University of Applied Sciences, Communication Engineering

M A S T E R T H E S I S

Indoor localization using beacons in Active and Assisted Living environment

A master thesis submitted in partial fulfilment of the

requirements for the degree of

Diplomingenieur

Author: Kristian Burfeindt

Matriculation Number: 0810286036

Academic Tutor: FH-Prof. DI Dr. Johannes Oberzaucher

Abstract

One of the major challenges western countries are currently facing is rapid overaging of the society. This results in different economic and social changes. The topic of research in this area is Active and Assisted Living (AAL). It is about assisting elderly people during aging. The subject of this thesis is to apply indoor localization to assist seniors at their homes. The thesis begins with a comprehensive introduction to AAL and the theoretical background of indoor localization principles. Possible error correction methods are discussed to improve the results. The next part will introduce the methods to develop and evaluate an indoor localization system. The thesis is finalized with the results of the implementation and their evaluation and how they may be used in context of AAL. Further development approaches are given.

Key words: AAL, indoor localization, error correction, Estimote beacon, smartwatch

Kurzfassung

Eine der größten Herausforderungen aller westlichen Länder ist die beginnende Überalterung der Bevölkerung. Diese wird zu unterschiedlichen ökonomischen sowie sozialen Veränderungen führen. Dieser Fachbereich ist Teil der Forschung für Altersgerechte Assistenzsysteme für ein selbstbestimmtes Leben (AAL). Das Ziel ist Senioren bei ihrem alltäglichen Leben zu unterstützen, damit sie bis ins hohe Alter selbstständig leben können. Diese Arbeit untersucht die Anwendung von Ortung im Gebäude um Senioren zuhause zu unterstützen. Sie beginnt mit einer umfassenden Einführung in AAL und die theoretischen Grundlagen von Ortung im Gebäude. Mögliche Fehlerkorrektur Methoden werden vorgeschlagen um die Resultate zu verbessern. Der Mittelteil dieser Arbeit definiert die angewandten Methoden um solch ein System zu entwickeln und zu evaluieren. Der Abschluss dieser Arbeit beinhaltet die Resultate der Implementierung und deren Evaluierung im Kontext von AAL. Zum Schluss werden mögliche weitere Entwicklungsschritte vorgeschlagen.

Suchbegriffe: AAL, Indoor-Lokalisierung, Fehlerkorrektur, Estimote beacon, smartwatch

Table of content

Chapter 1

Introduction & Background

This chapter provides an introduction into the general topic of Active and Assisted Living (AAL). The abbreviation AAL was formerly used for the term “Ambient Assisted Living” but was renamed by the Active and Assisted Living Programme. [1]

After the general motivation for this thesis is described, the requirements of the target users will be elaborated. Afterwards the general principles of indoor localization are outlined. The following part deals with possible methods of error correction in context of indoor localization.

At the end of this chapter possible AAL use cases are described in general. Resulting from these use cases leading questions of this thesis complete this chapter.

1.1 Motivation

It is widely accepted and statistically proven that the current population structure of the European Union (EU) will dramatically change in the next years and decades. One of the main statistical findings is that „The main outcome of the current low levels of fertility and mortality in the EU is a progressive ageing of the population.” [2] As a result there will be shifts in the broad age groups. Figure 1.1 illustrates the 2014 population pyramid and the extrapolated population pyramid for 2080 based on the whole population of the EU in percent. Each pyramid is divided into men and women, showing no major differences between the sexes in context of age development.

The main finding is that there is no growth in birth rate to be expected in the EU. This will lead to a continuous shift from the current working population to become elderly without enough children to fill the gap.

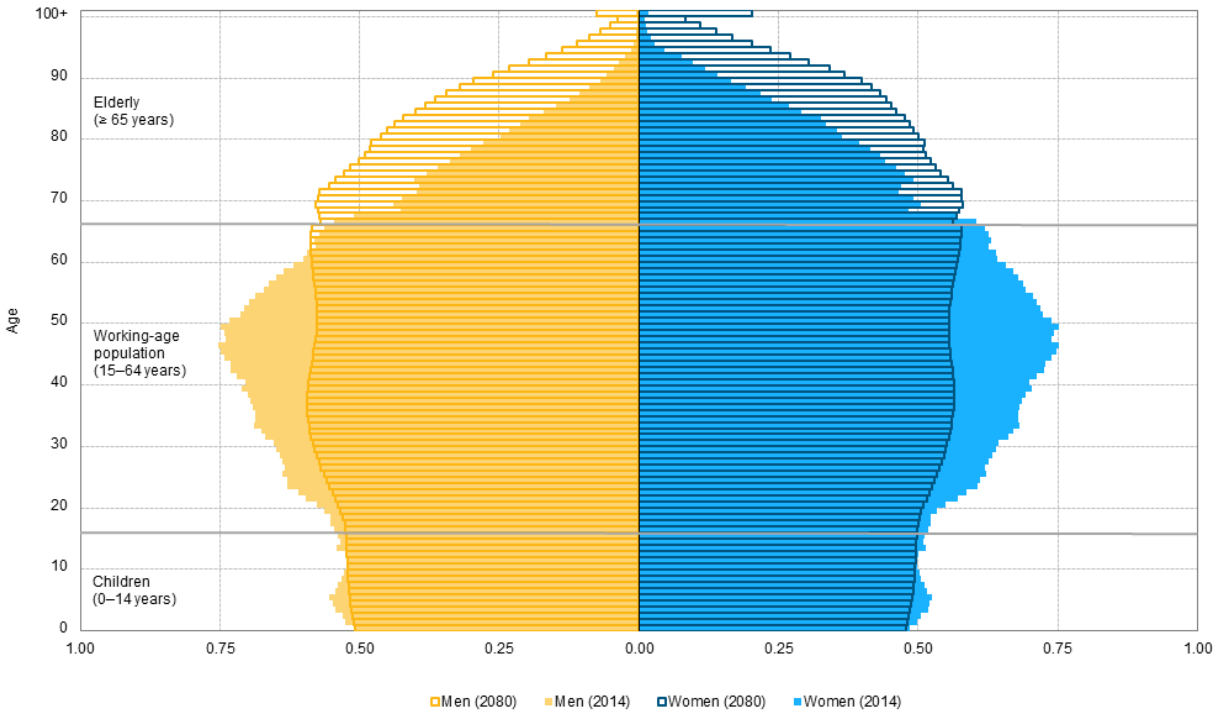

Figure 1.1: Population pyramids for 2014 and 2080 [2]

Current statistics from Germany show that only 10% of elderly persons older than 70 years live without illnesses [3, p. 7]. This leads to the conclusion that in the future more health care will be required in general. Of all seniors which require health care, about two third are living in their own homes. 46% of these elderly are nursed primarily by their relatives. This means that in Germany already 4.5 million persons were main care giver in 2010. In other words they already made up 5% of the population of 81.78 million. [3, p. 29f]

The other third of elderly requires stationary health care which is currently a lot more cost-intensive. That is why political decision-makers of the social systems prefer to allow seniors to live at their own homes as long as possible. [3, p. 34]

This creates a major challenge for the future of European countries. On one hand the number of working persons will continuously drop. On the other hand the demand for home care will increase by the same amount.

A first attempt would be allowing more persons in their working age to care for their own family or work in home care. The development of artificial intelligence (AI) is currently reaching a point where first AIs are being used to replace jobs. For example the Japanese Fukoku Mutual Life Insurance Co. replaced 30% of its staff with an IBM

Watson AI system in January 2017. Basically this AI is analyzing and interpreting customer related data. For example medical certificates issued by doctors or medical histories may be interpreted for payout decisions. In the end employees are still deciding final payouts. In general AIs reduce costs by minimizing work effort that is otherwise done by employees. [4] This approach however cannot be the only solution because only certain jobs can be replaced by AIs.

The approach of this thesis is assisting elderly persons in their familiar surroundings. The main question of assisting elderly in their used environments is how to allow them to stay as long as possible. This topic will be outlined and afterwards it will be described how indoor localization can be used to assist.

1.2 User requirements

This part will describe general requirements of the target users of this thesis. In general the target users for the indoor localization use cases, which will be proposed, are elderly. This does not mean that any group of users is excluded from the proposed solutions. On the contrary it will be shown that by taking the needs of elderly into account all groups of users can use the proposed solutions more easily.

There are different models for describing needs of humans. One commonly used model is Maslow´s hierarchy of needs. He originally described a hierarchy of different needs which build on one another. Later it was drawn as a pyramid of layers with each layer containing needs which belong together (see Figure 1.2). Maslow stated that some needs take precedence over others. Those needs which are most basic such as food, water and so on are located on the bottom of the pyramid. Other needs become more relevant when these most basic needs are fulfilled to a certain degree. The layers do not mean that they need to be fully fulfilled before needs of higher levels become relevant. It must be seen as general classification of needs and how important they are. [5]

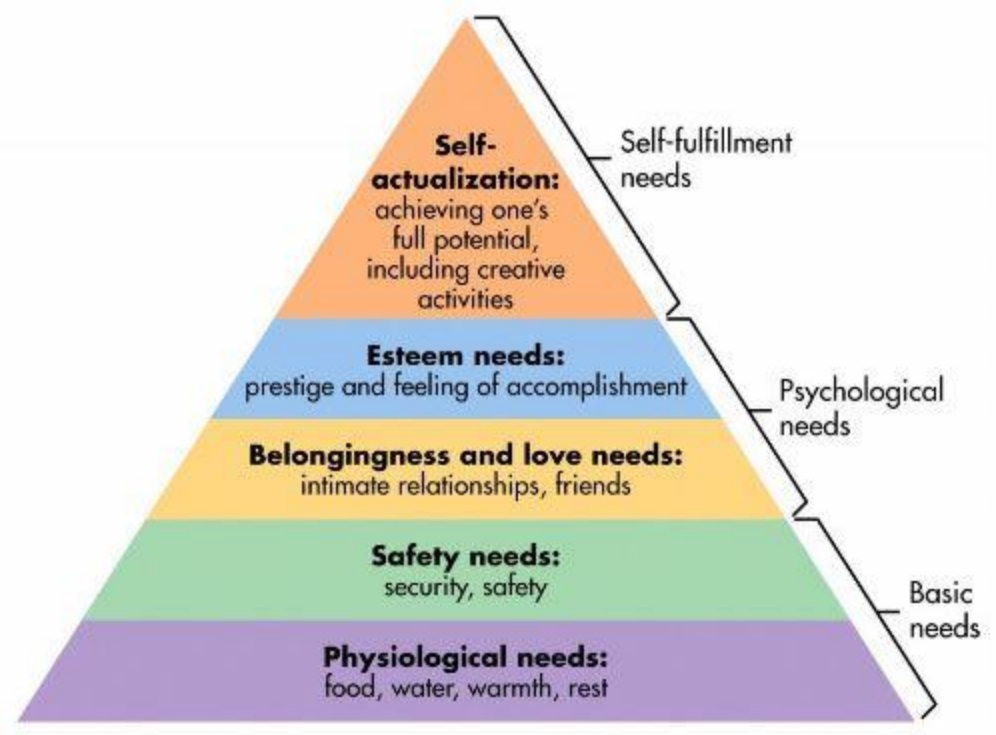

Figure 1.2: Maslow's Hierarchy of Needs [5]

In context of AAL and in specific indoor localization the relevant needs in Maslow´s pyramid of needs are mostly safety needs. Another aspect is communication with relatives and friends. This is inevitably part of applying mobile devices. Hence belongingness and love needs are also part of proposed AAL solutions.

1.2.1 Usability & feedback

In order to create best usability universal design should be applied. This means that the design should not be done for a specific group of users but rather allow any user to use the solution. For example AAL indoor localization applications must not only be focused on assisting elderly persons but also take into account any person which may require help. This is summarized under the term "Design for All". [3, p. 128]

An example for universal design is output. It lays in nature that visual, acoustic and haptic perception change while aging. For example acuity and ability of differentiation between colors and contrasts diminish when getting older. This leads to the conclusion that output on displays should be clear und not too small. The contrast of text output to background color should be maximized. The most important aspect is

that these rules apply for any user. Even children can read text more easily when the contrast of words to background is high. [3, p. 71]

This also applies for feedback to users. Any application should apply as much feedback for actions as possible. For example an alarm should not only be visible on the display but should also give haptic feedback and output sound if possible. On the topic of hearing studies show that the threshold of hearing is increasing as well as the qualitative recognition is changing when getting older. As a result higher frequencies become harder to understand. In addition to that recognition of consonants worsens in comparison to vocals. [3, p. 27]

Another aspect of sound output is that the volume of speakers should be easily controllable. This is often taken for granted but should not be underestimated. [3, p. 75]

Cognitive abilities like concentration and retentive memory are deteriorating during aging. On the other hand logical-analytical intellectual powers are not influenced much by aging. Hence AAL solutions should never require users to remember processes but should be intuitive in order to be accepted by elderly users. Elderly expect technology to adapt to their life. Solutions should never require their users to adapt to them. [3, p. 71f]

1.2.2 Accidents & security

The main challenge for elderly persons to stay in their used environments is security. About 44% of accidents of seniors occur in their homes. Possible reasons for accidents are missing lighting, faulty stair steps, faulty hand-grips in bathtubs or showers or slipping carpets. [3, p. 49]

The other part of security is the fear of elderly to forget turning off electric devices like stoves or irons. They also fear fires started from forgotten candles or faulty cables. One of the greatest fears is to falling and lying on the ground undiscovered. [3, p. 49] This is because of a tendency of elderly persons to live alone in their flats. One of the main reasons for this is the rising divorce rate. Also the tendency of children to live with their parents is dropping [3, p. 13f].

