum, more saturated red = higher error. Judging based on these information evaluation seems to be 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. Final tests were to check if all beacons are in place where 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 to further evaluation, hence new data collection was required.

Since this was a complete data collection it also served as a test for wear device how long it can stay operational and scanning. Multiple data were 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 all important information for wear scanning. First thing to note when comparing the percentages of power with time is that wear device cannot usually last for

	% 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)

more than three hours of scanning, which is no enough since all the scans took about 8 hours to complete. First two long scans show that wear can scan for around 35 places without the need for charging, which is confirmed by short scan where scanning 8 places takes around 45 minutes and 15% of the device battery. There are also some exceptions like the third scan which was done in the middle of Campus building where there are now beacons and it was needed to re-scan multiple places many times to achieve set requirements 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 is again with the WiFi scanning on wear where no records are found usually after 10 spots, 80 fingerprints, were collected. This is not a huge issue that has two easy fixes. First, is restarting the device which will enable to scan for 10 more spots without problems. Second and not ideal is to turn off and on the WiFi receiver on the wear device. 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 wear was always turned on and off after 15 successful spots.

For this scan only one specific change was made and that was starting "dummy" scan also for a mobile device, before this collection it was implemented only on wear device. It was changed to provide even more equal environment for both devices and make scanning procedure completely the same. Collecting of all fingerprints took about 7 hours and 15 minutes which is almost an hour less then previous which took 8 hours and 10 minutes. Battery information in Table 6.3 confirm results from previous scans and show that wear

	% 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)

device cannot scan more then three hours at the time before reaching power saving mode. This feature can be disabled after new update of wear device system but it was not used anyway since there is no guarantee it will disable all parts of power saving.

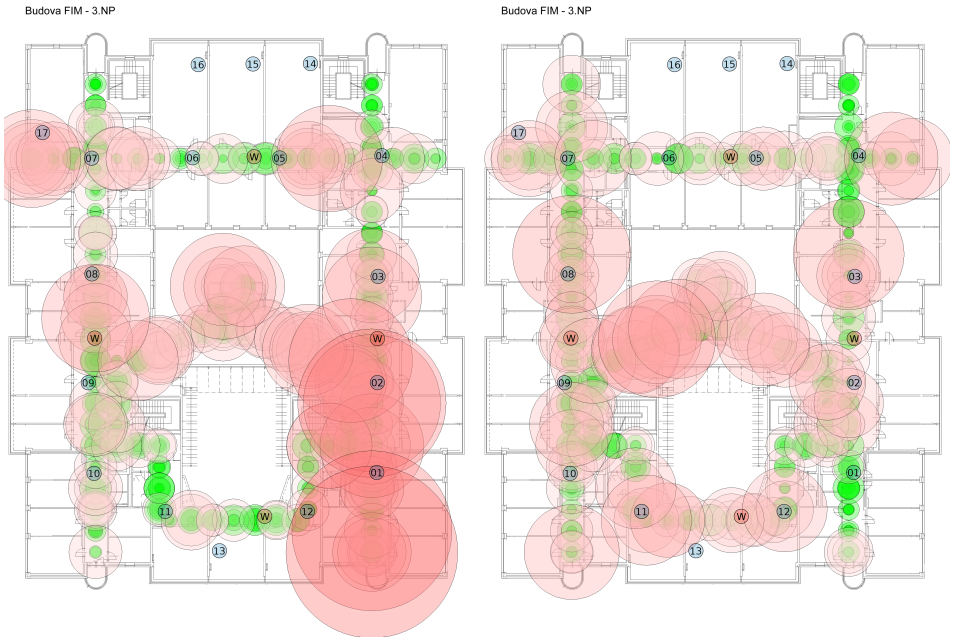

Figure 6.3: Map of errors for BLE (left) and WiFi (right)

Figure 6.2 shows two images of all fingerprint errors on the map thus creating eight overlapping circles on each spot. WiFi seems more precise in comparison with BLE and there might still be some problems around beacon 1. Places with one or two light red circles will usually be improved by other fingerprint on that spot but the others could increase localization error. On the other hand comparison of this figure and Figure 6.1 shows decrease of BLE errors for this scanning round. Comparison between WiFi and BLE suggests that WiFi should have smaller error 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 or multiple device types and radio signal technologies. During data collection was figured out that wear device is not able to scan for Cellular records so this radio signals will not be tested in the evaluation. First thing that needs to be decided is which k variable to use with evaluations using WKNN algorithm and after that multiple tests can be run to compare device types and how data can be used to improve each other.