In combination these reasons often lead to moving out of their homes and into care homes. By applying AAL solutions the aim is to aid the elderly in such a way that they feel safe and cared for.

1.2.3 Installation & neutrality

A major aspect of assisting elderly with AAL solutions is seamless integration into their flats and life. For example integration of sensors in flats should not require additional cabling and shall be unobtrusive. Any AAL is supposed to be neutral and never stigmatise its user to be “too old” or even “to be disabled” in any way. [3, p. 75]

If AAL devices run on battery, runtime becomes a major concern. When exchanging batteries is required it should be kept as easy as possible.

Elderly only hardly accept changes to their used environments. Humans tend to feel better in familiar environments. When getting older adapting to changing environments becomes more difficult in general. This is because the fluid intelligence is continuously decreasing after the age of 20. This means that abilities like recognizing context as well as concluding and finding solutions for problems becomes harder. Also time required for information processing rises. [3, p. 72ff]

1.2.4 Robustness

AAL solutions should be robust in a sense that users should not be forced to intervene. As said in 1.2 absent security is a major reason for elderly to leave their flats. In order for AAL solutions to be accepted and useful they need to be stable and trustworthy. It is not acceptable to apply unfinished solutions. [3, p. 75]

1.2.5 Privacy

When AAL solutions are applied it is extremely important to ensure privacy of any collected data. The main focus of this thesis is indoor localization and as such the location of primary users will be monitored. Information about the current location of a person is private to the highest degree. It is absolutely vital that location information is only handled locally. If for example an alarm is triggered by the mobile device only the alarm should be send. Private information should be kept private and thereby local on the user device.

Also any of the collected data must be locally encrypted so that security on the side of software is guaranteed. [3, p. 50], [3, p. 75]

1.3 Indoor localization

Locating mobile devices indoor is a major challenge up until today. Outside of buildings locating mobile devices is possible by use of GPS (Global Positioning System). GPS is very accurate outside of buildings as long as there are no major objects blocking the direct line of sight to the GPS satellites. For example walls of buildings are too thick and will block GPS reception when the receiver is inside. There are alternatives to GPS like the Russian GLONASS, Galileo from the EU or Beidou from China. However on the free market GPS is the de facto standard system for outdoor localization. [6]

All of these systems have in common that they are not sufficient for indoor localization. Different approaches need to be taken.

1.3.1 Indoor positioning technologies

GPS is the de facto standard system which is very common and is used all over the world. Because Indoor localization systems have to be implemented locally in possibly every room there is high design fragmentation. The main reason is that there is no single solution which may be defined as being optimal. Applied technologies differ in their delivered accuracy as well as in cost of implementation. The most common technologies for IPS (Indoor Positioning Systems) are distance measurements to anchor nodes, magnetic positioning and dead reckoning.

Another reason for high fragmentation is the significantly higher amount of measurement errors inside of buildings due to the fact that rooms are not empty space but are filled with objects, persons and so on. Reflection and diffraction from walls and objects cause multi-path propagation. The error budget needs to be reduced in order to get accurate positioning. [7, p. 231ff]

1.3.2 Distance measurement methods

According to [7, p. 235] the most commonly used methods are RSSI (Received Signal Strength Indication), ToA (Time of Arrival) and TDoA (Time Difference of Arrival).

RSSI makes use of the fact that the receiver always knows the strength of a signal that it receives. No additional overhead in communication is required to measure the RSSI. To interpret the RSSI as distance between sender and receiver the path loss model needs to be known. This is the main challenge of RSSI interpretation because as said earlier multi-path propagation comes into play. Resulting ranging errors are reported to be ±50% [7, p. 235f]. For this reason RSSI measurements require error correction methods.

Time of Arrival makes use of the relationship between distance and transmission time. The propagation speed of used waves inside the medium needs to be known. If both receiver and sender know the time when a signal was sent, the travelled distance can be calculated accordingly. This method highly depends on the wavelength of the signal as well as time synchronization between sender and receiver. This means that sound waves can be measured quite accurate while radio waves are hard to measure because of their high frequency. [7, p. 236]

Time difference of arrival is a method of combining measurements to improve accuracy. This is done by using two mediums with different propagation speeds to transmit the same signal. For example a sender transmits an ultrasound wave as well as a radio wave simultaneously. Because propagation speeds of both mediums are known, the time difference of arrival at the receiver may be used to calculate the distance to the sender. [7, p. 236]

Summarizing distance measurement methods the RSSI is quite obvious and very cheap because no extra hardware is required. The trade off is the inaccuracy without error correction. TDoA on the other hand is quite expensive and hard to implement because it requires two synchronized transceivers. The main advantage of TDoA is that it is very accurate with accuracies down to 2cm. [7, p. 236]

1.3.3 Angle measurement methods

Angle measurement is an alternative to directly measuring distances.

> *"Such an angle can either be the angle of a connecting line between an anchor and a position-unaware node to a given reference point (0° north). It can also be the angle between two such connecting lines if no reference direction is commonly known to all nodes." [7, p. 236f]*

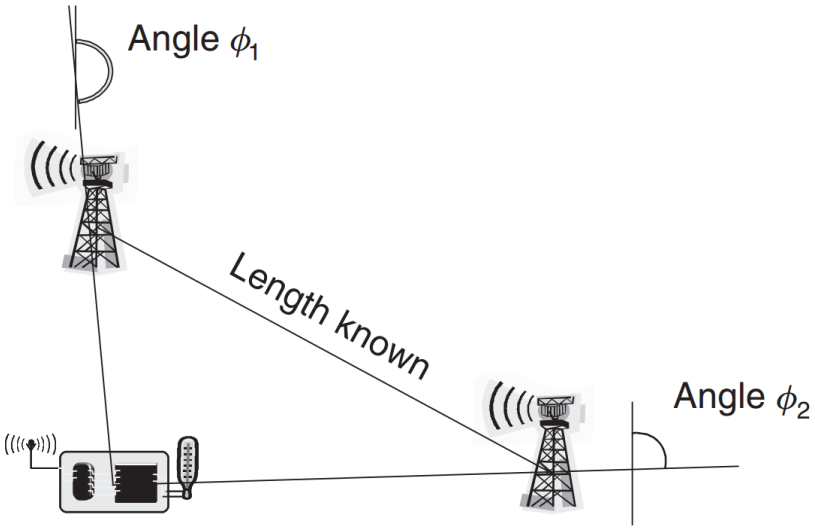

Figure 1.3: Angle measurement [7, p. 237]

The standard approach of measuring angles is using directional antennas rotating like radar (Figure 1.3). This method is simple by concept but is also not really applicable in mobile devices. An alternative approach is to use multiple antennas with known separation between each other. By using multiple antennas it is possible to measure the TDoA between the same signal on different antennas. Now the angle of arrival can be computed by applying the separation to the TDoA. [7, p. 237]

1.3.4 Triangulation & trilateration

Triangulation is an ancient localization principle which has already been used in the 6th century BC by Greek philosopher Thales. The principle of triangulation is to calculate the user location by measuring angles to three noncollinear locations. Noncollinear means locations must not lie on a line. The positions of these locations have to be known in order to calculate the user location. Figure 1.4 illustrates the

different localization principles. In the case of triangulation the known locations are B, C and D. The user location is called A. To triangulate the position of the user, the device measures angles of received signals from all three locations. [7, p. 234]

The other basic principle is trilateration. It also requires three noncolinear known locations. The difference is that instead of measured angles the distances to the known locations are measured. In the two-dimensional case the distances are used to create circles around the locations where the measured distances are the radii. The intersection of those circles is the location of the user (see Figure 1.4). In the real world the locations are three-dimensional. This means that in reality spheres are used instead of circles. [7, p. 234]

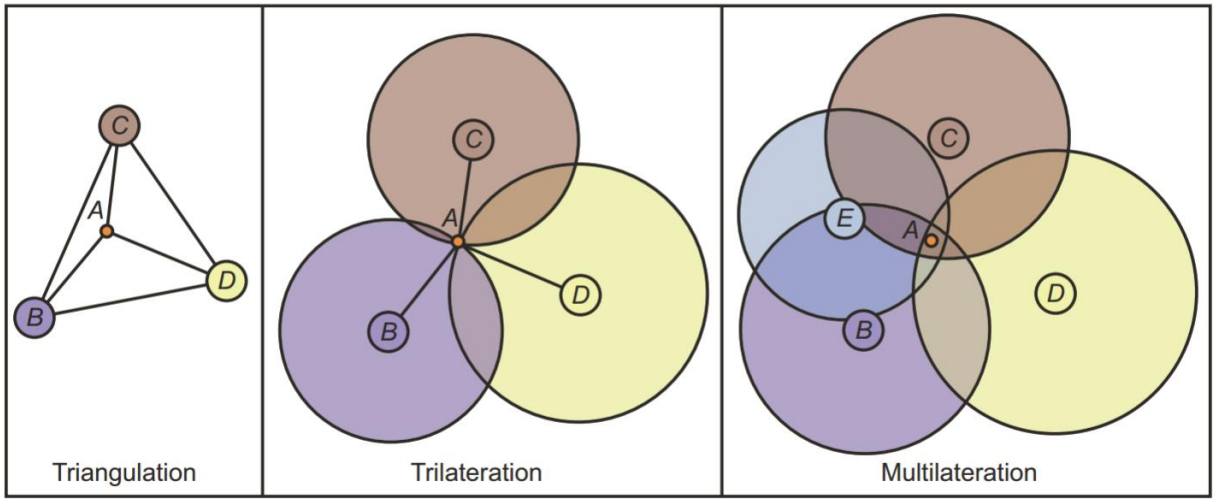

Figure 1.4: Triangulation, trilateration & multilateration [8, p. 113]

In reality measurements of angles or distances are imperfect. In general this will lead to more than a single possible solution or even no solution at all. One way to overcome this is applying more known locations which is mostly done with lateration. The resulting principle is called multilateration. [7, p. 234] Figure 1.4 illustrates multilateration with an extra known location E. Note that in this example the device location A is not directly overlapping with any of the circles from the known locations.

In general also multilateration does not overcome the issue that measurements are imperfect. It helps to further filter for possible solutions. There is no simple solution for finding the user location even with much more locations.

1.4 Error correction - Method of least sqaures

As shown before the main challenge of any type of indoor localization is measurement errors. In principle the problem is “[...] to fit a model to observations subject to errors.” [9, p. xv]. In context of indoor localization the observations are the measured distances to the known locations. Mathematically the problem is to solve an overdetermined system of equations where more measurements than unknown variables exist. [9, p. 1]

Already in the year 1795 Gauss developed the method of least squares. “It can be shown that the solution which minimizes a weighted sum of squares of the residual is optimal in a certain sense.” [9, p. xv] In context of indoor localization a minimum in the model function can be seen as optimal. A minimum is the location within the room where the distances to all known locations is minimized. It is of course possible that there are several or no minima in such a function.

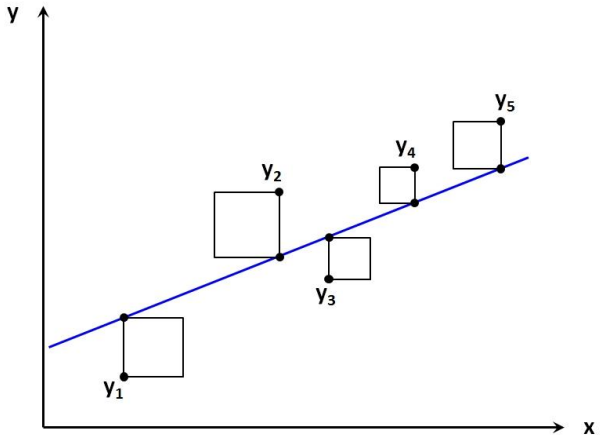

Figure 1.5: Least squares example [10]

Figure 1.5 illustrates this method in a simple two-dimensional example. The measured values are called y_1 to y_5. The residual is the distance of each measured value to the model. The model is a straight in this example and can be seen as optimal by minimizing the squares created from the residuals.

Least squares problems can be linear or nonlinear. If the unknown variables of such a problem are affecting the model linearly the problem is also linear. Hence the example of a straight in Figure 1.5 is linear. The according formula is shown in (1.1).

$$f(x) = \mathrm{a} * \mathrm{x} + \mathrm{b} \tag{1.1}$$

If unknown variables affect the model function in a non-linear way the resulting problem becomes nonlinear. (1.2) is an example of a non-linear model of a growing sine wave.

$$f(x) = \mathrm{a} * \mathrm{x} + \sin(b * x) \tag{1.2}$$

Almost every real world problem is nonlinear. Hence using least squares for error correction the method is nonlinear. Because there are no simple mathematical solutions for nonlinear least squares problems the solutions are iterative. [11, p. xi]

Several different iterative algorithms have been developed. One of the fastest and most reliable is the Levenberg-Marquard algorithm. It converges to a solution in almost any case. The iteration rule of this algorithm is shown in (1.3) [12, p. 12.7]. Its derivation in general is described in [12].

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k \tag{1.3}$$

where
- k … current iteration
- w … weight vector
- J … Jacobian matrix
- μ … always positive, called combination coefficient
- I … identity matrix
- e … error vector

1.5 Localization use cases in AAL

After the principles of indoor localization have been shown the next part will outline possible AAL use cases and how they can be applied. They will be described in general. In the methods chapter test cases for indoor localization will be derived from the following AAL use cases.

1.5.1 Movement monitoring

The first and most relevant application for indoor localization is movement monitoring in context of AAL applications. A mobile device can be used to measure the indoor location of a user and to interpret it based on predefined use of rooms. If a user is measured at around the same location within a storeroom for more than an hour it is possible that the user is in some sort of trouble. In order to save battery power the main measurement interval should be quite low, for example 2-3 times per hour. The main question rising from this assumption is, whether it is possible to locate a mobile device fast enough without running the indoor localization the whole time.

If this is possible this use case could be expanded so that the localization may become more granular when a user is measured in almost the same location for several times in a row assuming the calculated location of the user is ambiguous.

The following question is the accuracy of the localization. If the indoor localization is exact enough to locate a user within one to two square meters, this could be used to make assumptions if the user is in a normal location or may have fallen.

1.5.2 Automatic assistance

Another indoor localization approach is offering automatic assistance in context of the current position of the user. As shown in 1.2.1 it is normal that the short time memory is weakening when becoming elderly. Often persons go to a room and when reaching it they have forgotten the reason why they went there.

The use case of indoor localization can be to recognizing firstly that the user is moving to a location and secondly when the user reached it. If the user now is standing still for a certain amount of time a mobile client could offer assistance based

on the room or even on the location of the user in the room. Options based on the room would be for example for the bathroom:

- Toileting
- Bathing
- Do the laundry

An example for the explicit location is when a person is going from the living room to the kitchen and is now standing in front of the refrigerator for a certain time. Possible reasons for moving to the refrigerator are:

- Getting a cold drink
- Getting a snack
- Getting ingredients for cooking

This approach would require a high measurement interval. It would be required to recognize exactly when the user is moving to a certain room and how long the user is at a certain location. Another aspect is that for the exact case also very high accuracy would be required to recognize a device the user is standing in front of. Any accuracy above one square meter would be too inaccurate.

1.5.3 Activity monitoring in combination with sensors

Activity monitoring is a possible expansion of the thesis “Implementation of a Non-Invasive Sensor Network for ADL/iADL Recognition in the Context of Active and Assisted Living” by Johanna Plattner [13]. The Activities of Daily Living (ADL) are the activities which any person needs to be able to execute every day. They are cited from [14, p. 915] as follows:

- Bathing
- Dressing
- Going to toilet
- Transfer (moving in and out bed)
- Continence
- Feeding (eating without help)

There are also instrumented Activities of Daily Living (iADL) which are not fundamentally necessary in order to be functioning. However they are required to be able to live independently. They are cited from [15]:

- Ability to use the telephone
- Shopping
- Food preparation
- Housekeeping
- Doing laundry
- Mode of transportation (using public transportation or own car)
- Responsibility for own Medications
- Ability to handle Finances

Of all these activities four were chosen to be recognized by a sensor network. The chosen activities were going to toilet (ADL), sleeping (ADL), food preparation (iADL) and leisure activities (iADL). Sleeping and leisure activities were not cited directly from the according sources but are defined in general nursing literature. [13, p. 18f]

The result of her thesis was applying a non-invasive sensor network to recognize these activities is possible in general. However it should be seen as a first step and further stability optimization is required. One of the proposed methods is adding more sensors to acquire more data which may be associated with unique activities.

Adding indoor localization to such a sensory network can possibly deliver more data or even allow distinguishing activities further. For example when there are other persons visiting it can be possible to distinguish between activities of the owner and visitor. The owner would wear a mobile device for indoor localization while obviously a visitor would normally not. When a sensor now recognizes an activity the additional indoor localization allows recognizing if the mobile device is in the vicinity of the sensor.

The derived requirements for this use case are similar to 1.5.2. In order to recognize the location of the owner when a sensor found an activity, the localization must be quite fast. Where this use case differs from 1.5.2 is the required accuracy. It is not as important to know the exact position in activity monitoring. It would be sufficient to

know if the mobile device is in the vicinity of a sensor or not. The required accuracy therefore matches 1.5.1 with about one to two square meters.

1.6 Leading questions

The indoor localization use cases explained in 1.5 have in common that they are all dependent on battery runtime as well as accuracy of indoor localization. The following leading questions were derived for this thesis:

- Can the accuracy of indoor localization be improved with least squares error correction?
- What is the distribution of the results? E.g. does the distribution suggest that taking the mean of several results improves accuracy?
- What is the impact on battery runtime when running indoor localization?
- Is it possible to use the results for AAL use cases?

Chapter 2

Methods

The methods chapter is divided into two parts named implementation and evaluation. The implementation part describes aspects which are required in order to implement an indoor localization AAL solution. At first the devices and afterwards conceptional design aspects for the apps are discussed.

The evaluation part describes how the indoor localization will be tested in context of the AAL use cases described before. It therefore consists of design of different tests as well as mathematical principles to interpret those tests.

2.1 Implementation

The implementation is structurally divided into two applications. The first app Estimote locate is focused on indoor localization. It is targeted for use with normal sized phones. The first use case is defining rooms and beacons inside the rooms. The other use case is to test indoor localization within those rooms.

The other app is a progression from the Confidence project and is done in cooperation with the company ilogs mobile software GmbH. It will be used on smartwatches and will be adapted accordingly. The main use case of this app is an alarming system. Indoor localization can be used as a separate module using only the localization part of Estimote locate.

2.1.1 Devices for indoor localization

There are two types of devices required for indoor localization. The first type is the beacons which are placed inside of rooms. The second aspect are the mobile devices. It will be discussed why smartwatches are a fitting choice for indoor localization in AAL context.

2.1.1.1 Estimote beacons

The FH Kärnten already made study projects about indoor localization of mobile devices. The approach was to utilize BLE (Bluetooth Low Energy) beacons. There are several different companies providing different beacons. The FH chose Estimote beacons for projects in the past. These beacons were provided for this thesis. There are two different kinds of beacons available from Estimote which are relevant for indoor localization. The first type is called Estimote proximity beacon shown in Figure 2.1 (a). The second type is called Estimote location beacon shown in Figure 2.1 (b).

Differences between these types of beacons are battery runtime, possible range and number of packets. In general the idea of proximity beacons is recognizing which of the measured beacons is closer. This information can for example be used in an app highlighting products in a store which are in the vicinity of a mobile device. Their approximated lifetime is two years on a single battery and they can be used at a range of up to 70m. In contrast the Estimote location beacons are focused on more exact localization. They can be ranged up to 200m and their battery lasts about 5 years. Another practical aspect of location beacons is that they can be recharged with an USB cable. This is not the case for proximity beacons that have to be cut open to replace their battery. [16]

(a) Estimote Proximity Beacons (b) Estimote Location Beacons

Figure 2.1: Estimote Proximity and Location beacons [17]

The basic functionality of beacons is that they broadcast their IDs in definable intervals. Monitoring devices do not need any authorization or coupling with beacons

to read this broadcast information. When a monitoring device recognizes a beacon it thereby simultaneously acquires the RSSI of that beacon.

The generally good battery runtime allows placing Estimote beacons anywhere within rooms without additional cabling. Another aspect of installation is Estimote beacons coming with a backside which sticks to almost any surface which must not even to straight.

Combining these properties it is possible to create an indoor positioning system using trilateration (see 1.3.4) based on distance measurements (see 1.3.2). The Estimote developers documentation states that:

> *„While received signal strength, proximity zone and accuracy values can theoretically be used to derive a distance estimation, in practice this is far from trivial and requires complex mathematical models to account for fluctuations in the signal strength.*
>
> *Long story short: do not expect distance estimations from beacons*.“ *[18]*

This statement matches the ±50% ranging error of RSSI based systems described in 1.3.2. The previous study project, which tested this type of indoor localization, was provided by the FH Kärnten. It is named *iLocate* and will be analyzed in this context before any new approach is implemented.

Estimote beacons provide two different modes of operation called background monitoring and ranging. Background monitoring basically operates as a geofence with “exit” and “enter” transitions as described in [18]. This of course is not sufficient for exact indoor localization. Ranging is the alternative which shall be analyzed in this thesis. The following states the tradeoffs of ranging:

> *„Ranging provides more granular and comprehensive beacon data, but this comes at the expense of draining the battery faster than monitoring. This means that it’s usually not a good idea to run ranging for extended periods of time, e.g., hours. It certainly wouldn’t be viable to run it at all times*.“ *[18]*

This is directly referring to one of the leading questions of this thesis. The impact of ranging for longer periods of time will be tested.

2.1.1.2 Android Smartwatches

In context of installation and neutrality (see 1.2.3) the most apparent question is how to excite elderly users to carry mobile devices with them at all time. As stated in 1.2.3 no AAL system should ever require its users to adapt to the system. Forcing users to carry their mobile phone all day long would certainly be a bad example. As mobile devices are definitely required for the types of proposed indoor localization, another approach would be using smartwatches. As Android based smartwatches are now widely available on the market, the FH is highly interested in utilizing them for indoor localization.

The most relevant part is that Android Wear is dependent on host devices such as mobile phones. [19] Because smartphones shall not be forced to be carried at all times by users, Android Wear is no real alternative. However there are some companies like simvalley MOBILE producing smartwatches which run full Android operating systems and thereby are independet from any other mobile devices. [20]

Figure 2.2 shows two examples of simvalley smartwatches in different styles. The internal differences of simvalley smartwatches are minimal. Their most relevant features are: [20]

- 1.5'' IPS screen
- WiFi, Bluetooth 4.0 & GPS
- SIM slot
- Android 4.2.2
- Dual core Cortex A7 processor
- 3 megapixel camera

The only major differences are design and used materials. For example the GW-420 in Figure 2.2 (b) is build witth real gold whereas the AW-414.GO in Figure 2.2 (a) is build with aluminum.

(a) simvalley AW-414.GO (b) simvalley GW-420 Gold-Edition

Figure 2.2: simvalley AW-414.GO and GW-420 smartwatches [20]

The FH already made several study projects in cooperation with the company ilogs mobile software GmbH. One was the confidence project. [21] The aim was to apply mobile phones to aid elderly persons with dementia. Main use cases were a SOS function as well as a reminder service. The complete implementation of this system is described by Pierre Schaschl in his master thesis “Implementing an Ambient Assisted Living System Utilizing the Android Platform”. [22] Both the FH and ilogs are interested in expanding this implementation to run on smartwatches. Because of this several different simvalley MOBILE smartwatches running Android were provided by ilogs for this thesis.

The optimization for smartwatches will be a minor part in thesis while the main focus will be led on the described indoor localization use cases.

2.1.2 Concept of storage - greenDAO

One of the main questions of this thesis is the impact on battery runtime when running different indoor localization approaches. This means most test cases will run through the whole battery life time from 100% to 0% until the device is shutting down. For the most demanding test there will be one test result each second. Estimating the impact on battery runtime optimistically a runtime of 16 hours is expected for this test. This means that an estimated number of 57.600 results is to be expected. This amount of results is of course not intended for productive use. However a saving structure for this is needed nevertheless.

Pushing each result to a webservice could be a solution. However it is certainly not matching the criteria, that highly private data like the exact user position should not be transmitted for no reason. It should be kept local and private as said in 1.2.5. Only in cases were a solution derives conclusions such as if a user may have fallen, emergencies can be triggered online. Concluding from this the only realistic solution for saving indoor localization results is using a local database for storage and interpretation.

There are many ORM (Object-Relational Mapping) solutions available for Android. At the time of this thesis the best ORM for Android was arguably the greenDAO.

Figure 2.3: greenDAO concept [23]

In general ORM solutions like the greenDAO offer a concept of automatically mapping java objects to a database as shown in Figure 2.3. Java objects are simply annotated as "*@Entity*" and are thereby defined as database objects. The library then automatically generated accessor DAO classes and database tables in background for every annotated object at the next *make* of a project. The only other thing a developer is required to do is to define the name of the database and to open it afterwards. [23]

The reason why the greenDAO is currently accepted as the best ORM solution for Android is because it does not have any drawbacks [23]:

- It is easy to implement with a single Gradle import
- The imported library is only adding ~100KB to an app
- It is very secure and completely open source
- The database can be automatically encrypted
- Memory consumption is minimal
- Performance of data access is best for all known Android ORM solutions

Especially the performance aspect is highly relevant for this thesis. When data is stored fast and does only take minimal time to load, it can be derived that computing time for these actions is also minimal. Hence this indicates that the impact of data storage with greenDAO has only minimal impact on battery consumption.

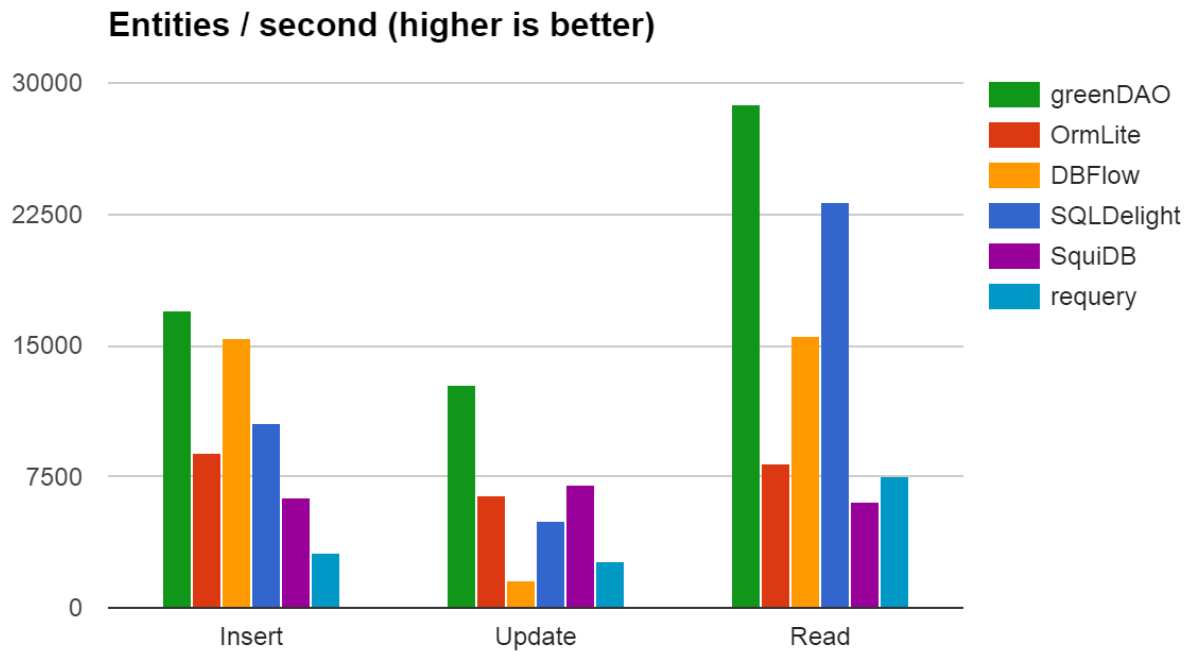

Figure 2.4: Android ORM performance comparison [24]

The developers of greenDAO offer open source performance tests for most well known Android ORM solutions. The results of these tests are shown in Figure 2.4.