6.4.1 Decide K values for WKNN

For selecting k value all data was tested without their device information, thus putting all the data together and test them against each other. Only $k \in \{2, 3, 4\}$ were used in final testing because using $k = 1$ defeats the purpose of WKNN algorithm and setting $k > 4$ results with too big errors. Following tangle shows collection 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 main decision variables and maximum is used 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 mean and median are better for $k = 2$ and maximum for $k = 4$. Main variables are better when using $k = 2$ and that is why it was selected for next evaluations. It also achieves better accuracy with WiFi, which is more precise than BLE technology, and this value was already tested and used in previous years [9].

Figure 6.4 shows comparison of both radio technologies used for evaluation and their combination. It correlates with Table 6.4 showing WiFi better than BLE and their combination

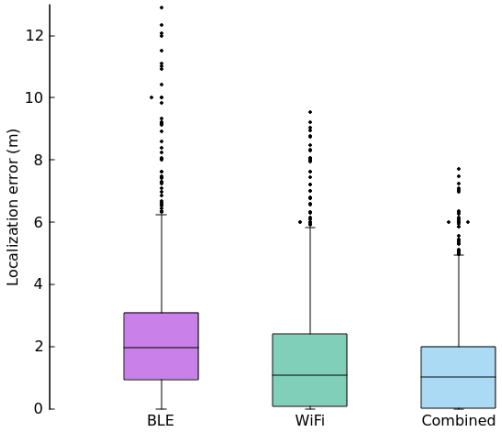

Figure 6.4: Comparison of localization accuracy for $k = 2$

improved than using single of these technologies. This graph is mainly used to compare this evaluation method with future iteration of WKNN algorithm considering device technology recording fingerprints.

6.4.2 Compare device technologies

After it is knows which parameters to use evaluation of device technologies can be started. For this part minimum of data filtering was used and all the fingerprints were tested based on the device technology. To compare all data were also put together and tested against each other without taking device technology into consideration, meaning position of phone fingerprints can 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 in about 0.2 meters which makes them comparable. The WiFi, on the other hand, shows a big difference between these two types making mobile a better solution with over two times lower mean error. Fingerprint data were checked to find precise reason behind this difference.

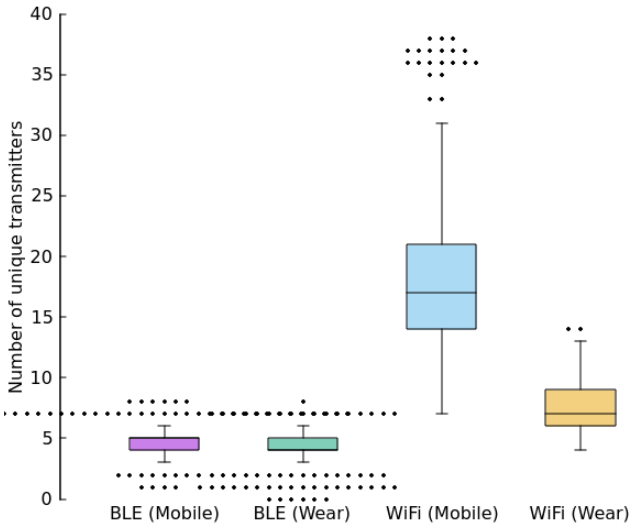

Figure 6.5: Number of transmitters for all fingerprints

Figure 6.5 shows number of transmitters for all fingerprints based on technology and device. It shows close numbers for BLE but there seems to be lower amount of WiFi transmitters for wear device. It seemed that wear was not able to record signals from some transmitters and after comparing data from wear to mobile it was discovered that missing transmitters use 5 GHz technology. Mobile fingerprints contain over 7 300 WiFi records originating from access-point on 5 GHz where wear has 0 of them. Thus figuring out that wear device does not contain 5 GHz WiFi receiver. Every manufacturer is in control of hardware for its devices and Huawei decided not to implement it due to high power usage. 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 to mobile device.

Combining data from both wireless technologies together and comparing devices shows wear error almost three times higher than 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, meaning using only mobile device would result in better precision instead of combining both 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 are displayed in Figure 6.6. The image confirms information that using wear device fingerprints will result in higher error than mobile but there is few places where using wear is better, but unfortunately there is not a lot of them.