Because of this the greenDAO will be used for storage of test results. Also all objects like rooms and the beacons in the rooms will be saved as greenDAO entities.

2.1.3 Design of rooms

All of the AAL use cases described in 1.5 are located at home of individual end users. As said in 1.2.3 the installation of AAL devices must be easy. This applies to mounting to the wall as well as virtually entering the flat rooms into an application.

The design goal of Estimote beacons is that a fixed site is equipped with beacons. For example a store may have a single beacon per product group. This allows to automatically show advertisement on mobile devices of customers nearing a beacon. The Estimote SDK (Software Development Kit) was implemented accordingly. Each beacon is explicitly mapped to a development account and its settings may only be changed by the user of this account. [25] However ranging beacons is possible without connection or even owning the beacons.

For the use cases discussed in this thesis such a design creates several challenges. It means installation in a flat would require that each beacon kit must be matched to a development account. This would create massive overhead during installation of beacons. For this reason an own concept for recognizing beacons was implemented without the need to explicitly map a development account.

This decision on the other hand has to trade off some functionality of the beacons. For example if beacons are mapped to an account their color may be directly queried. As said before the beacon settings may only be changed by the owner of the beacon. For example distribution power as well as broadcast interval are fixed because of this.

The design of the class *Room* with its beacons is shown in Figure 2.5. The classes *Room*, *Room2EstimoteBeacon* and *EstimoteBeacon* are greenDAO entities and are therefore saved to the database. The general design target was to also allow beacons from different vendors to be mapped to a room. The abstract class *Room2Beacon* may be implemented by any other class and therefore allows any type of beacon to be mapped to a room. Any class implementing Room2Beacon must define the coordinates of the beacon as well as its color. The measured accuracy is queried at runtime during ranging and is not persisted.

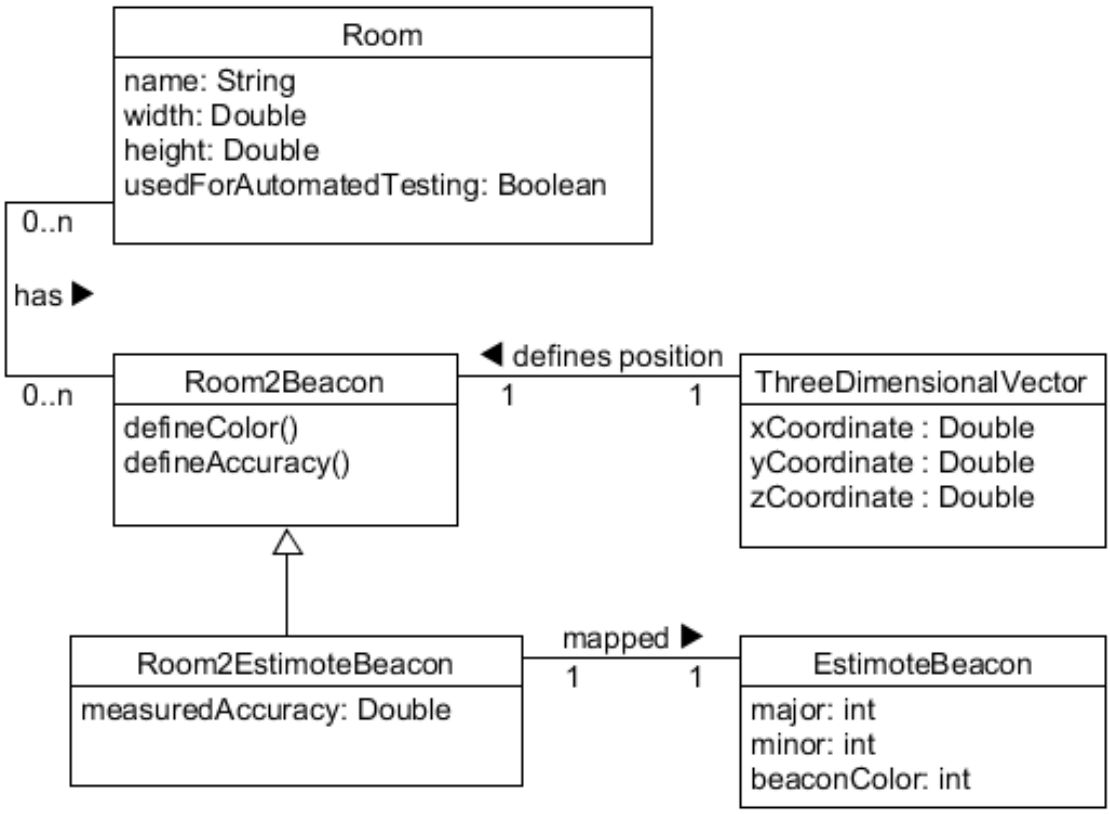

Figure 2.5: Room class diagram [26]

2.2 Evaluation

This section is about the design of specific test cases for indoor localization and the mathematical principles that will be used for evaluation of test results.

2.2.1 Design of test cases

Three basic test cases are derived from the AAL use cases defined in 1.5. They are designed to answer the leading questions of this thesis.

All of these tests will be run with two different devices for later evaluation. The testing devices are the simvalley AW-414.GO shown in Figure 2.2(a) and a Samsung Galaxy A3 2015.

The purpose of using a Galaxy A3 is comparing the results of the smartwatch with a well known device. The question at hand is if using a smartwatch has any impact on bluetooth reception and therefore accuracy of localization.

2.2.1.1 Battery runtime

The first test is about measuring the battery runtime of mobile devices. The battery state is measured with all different tests. This explicit test is used to find the maximum battery runtime and to compare it later. It is designed as sequence diagram in Figure 2.6.

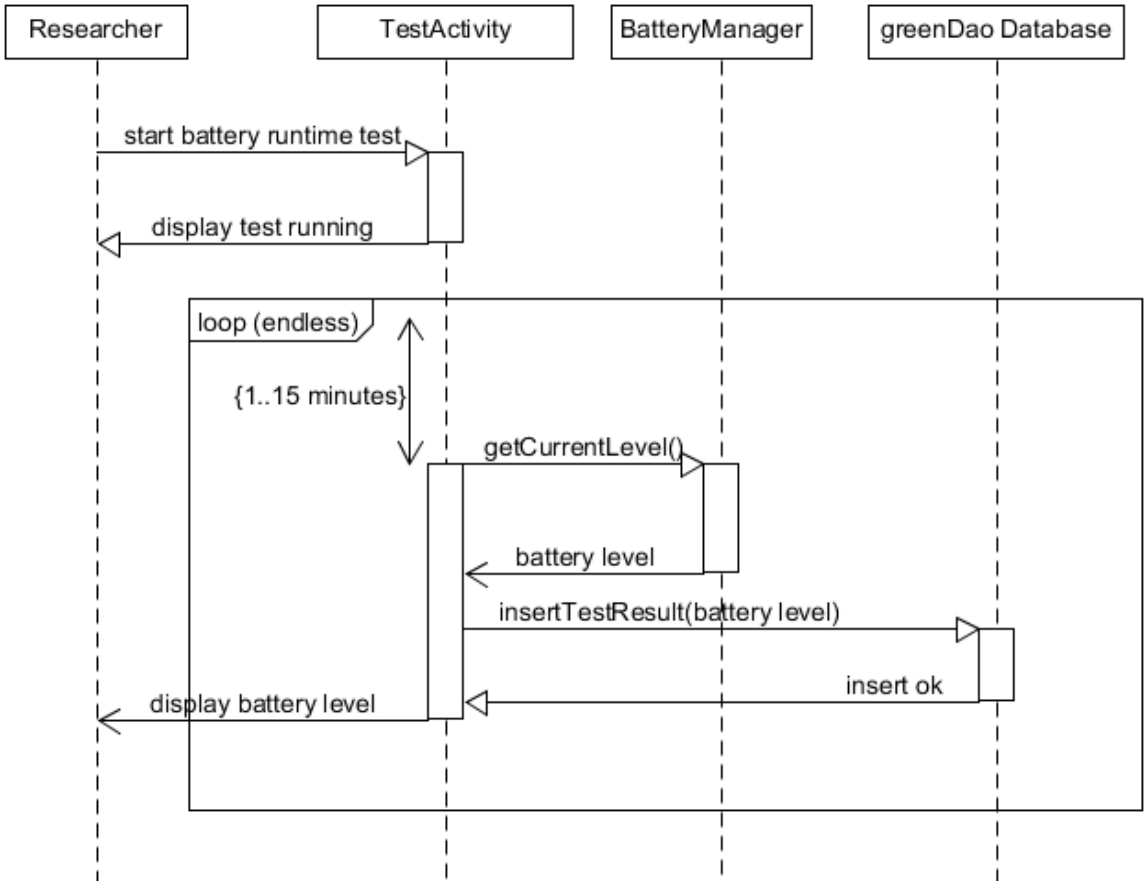

Figure 2.6: Battery runtime sequence diagram [26]

2.2.1.2 Accuracy test

The most relevant question in this thesis is the accuracy of indoor localization. The question is if error correction can successfully be used to improve the resulting accuracy. This Test will be run multiple times for multiple locations and different dimensions. Dimension refers to differentiation between two- and three-dimensional tests.

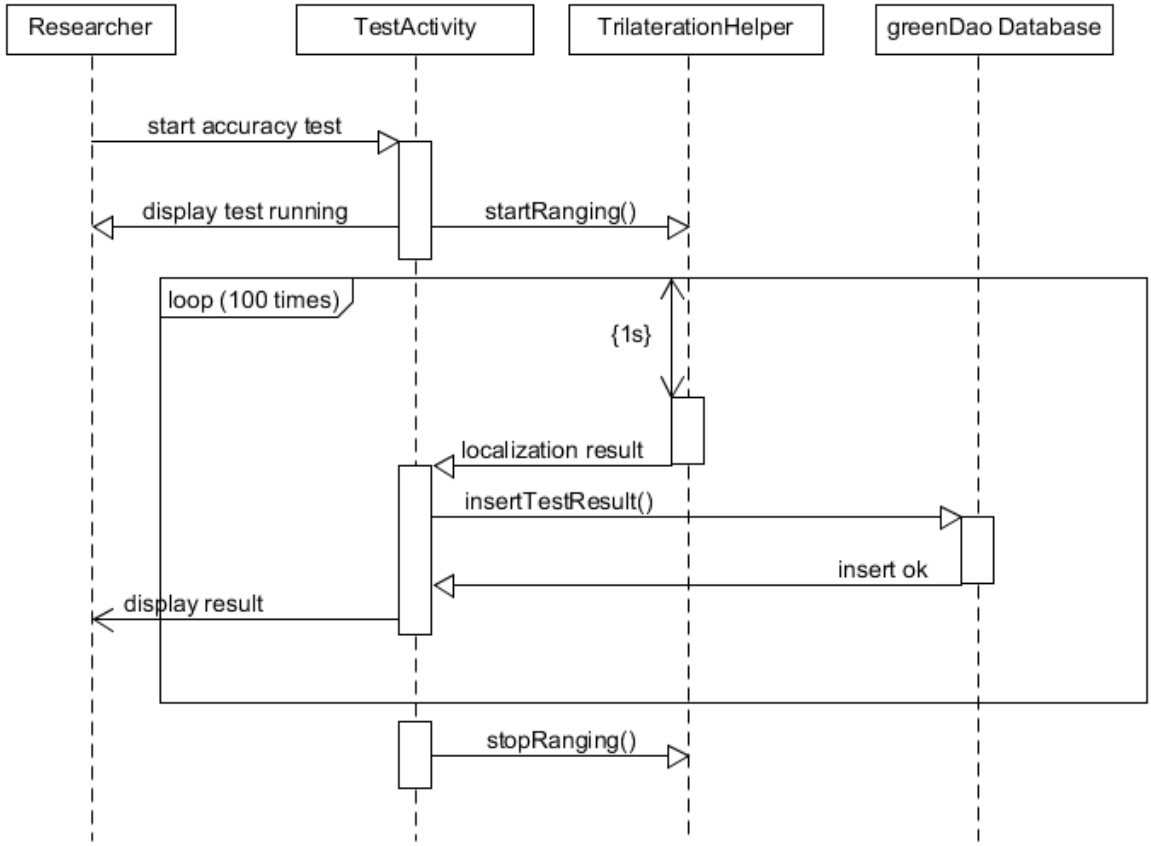

Figure 2.7: Accuracy test sequence diagram [26]

2.2.1.3 Movement monitoring

Movement monitoring is the test for actual usage of indoor localization. It triggers background ranging every 15 minutes and saves the first 5 trilaterated locations. The test case is a combination of testing accuracy over longer periods of time and the impact on battery runtime by doing so. The according sequence diagram is shown in Figure 2.8.

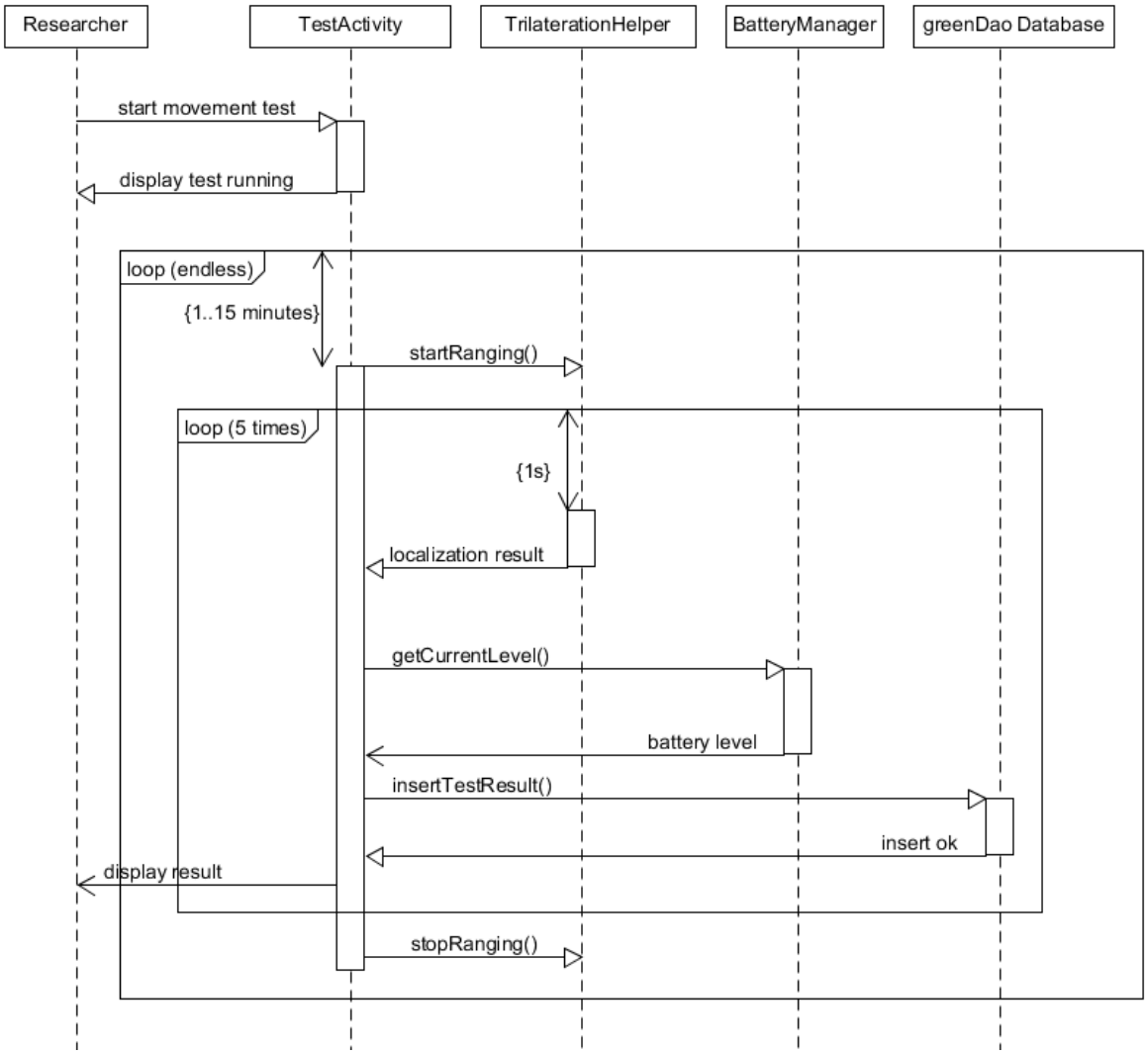

Figure 2.8: Movement monitoring sequence diagram [26]

Another test was designed with almost the same sequence diagram. The test is called “always ranging” and it tests the impact on battery runtime of keeping ranging on at all time. The only difference to the sequence diagram in Figure 2.8 is that the endless loop is removed and the inner loop becomes endless instead of running only five times.

2.2.2 Device settings for testing

Because battery runtime of mobile devices is currently a topic of research there is no standard approach.

To get results which can be reproduced easily mobile perception was deactivated for all devices. The reason is high fluctuation of mobile perception due to scattering,

reflection and so on of radio waves (see 1.3.2). Resulting fluctuations of these phenomena grow when transmitting stations are far away. For this reason only local localization was activated for testing. For real world use cases where mobile perception is good the results should be quite accurate with only few percent deviations. However in rural areas the mobile perception is usually not as good as in cities. This may result in higher battery drain due to mobile communication.

The following settings were made on all devices:

- Mobile Perception deactivated
- WLAN active
- High precision localization active
- Bluetooth active

It is obvious that because of these settings measured battery runtime will not exactly match the real battery runtime. As conclusion it must be noted that the results of battery runtime tests may not be compared with other studies or tests. They can only be used for comparison of the measurement results of this thesis.

2.2.3 Testing flat

It is possible to use a fully equipped flat which was designed to test AAL use cases at the FH Kärnten. However the laboratory has open walls to allow observation during testing. Indoor localization is massively influenced by the size and shape of rooms. Especially walls are relevant because of reflection of radio waves. Because of this a private flat was preferred to produce real world data instead.

Figure 2.9 shows a schematic of the flat used for testing. The living and dining room as well as the hallway are used for testing.

The beacon locations inside the hallway are (in meters):

Left blue beacon $\begin{pmatrix} 0 \\ 0.63 \\ 1.7 \end{pmatrix}$ Green beacon $\begin{pmatrix} 1.3 \\ 0 \\ 1 \end{pmatrix}$

Purple beacon $\begin{pmatrix} 1.56 \\ 1.27 \\ 1.66 \end{pmatrix}$ Right blue beacon $\begin{pmatrix} 2.73 \\ 0.55 \\ 1.62 \end{pmatrix}$

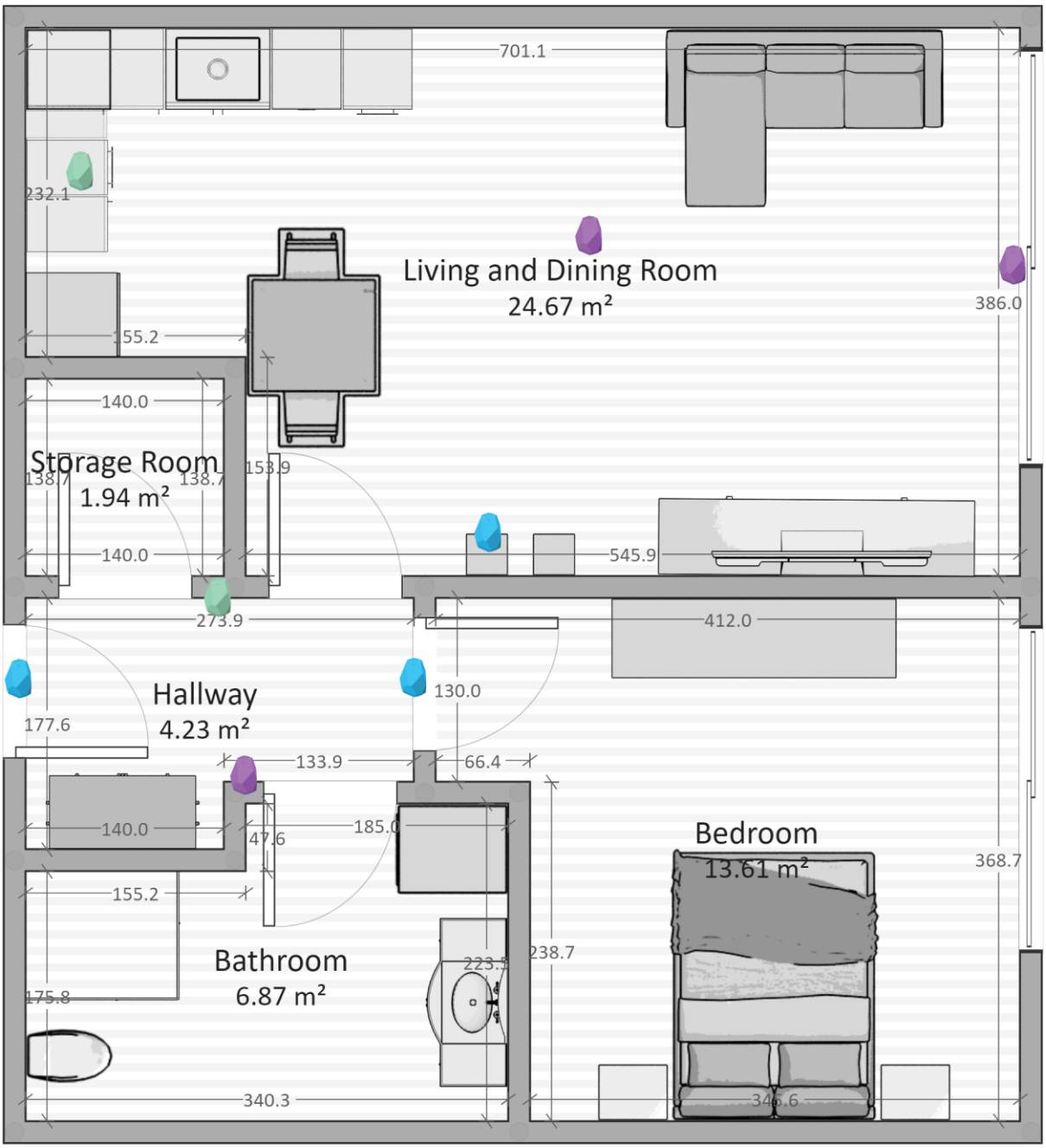

Figure 2.9: Schematic layout of flat including beacons [27]

The beacon locations inside the living and dining room are (in meters):

Green beacon $\begin{pmatrix} 0.36 \\ 1 \\ 1.7 \end{pmatrix}$ Centre purple beacon $\begin{pmatrix} 4.4 \\ 1.7 \\ 2.6 \end{pmatrix}$

Blue beacon $\begin{pmatrix} 3.13 \\ 3.5 \\ 1.63 \end{pmatrix}$ Right purple beacon $\begin{pmatrix} 7.04 \\ 1.5 \\ 1.6 \end{pmatrix}$

A rendered picture of the living room is shown in Figure 2.10. The viewpoint is from standing in the kitchen.

Figure 2.10: Rendered living room with beacons [27]

2.2.4 Explicit location test cases

When testing accuracy of RSSI based indoor localization many non linear parameters influence the result as discussed in 1.3.2. It is important to reduce the amount of them as much as possible to produce results which are reproducible. For all test cases designed in 2.1.1 the devices will be left on a surface without touching them during the test. Three locations were chosen to evaluate the results:

Case 1: Cooking $\begin{pmatrix} 2 \\ 0.5 \\ 0.94 \end{pmatrix}$

Case 2: Lying in front of television $\begin{pmatrix} 6.6 \\ 3 \\ 0 \end{pmatrix}$

Case 3: Lying in hallway $\begin{pmatrix} 0.5 \\ 0.5 \\ 0 \end{pmatrix}$

2.2.5 Variance

The population variance measures how far a set of numbers are spread around their mean (2.1). To calculate the population variance the probability function and thereby the probability of each measured value needs to be known. [28]

$$s^2 = \sum_{i=1}^{N} P(x_i)(x_i - \mu)^2 \tag{2.1}$$

where $P(x_i)$ … probability of x_i
x_i ... measured value
μ ... population mean

When the probability of measured values is unknown the sample variance is used to find an estimate of the population variance. The formula for calculating the sample variance is shown in (2.2). [28]

$$s_N^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{2.2}$$

where $\bar{x}$ ... sample mean

Another important aspect is that the sample variance is not an unbiased estimator for the population variance. The “bias-corrected sample variance“ is often used instead. It is shown in (2.3). [28]

$$s_{N-1}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{2.3}$$

2.2.6 Standard deviation

The standard deviation is the square root of the variance. (2.4) shows the standard deviation for the “bias-corrected“ sample variance. In general it is used as a measure of fluctuation around the mean value. [29]

$$\sigma_{N-1} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{2.4}$$

It is possible to estimate the standard deviation of optimized parameters. They are from a single iteration of the Levenberg-Marquard algorithm.

As shown in (1.3) the Jacobian matrix is required for the Levenberg-Marquard algorithm. After optimizing the model function and therefore finding a solution it is possible to calculate the covariance matrix. This is done by finding the inverse of the $J^T J$ matrix. There are several different approaches for this. A standard approach is QR decomposition which is explained in [9, pp. 19-23].

After the covariance matrix is estimated it can be used to estimate the standard deviation of each optimized parameter of the model function. Covariance describes how two variables behave as a pair. The covariance matrix contains the covariances between all variables. This means that in the diagonal of the covariance matrix each value is the covariance of each variable with itself. The covariance of a variable with itself is the same as its variance. Therefore in the example above the values in the diagonal in the covariance matrix are the variances of the according optimized parameters.

It is important to keep in mind that using this method of estimating the standard deviation, the result is only describing the deviation of the optimized parameters of the Jacobian matrix.

2.2.7 Normal distribution

Another aspect of evaluation of test results is their distribution. According to [30] assumptions about random variables can be made by describing their distribution. For example if the measured values are normal (or nearly normal) distributed it is possible to use an average over a sample of measurements for better results.