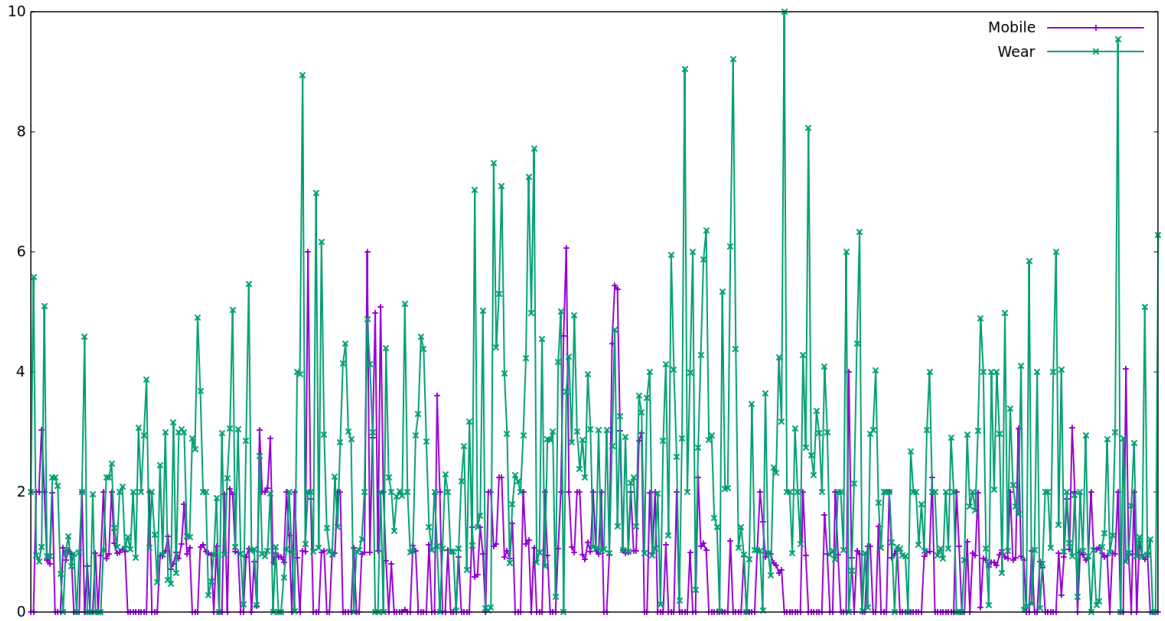

Figure 6.6: Comparison of errors based on device

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 device origins and runs the WKNN on them. These fingerprints have to be from the same location and also from the same scan group, which is identifies 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 implementation of this evaluation. First, compares both fingerprints to all ignoring device technology like previous evaluation. Second, each of the fingerprint is compared only to its own origin device type which is used to check the influence to calculations based on data sources, thus figuring out if testing fingerprints against its own technology can improve or make the precision worse.

Comparing this data with Table 6.4 shows improvement in BLE precision but mean values gotten worse for WiFi and thus making combined error higher from 1.02 to 1.33 meter, which was expected due to results from previous evaluation. On the other hand there is a big decrease of maximum errors for both technologies and decreasing combined from 10 to 5.78 meters. From these values it is not really easy to compare if precision got better on not but it

	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

can be 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.

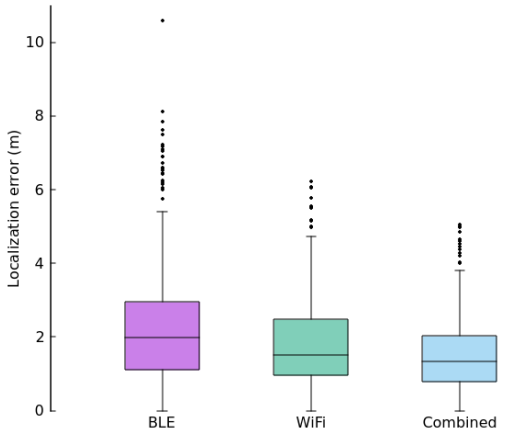

Figure 6.7: Comparison of localization accuracy for testing multiple fingerprints

When comparing this figure with Figure 6.4 its shows decrease of median accuracy mainly for WiFi which also reflects on combined evaluation. Overall accuracy improved because errors are more clustered together and more evenly spread, there is also decrease of maximum error from 10 to 5.78 making it more viable in complex environments.