A normal distribution is given when:

- *“The mean, mode and median are all equal.*
- *The curve is symmetric at the center (i.e. around the mean, μ).*

- *Exactly half of the values are to the left of center and exactly half the values are to the right.*
- *The total area under the curve is 1." [30]*

The result of these properties is a bell shaped curve shown in Figure 2.11. Another aspect which can be seen in the same figure is the empirical rule also known as 68 95 99.7 rule. It describes that

- *"About 68% of values fall within one standard deviation of the mean.*
- *About 95% of the values fall within two standard deviations from the mean.*
- *Almost all of the values — about 99.7% — fall within three standard deviations from the mean." [30]*

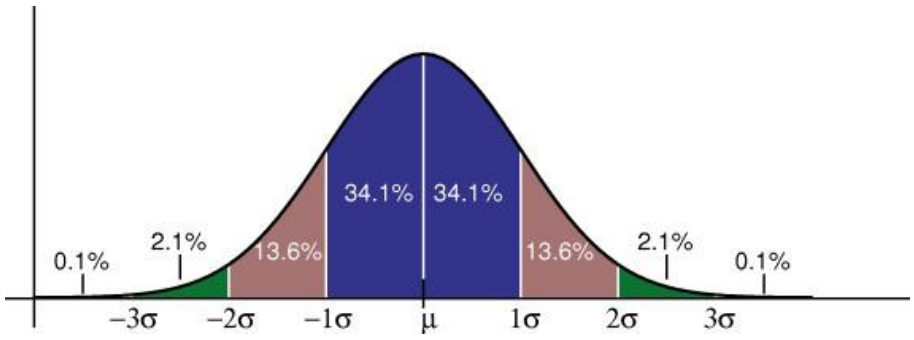

Figure 2.11: Empircal rule for normal distribution [30]

2.2.8 Euclidean distance

Evaluating localization results a basic aspect is the calculation of the distance between the estimated device location and the real device location. During runtime the real location is of course unknown. The test cases will be implemented with a predefined actual location where the device will be located during the test. To evaluate the test results the distance can be calculated with the Euclidean distance shown in (2.5). [31]

$$d = |x - y| = \sqrt{\sum_{i=1}^{n} |x_i - y_1|^2} \tag{2.5}$$

Chapter 3

Results

This chapter contains the results of this thesis. They are divided into the same parts implementation and evaluation as Chapter 2 was. The implementation part mostly consists of the new indoor localization implementation. Another part is the general GUI design as well as the GUI design for the smartwatches.

The evaluation part will analyze the results of the implementation with methods defined in 2.2.

3.1 Implementation

The implementation part is mostly about programming the new indoor localization approach. It starts with the analysis of iLocate. As a result a new GUI for better visualization is designed and afterwards a new localization approach. The final part of the implementation is the GUI design for the smartwatch client.

3.1.1 iLocate analysis

As first step of the implementation the provided app iLocate was analyzed and tested for accuracy of indoor localization. The provided beacons are entered into the app by defining their minor values. The indoor localization is a type of triangulation in two dimensions. This requires x and y coordinates of all three beacons as well as the estimated distance between device and each beacon. The coordinates of the beacons are entered in form of hard coded pixels as shown in Figure 3.1.

```
beacon1 = new Coordinate(10, 0);
beacon2 = new Coordinate(260, 0);
beacon3 = new Coordinate(130, 300);
```

Figure 3.1: beacon coordinates in pixels

The triangulation is done in the method *getPointX (*Figure 3.2) of the class *TriangulationCalculation*.

```
public static Coordinate getPointX(Coordinate beacon1, Coordinate beacon2,
Coordinate beacon3,double distanceB1, double distanceB2, double distanceB3,
double distB1B2, double distB2B3, double distB1B3 )
```

Figure 3.2: iLocate triangulation

Required parameters are the coordinates defined before as well as all three measured distances between device and the according beacon. For example *distanceB1* defines the measured distance between beacon1 and the device. The distances between all beacons are also hardcoded. This is unnecessary because these can be calculated with the Euclidean distance (see 2.2.8).

The distances between beacons and the device are estimated using the Estimote SDK method *Utils.computeAccuracy(beacon)*. This function delivers an estimated distance in meters. Because all calculation is done in pixels the measured distance is multiplied by a hardcoded factor of 100. The triangulation method *getPointX* uses hardcoded multipliers for x and y. A multiplier of one would mean that one pixel is interpreted as one centimetre.

In summary the room in iLocate is defined as a virtual object where beacons are located at fixed pixel locations. Afterwards actual distances are calculated with fixed modifiers during calculation of the device location. This means that for each room, not only the positions of beacons but also the multiplier for the distances as well as the interpretation within the *getPointX* method must be adapted.

The result is drawn in a fixed bitmap with a size of 220 * 230 pixels. This means that with a multiplier of one, the room is interpreted to 2,2m * 2,3m. The drawn room is shown in Figure 3.3.

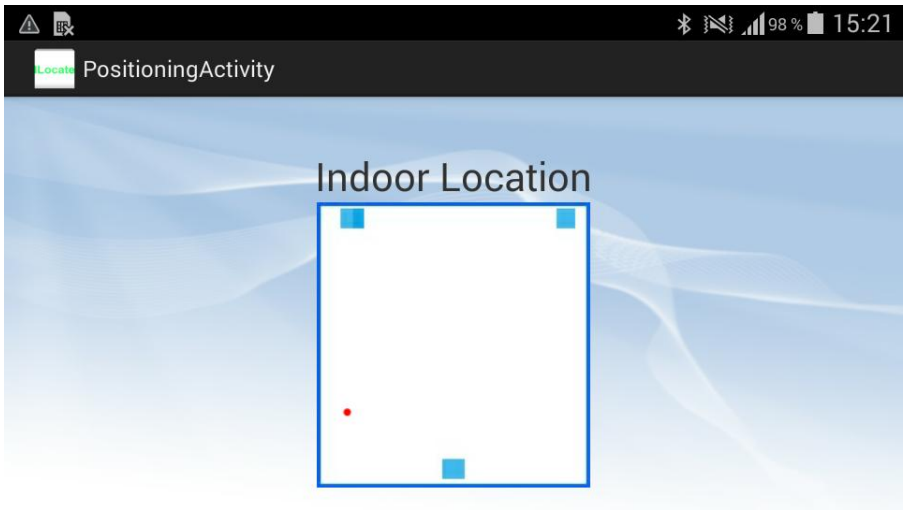

Figure 3.3: iLocate localization

The blue squares within the bitmap represent the beacons and are part of the fixed bitmap. This means that they do not represent their actual positions in the room defined earlier. The red circle is the calculated position of the user.

Another aspect of this bitmap is that the predefined values for the beacon coordinates (Figure 3.1) lie outside of the room. Beacon3 has a pixel height of 330 while the room is only 230 pixels high. As said earlier the distances between the beacons were also hardcoded. Those hardcoded values were also not correct based on the Euclidean distance.

First observations were that most times the calculated user position was outside of the bitmap and therefore never drawn. The location of the user did not stay at an expected range but varied unpredictably even when the device was not moving at all.

Because of this the calculation inside the *getPointX* method was analyzed. Mathematically the implementation is unclear. Most confusion arises from the fact that the comments inside the method do not match the code. Trying to better understand the results they were tested with unit tests in the new Estimote locate app. The unit tests are described in specific in A.1. In general the unit tests deliver the “measured” distances from device to each beacon as the Euclidean distance between them. This means that this test is linear and allows only a single solution which is already given. The result is that iLocate cannot find the solution and creates an error of more than one meter.

3.1.2 GUI design

The iLocate analysis shows that during implementation some confusion between drawing on a display and calculation of a position in meters happened. As a conclusion a new implementation must strictly distinguish between actual coordinates of objects like rooms and beacons and explicit drawing on a display. Actual coordinates should use the SI (International System of Units) unit of meters. Output on a display should be drawn as a scaled equivalent to the objects.

3.1.2.1 Dynamic drawing of the room

According to the Android developers documentation the standard concept for drawing static objects is Canvas [32]. It supports drawing of different objects in a two-dimensional plane.

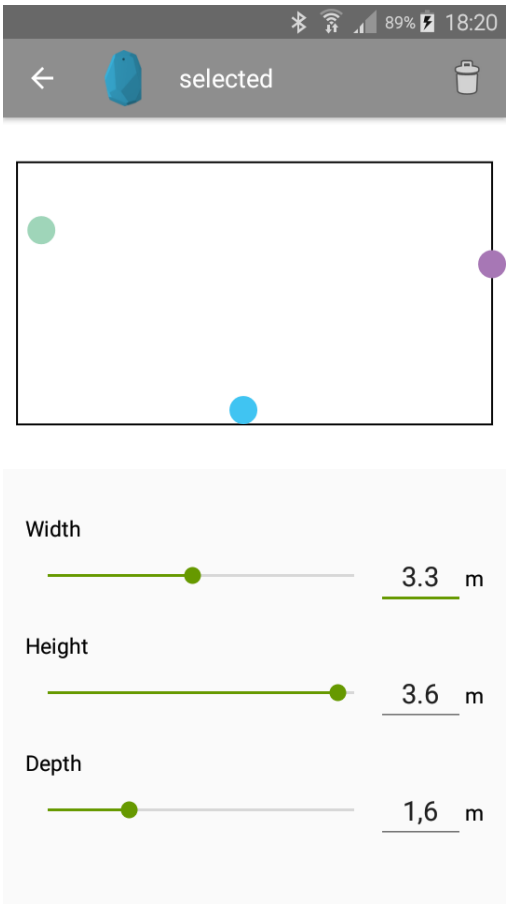

Figure 3.4: Room with beacons

The class *BeaconLocationView* is a custom view that draws the room and all of its beacons scaled to the available pixels of the device the app is running on. The basic

drawing class was taken from [33]. This class could only draw a circle on an explicit position on the screen and has been extended for drawing of rooms with their beacons in them.

Figure 3.4 shows the activity of moving a beacon (the blue one) inside the room. It can be seen that the room is drawn as a scaled black rectangular.

3.1.2.2 Multi-touch Zoom & Drag

To allow better analysis in bigger rooms with multiple beacons multi-touch zoom and drag were implemented (Figure 3.5). These gestures are not offered from any Android SDK class.

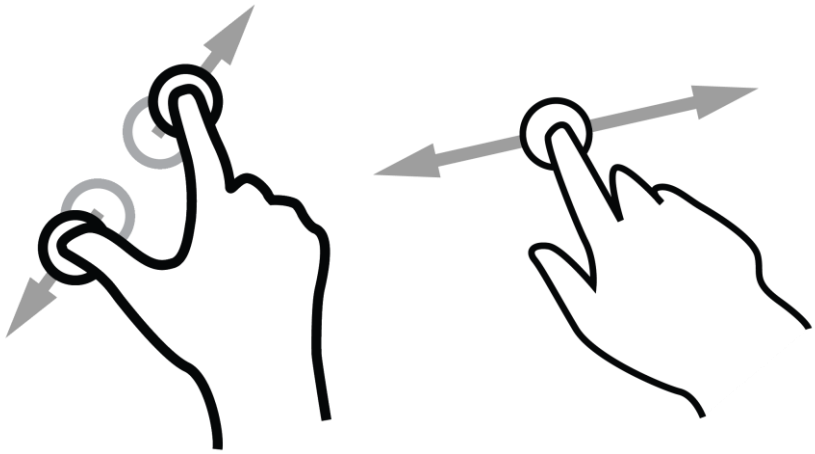

Figure 3.5: Multi-touch zoom and drag [34]

An abstract class called *MultiTouchView* implements these gestures. The basic logic was taken from a post from Adam Powell in the Android Developers Blog [35]. The example was adapted so that any view can extend the *MultiTouchView* and thereby implement these gestures. In general the drag and zoom functions are implemented by changing the scale and offsets and then calling *onDraw()* manually by invoking *invalidate()* to request a redraw of the canvas with the new values.

The class *BeaconLocationView* from 3.1.2.1 extends *MultiTouchView* and thereby allows zoom and drag.

3.1.2.3 Color of beacons

There are many possible colors for beacons which may also change in the future. It would be impractical to have different icons for each different color. To overcome this issue a beacon icon was taken from Estimote [17] and colorized gray manually. This allows multiplying any chosen color of a beacon onto the alpha level of the icon. The color is changed dynamically during runtime and can be chosen by the user. The gray beacon icon colorized in blue can be seen in Figure 3.4.

For selection of colors there is no default Android SDK implementation. There are several custom libraries published on GitHub for this reason. The library *ColorPicker* by Kristiyn Petrov [36] was found to be best fitting because it is currently the most up-to-date library and can be directly imported using Gradle.

This picker is initialized very easy by supplying a list of colors and the headline of the dialog. The resulting dialog is shown in Figure 3.6.

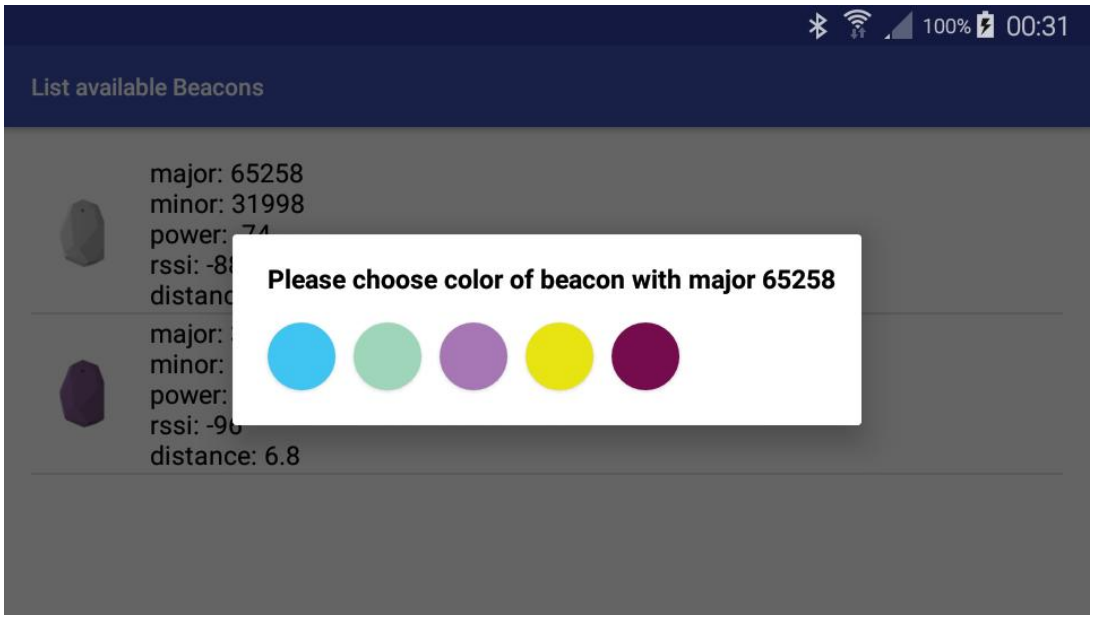

Figure 3.6: ColorPicker for beacon

3.1.3 Indoor localization

A java library called trilateration was found on GitHub that solves multilateration problems [37]. It provides several possible linear and non-linear approaches of which one is the Levemberg-Marquard method. The model function describing the trilateration problem is shown in (3.1).

$$r_i^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2, \qquad (i = 1,2, \dots, n) \tag{3.1}$$

where i … number of beacons

r_i ... distance of beacon i to the mobile device

$x_i, y_i \ z_i$ … tree-dimensional components of the location of beacon i

The model function is adapted automatically based on the given dimension of the beacons. For example it is possible to only use x and y coordinates. This would lead to the $(z - z_i)^2$ term to be removed from the function.

The library was included as described in its GitHub documentation. The beacons are entered as a double[][] where the first dimension defines the beacon and the second defines its location in the room which is n-dimensional. The measured distances to the beacons are entered in a second double[] where the index must match the according beacon in the first array.

While testing this implementation the results were good in comparison to the old iLocate implementation. However divergence from beacons with good (=low) RSSI was observed. The implementation seemed to favour intersections of distance spheres instead of single beacons with low RSSI. Figure 3.7 shows an example of this behaviour. Here the black user icon illustrates the estimated user location and the red circle shows the actual user location. The actual location was inserted afterwards.

For better understanding a testing mode was implemented where the measured distance of each beacon is drawn as a circle around it. The centre of the circle is the position of the according beacon and the radius is the estimated distance. In Figure 3.7 it can be seen that the implementation favours the intersections of the blue and green beacons. It should be expected that an error correction algorithm favours such beacons which deliver better estimated distances. In this example this would be somewhere between the blue and magenta beacons and would therefore better match the actual device location drawn as a red circle.

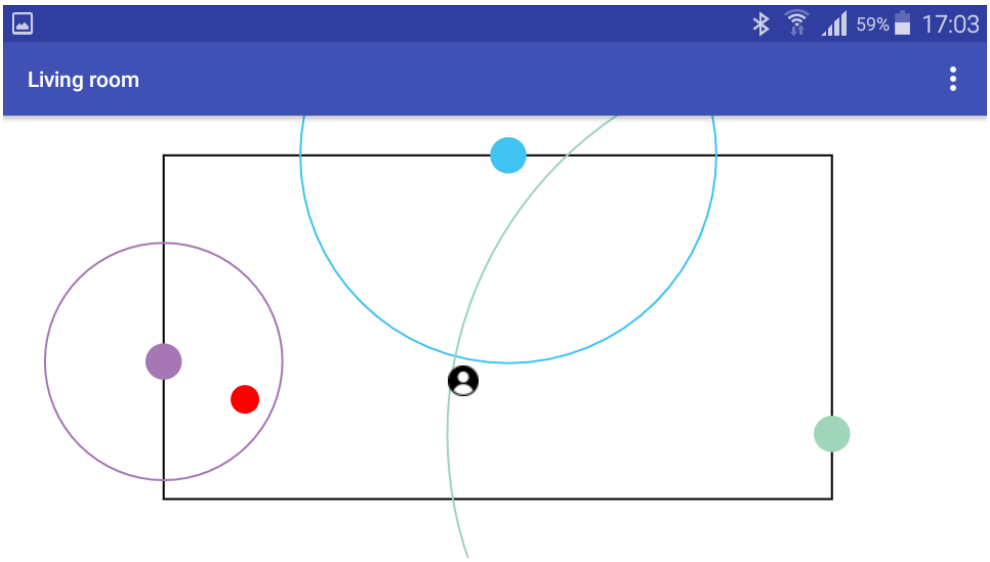

Figure 3.7: Divergence from near beacon (user icon from [38])

Another observed aspect is that in most cases the drawn circles do not intercept and are highly erratic. This was expected due to the nature of RSSI measurements (see 1.3.2). For better results it is necessary that the implementation does not favour intersections but rather find a solution in between beacons with good RSSI. For this reason the implementation was analyzed for any possible errors.

In the class *NonLinearLeastSquaresSolver* the trilateration problem is initialized as an instance of *LeastSquaresProblem* provided by *org.apache.commons.math3.fitting.leastsquares*. An instance of *LeastSquaresOptimizer* is then used to optimize the problem. In the implementation a *LevenbergMarquardtOptimizer* is used for this. This is also provided by *org.apache.commons.math3.fitting.leastsquares*. It was assumed that none of the default apache implementations had any errors. The only possibility was the initialization of the *LeastSquaresProblem* (see Figure 3.8). Relevant parts are the model function of the problem, the target value of the optimization, the initial point or the weights.

```
LeastSquaresProblem leastSquaresProblem = LeastSquaresFactory.create(
        function,
        new ArrayRealVector(target, false),
        new ArrayRealVector(initialPoint, false),
        new DiagonalMatrix(weights),
        null, MAXNUMBEROFITERATIONS, MAXNUMBEROFITERATIONS);
```

Figure 3.8: LeastSquaresProblem initialization [37]

The target value of optimization is a minimum in the model function and is therefore 0 for all dimensions (see Figure 3.9). The Levemberg-Marquard method is highly dependent on the initial start point from which the model is optimized. This can certainly be a reason for errors. In the implementation the centre of all beacons is used as start point. However errors resulting from badly chosen starting points would lead to different errors where the error correction would fail to deliver a result. [12] The results however are mostly within the room. The weights are the next logical possibility because they weight the measured distances. As shown in Figure 3.9 the original weights are the square of the according distance.

```
[...]
double[] target = new double[numberOfPositions];
double[] distances = function.getDistances();
double[] weights = new double[target.length];
for (int i = 0; i < target.length; i++) {
    target[i] = 0.0;
    weights[i] = (distances[i] * distances[i]);
}
[...]
```

Figure 3.9: Original weights [37]

This means that the higher the distance gets the higher its weight gets. This is certainly matching the result of Figure 3.7 as the result focuses on the intersection with the worst measured distance which is the green beacon.

3.1.3.1 Inverse square law

The inverse square law states that for point sources which radiate their influence equally in all directions the measured intensity is inversely proportional to the square of the distance. Figure 3.10 illustrates that this law comes from strictly geometric considerations. The result is that it is only true when there are no objects blocking the path of the radio waves. [39]

For indoor localization systems based on RSSI measurement this is not exactly true as discussed in 1.3.2. Consequently the result can never be perfect by applying the inverse square law. It is nevertheless a better approximation for weighting the measured distances than taking them to the square. This means that the original implementation shown in Figure 3.9 is the opposite of the inverse square law.

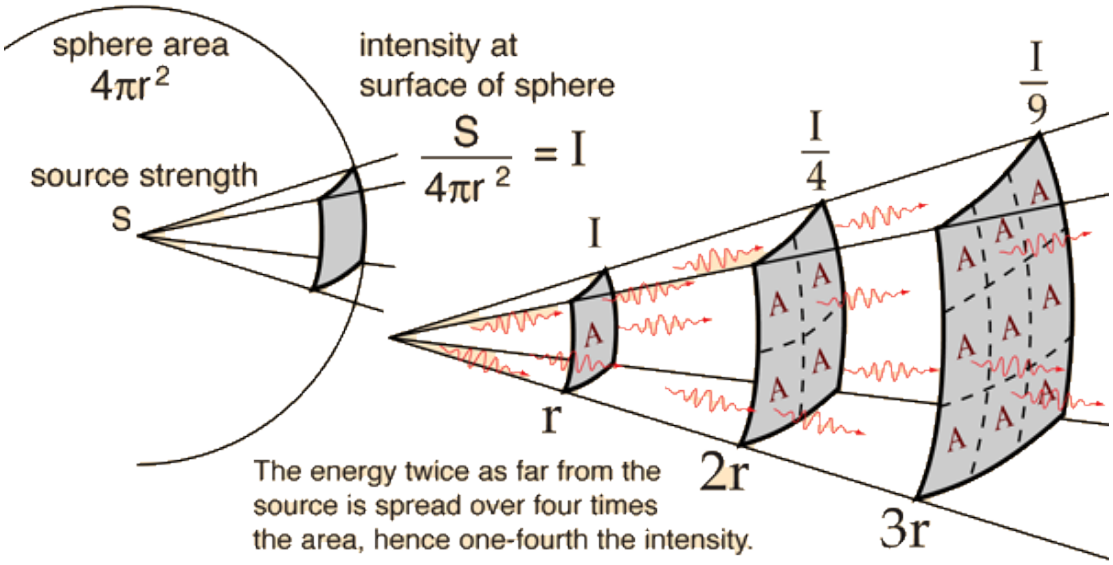

Figure 3.10: Inverse square law [39]

The original implementation from Figure 3.9 was adapted according to the inverse square law as shown in Figure 3.11.

```
[...]
for (int i = 0; i < target.length; i++) {
    target[i] = 0.0;
    weights[i] = inverseSquareLaw(distances[i]);
}
[...]
private double inverseSquareLaw(double distance) {
   return 1 / (distance * distance);
}
```

Figure 3.11: Inverse square law implementation

Figure 3.12 shows a test of this bug fix with the same device location as in Figure 3.7. The result now is very accurate to the actual user location. Keep in mind that during the test the location was jumping around for each measurement creating several different results in the vicinity of the magenta beacon.

Because of these results the original GitHub library was forked and a pull request with this bug fix was made. The changes were accepted as proposed and were released on the 9th of May 2017 (version 1.0.2).

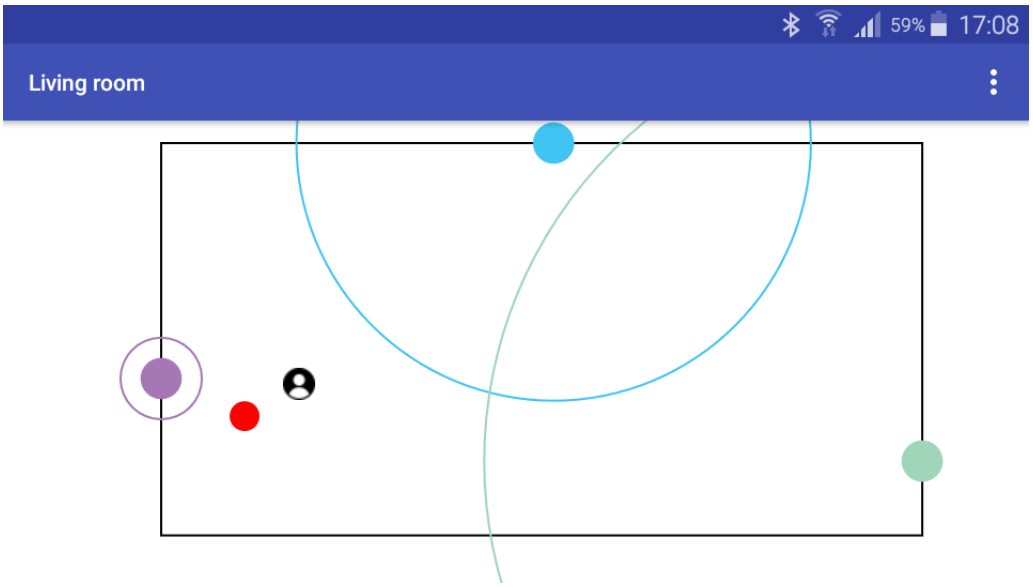

Figure 3.12: Fixed convergence to near beacon

3.1.4 Smartwatch GUI design

Considering smartwatches there are several aspects that need to be taken into account. The most important aspect is that with a screen size of only 1.5'' the amount of content on each view should be kept minimal. Figure 3.13 shows the two original Confidence design types which were used for normal sized android smartphones.

(a) iHomeLab design (b) high-contrast design

Figure 3.13: Original Confidence design types [22, p. 81]

The original iHomeLab design in Figure 3.13 (a) was chosen and adapted to smatwatch sized icons. The new design can be seen in Figure 3.14.

3.1.4.1 Scrolling between Pages - Infinite View Pager

In the original Confidence project scrolling between main pages was done with manually implemented animations. The user could switch the pages by moving a finger across the screen in the direction of the next page. This means that the user would move a finger left or right on the screen, to reach the next page. The implemented animations only animated the transition between the pages and not the movement of the finger. This means that there was no feedback whether the user's actions were being recognized or not.

The Android developers documentation for swipe views [40] states that developers should not wait for a gesture to complete before transitioning between pages. Android already offers an implementation for swiping views which animates the transition correctly. This implementation is called *ViewPager* [41].