6.4.4 Combining fingerprint data

Last evaluations did not prove tangible improvement of localization accuracy using wear technology in combination with mobile. For this reason next evaluation takes a different approach in combining data from multiple fingerprints based on device into one. Fingerprints are grouped based scan id, which identifies them being taken at the same place and time using different devices. Data of such fingerprints are combined into single one, reduc-

ing count of data for evaluation but hopefully increasing accuracy. Since there are multiple fingerprints combined the information about origin device is lost and there is no way to compare the technologies against each other.

There are two possible ways this will be implemented. First, copy all data into one fingerprint. Second, copy only BLE data and keep WiFi from phone, because it is known that wear WiFi precision is not sufficient and worsens the 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 big overall improvement in accuracy with a decrease when considering maximum error for BLE. Copying all data together improves precision compared to previous evaluation but it makes it only comparable to using mobile only. 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 of this measurement. Improvement is only in median but other two values increase in comparison.

Second approach of copying only BLE records to mobile fingerprints and keep its current WiFi precision while improving BLE proved to be finally better than only mobile precision. Comparing data from Table 6.7 shows decrease of error for mean from 0.85 to 0.83 meters which is around 2.4% and median from 0.88 to 0.83 meters reaching to 6% improvement over mobile. 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 approaches. BLE errors are 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 multiple fingerprints testing. However, WiFi shows the best localization precision of all previous iterations. Both of these approaches seems to be comparable and each one has different merit to it, combination of all data seems to increase mean and median but lower down the maximum errors being more reliable in problematic places where the other one is the opposite.

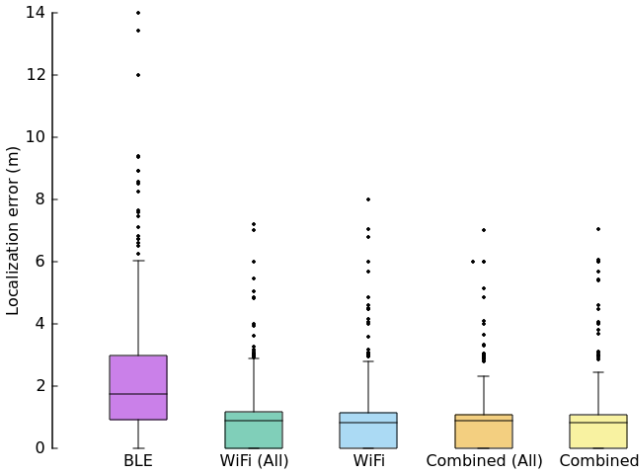

Figure 6.8: Comparison of localization accuracy for combining fingerprints

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 error 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 result data where errors under 200 centimeters are set to display as a circle with such error which is mainly visible in the last image of Figure 6.9 because there are multiple fingerprints with 0 meters error.

Figure 6.9 shows map images for all previously described and evaluated algorithms and image with error only from mobile device. Top left image shows errors for classic evaluation ignoring device of origin. Top right is for next round of evaluation which takes two fingerprints with different device origin, 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 error just from phone device for a comparison with algorithm using data from wear.

Classic evaluation shows a big amount of orange circles, which means that error of a specific fingerprint is over 3 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 places are not considered as problematic since they usually have only one fingerprint with error over 3 meters. There is 119 fingerprints with error higher than three 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 count of fingerprints

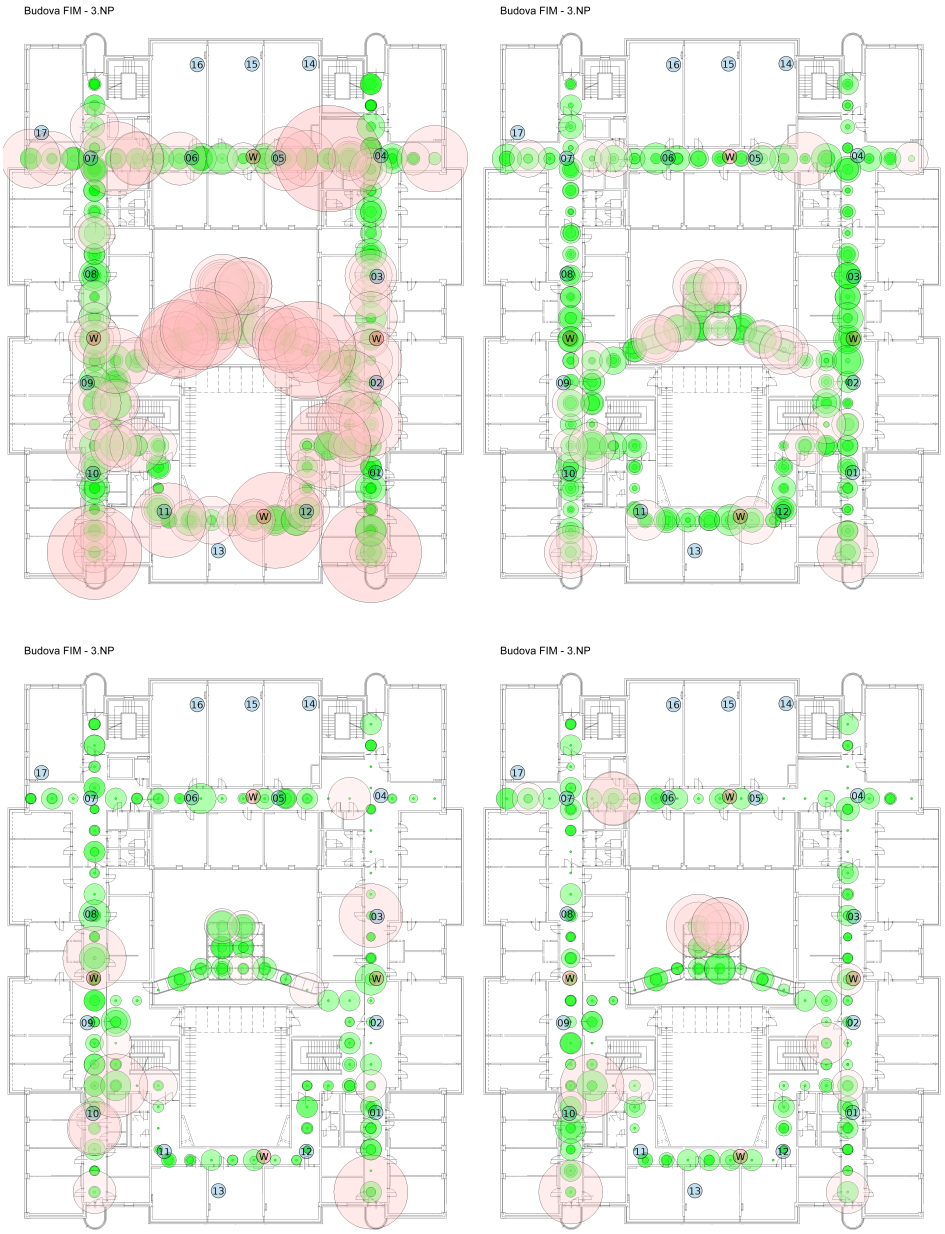

Figure 6.9: Maps of errors for all algorithms with last one for mobile error only

with high error from 119 to 35 which is just 4%. As for final image, it illustrates the best precision of all algorithms with 82 fingerprints with 0 error and just 15 above 3 meters bringing it just under 2%.

7 Conclusion

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

Evaluation was composed of three main algorithm implementations to figure out how to combine data from multiple device types and improve overall localization accuracy. First, testing data from each device type separate 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 networks. This actually makes overall localization less accurate when combining the data together, where mean error increased from 0.85 to 1.52 meters. Second, testing one fingerprint per device type and averaging them improved 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 based on device type together.

Best results were reached when combining fingerprint data from both devices into single one with up to 6% improvement of overall accuracy which should be even higher in locations not using 5 GHz WiFi networks. When this network is used and wear device does not support them, it is better to not include WiFi data from wear in the evaluation.

Wear device improves localization when used with the phone but it is important to keep in mind that at this time wear device type does not possess high battery life and cannot be used to scan for a long period of time. Higher battery life could also mean the possibility for manufacturers to implement 5 GHz WiFi receiver and effectively improve localization

bringing accuracy to the level of mobile device.

7.0.1 Future improvements

This application is meant to be used in different environment 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 specific location. Another likely improvements is to make scanning more viable by changing its settings like length. 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] The Estimote Team Blog. *Reality matters — Preorder for Estimote Beacons available, shipping this summer* [online]. Estimote, Inc., 2013 [cited 2018-01-27]. Available at: http://blog.estimote.com/post/57087851702/preorder-for-estimote-beacons-available-shipping

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

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

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

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

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

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

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

[83] 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

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

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

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

[87] 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

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

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

[90] *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

[91] *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/

[92] 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 with application
    a) Application data
2. Public Github repository with application and thesis text.
   `https://github.com/Del-S/WearNavigation`

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