Figure 3.14: ViewPager Swipe (moving hand from [42])

For safety and usability reasons it is extremely important that users never reach a dead end while swiping through the views to use e. g. the SOS function. The default adapter for undetermined number of pages in a *ViewPager* is the *FragmentStatePagerAdapter* [41]. However this adapter does not allow circular rotation between views because each view has a fixed position in the adapter. The following easy example illustrates this implementation where each name represents a view in the *ViewPager*: *SOS, Telephone* and *Calendar*. It is impossible to swipe right from the last page *Calendar* to reach the page *SOS*. Dynamically adding static

views to either side is supported but requires the whole *ViewPager* to refresh its adapter. Hence this solution is inefficient and would also require adding duplicate static views every time users swipe between pages.

There are several open source implementations which have already been implemented to overcome this issue. The easiest solution was found to be the *InfiniteViewPager* [43]. This implementation allows endless adding of views to an adapter. Also the adapter allows iteration by generic types (T) in a user defined order. Possible types of T would be any class like Calendar, Integer etc.

To allow the adapter to know the next and previous View, it has two abstract methods called *getNextIndicator()* and *getPreviousIndicator()*. The current indicator may be queried from the adapter to know the current shown view based on the chosen generic type. The obvious implementation would be a calendar which allows swiping through days, months, years and so on. This allows implementation of the required circular rotation between views.

```
private List<PageHolder> items;
[…]

@Override
public Integer getNextIndicator() {
    if (getCurrentIndicator() == items.size() - 1)
        return 0;
    return getCurrentIndicator() + 1;
}

@Override
public Integer getPreviousIndicator() {
    if (getCurrentIndicator() == 0)
        return items.size() - 1;
    return getCurrentIndicator() - 1;
}
```

Figure 3.15: InititePagerAdapter indicators

Figure 3.15 shows the implementation where the views are represented in a List<PageHolder> called ‘items’. The position of each item in this list matches the Integer indicator in the *InfinitePagerAdapter* implementation. When the adapter now asks for the next or previous indicator it dynamically compares the current indicator with the number of items. For example in case of reaching the last item this custom implementation returns 0 for the next page.

Going back to the earlier example with the three pages *SOS, Telephone, Calendar* this would mean that Calendar as the last page now references SOS as next indicator.

3.1.4.2 Back button implementation

Google generally states for all Android devices that

> *"Back navigation is how users move backward through the history of screens they previously visited. All Android devices provide a Back button for this type of navigation, so your app should not add a Back button to the UI." [44]*

Figure 3.16: Software back button

However all relevant smartwatches which were discussed in 2.1.1.2 are too small to implement a hardware button for the Android back button. To solve this, a swipe action is used by the manufacturers. The action is recognized as a swipe from outside the right border into the middle of the screen. This solution has several flaws in context of an AAL application. On the one hand it is not intuitive for the user how to navigate back to the previous view. Because there is no button the action must be known. On the other hand this solution does not recognize the correct action every time. Most times a normal swipe is recognized because the finger was not far enough outside of the screen when starting the swipe.

Because of this, software Back buttons are used in this app even while Google states that they should not be used.

3.1.4.3 Options Menu

For AAL applications complexity of each view should be kept low at all costs. This is why the ActionBar is not used and therefore also overflow option menus are not available. However like any other software this app needs admin settings to allow updates and information about the current user and version.

To allow admins and relatives to update the client a hidden approach was chosen to avoid confusion of primary users. It is possible to recognize LongClick events on the hardware home button. The user Alexander at stackoverflow has implemented a custom listener for this [45]. When *onHomeLongPressed()* of this listener is fired a Dialog with all possible actions is shown.

This solution is definitely not optimal as users may accidently open this dialog. It must be seen as a least problematic approach to still allow updates without connecting any other device for example a PC (Personal Computer). If a user accidently opens this dialog it may be closed by clicking outside of the dialog or by choosing the “back” option.

3.1.4.4 Optimizing recognition of click events

Pierre Schaschl describes that during an acceptance test of Confidence

> *“[…] only three persons pressed the buttons too long which resulted in unexpected behavior. It has to be considered to not use the OnClickListener for the buttons which are only fired after the button is released from the tap, but an OnTouchListener instead, where the down motion of the tap can already be detected.” [22, p. 84]*

This observation is seen as a major possibility to improve feedback to the user. In general people tend to press and hold buttons until some expected action happens. Without going deep into this subject it should suffice that most people know this behavior when a remote control does not register a pressed button. Most people tend to press the button harder hoping that it would work then.

As first company Apple developed a capacitive touchscreen which can also recognize the amount of force applied while touching. Apple calls this technology *Force Touch* and in the case of the iPhone 6 *3D Touch.* [46] This technology would

certainly match the above requirement but is only implemented in Apple devices until now. In case of smartwatches only the Apple Watch features this technology.

Because of this a software approach was developed. An *OnTouchListener* is used in combination with a Timer. This new class was called "*ExtendedOnClickListener*" because it is only used in context of click events. The full source to this class may be found in the appendix (Figure A.4).

The standard Android MotionEvent actions [47] are used to recognize the action a user is performing. *MotionEvent.ACTION_DOWN* is fired when the user starts a touch action. With this action a *Timer* is instantiated which counts the time the user presses the view. *MotionEvent.ACTION_UP* is the action indicating that the user has finished a touch action. *MotionEvent.ACTION_CANCEL* and *MotionEvent.ACTION_OUTSIDE* indicate that the user cancelled the touch action on the current view. However recognizing the sequence from ACTION_DOWN to ACTION_UP without a cancelling action is not enough for registering a click. This sequence could also mean that the user actually executed a swipe. Because of this the start position in the screen is saved and compared with the end position in a method called *isAClick* shown in Figure 3.17.

```
private static final float CLICK_ACTION_THRESHOLD = 12f;
[…]
private boolean isAClick(float startX, float endX, float startY, float
endY) {
    float differenceX = Math.abs(startX - endX);
    float differenceY = Math.abs(startY - endY);
    return !(differenceX > CLICK_ACTION_THRESHOLD || differenceY >
CLICK_ACTION_THRESHOLD);
}
```

Figure 3.17: Method isAClick

This method checks if the distance in the x plane or the y plane is greater than a defined threshold. During testing a reasonable threshold was found to be 12 pixels. In combination this implements the default case of a standard onClickListener. The timer mentioned before is now implementing the new press and hold action. In the implementation 700ms were found to be a reasonable time until a view should respond to the action of a user if there was no cancelling action. The timer is cancelled if such an action is recognized or a normal click was executed.

After either a click or a long click was recognized, the corresponding method in the interface *TouchActions* is called.

3.1.5 Active voice feedback

For critical features of the AAL app it is important to use all possible forms of communication with the user. Android offers developers the option of voice output of given texts. The according class is called *TextToSpeech* [48]. The main use case of this feature is the SOS function. It is always possible that a user chooses this function by accident in the main view. To avoid false positive alarms, voice output as well as vibration is used to inform the user that he is issuing an alarm. Figure 3.18 shows the view when an alarm is issued.

Figure 3.18: Issuing alarm

The TextToSpeech class is instantiated quite easy and can directly output any given text. As mentioned in 1.2.1 elderly hear lower tunes better than higher pitched tunes. TextToSpeech offers the option to set the pitch of voice as well as the rate of which text is spoken. This is best done after the object was initialized.

Most devices use a female voice by default for TextToSpeech. The pitch and speech rate have to be seen as deviation in percent to a normal female voice. In order to allow elderly persons a better understanding the voice output values were chosen to be 80% of the normal pitch and 90% of the normal speech rate.

Another aspect of voice output is that the volume depends on the system output level. Because many devices offer hardware buttons for increasing and decreasing the volume level it is possible that users can change it by accident. These buttons often cannot be overwritten or blocked without root rights. Because of this the volume

level is overwritten when speaking with *TextToSpeech*. This is done with the *AudioManager* in the method *setVolumeAndSpeakDelayed()*.

Another aspect of this method is that it is waiting 4 seconds before it starts speaking using a Handler called *mSpeakerHandler*. This is done because most times users realize if they accidently started to issue an alarm. If the activity if finished by a user interaction like pressing the software back button this handler is cleared. By leaving the activity, users may avoid that the client starts talking with *TextToSpeech* at all.

The method *speakText()* is responsible for the actual voice output. The most notable aspect of this method is that *TextToSpeech* can interpret punctuation marks. For example by adding dots pauses between sentences can be stretched.

3.1.6 Waking the device during test cases

For testing battery runtime it is important to keep the impact of testing methods minimal. They should not interfere with the energy management and should only wake the device for short intervals to check the current battery state.

According to the Android developers documentation [49] the correct method for testing battery runtime is to use an AlarmManager with inexact repeating. This means that an interval is defined in which the device should be woken. However inexact repeating does not guarantee that the device is woken exactly at each interval. The device is allowed to shift all present events which may wake the device. This allows execution of all inexact events at a time and going to sleep afterwards. The main advantage of this procedure is that the device can stay in deep sleep mode longer without need to wake up all the time. Hence this type of alarm reduces impact on battery runtime to a minimum.

The trade off that the interval of waking up is not guaranteed is irrelevant for most test cases of indoor localization. It does not matter if the device is woken exactly every 15 minutes. The interval may spread around as long as it is called at least somewhere in between the 15 minute interval.

After the device receives a Broadcast from the alarm it needs to ensure that it has enough time to read all required information and save it to the database. This is not guaranteed for a normal *BroadcastReceiver* which is normally used for in combination with *AlarmManager.* It executes the *onReceive()* method and goes

directly to sleep afterwards. A *StatefulBroadcastReceiver* is used in order to start a *WakefulService* which holds a wake lock. After the required work is done the wake lock is released and the device can go back to sleep.

3.2 Evaluation

This part is about evaluating the results of this thesis. First some fundamental observations are described. Afterwards the results of test cases defined in 2.2.1 are shown and analyzed.

3.2.1 Distribution of measured distance

During the ranging test of the Galaxy A3 2015 about 71,000 localization results were saved until the battery died. The chosen location for this test was “cooking” (see 2.2.4) with two dimensions and therefore three beacons.

Figure 3.19 shows the distribution of the first 30,000 results from this test. It can clearly be seen that the results are almost normal distributed with an average Euclidean distance to the actual location of ~1.8m.

The three sigma rule only applies for the standard normal distribution, which is definitely not given in this case. However for this test result the sample standard deviation was 0.499 and the three sigma rule was fulfilled with 66.61%, 95.64% and 99,89%.

Referring to 2.2.7 the conclusion is that by taking an average of several measurements the result should improve and some deviant results could be filtered.

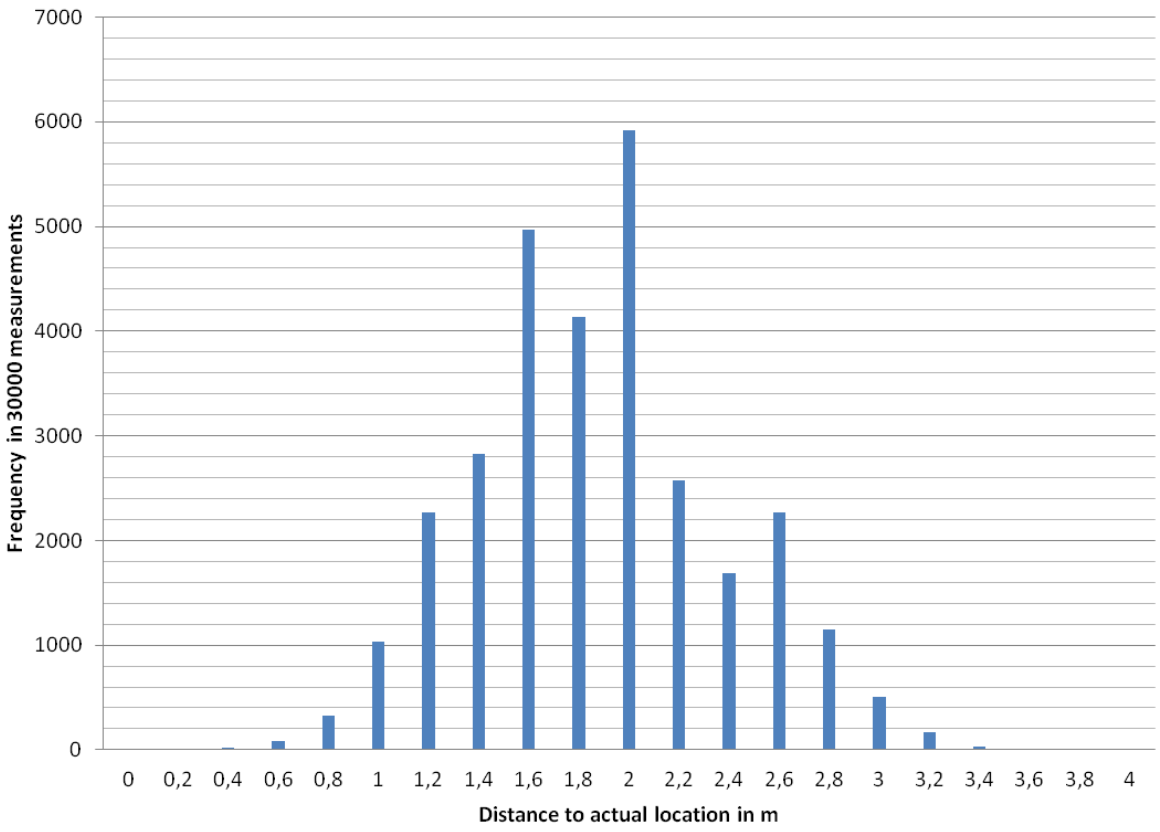

Figure 3.19: Distribution of measured distance

3.2.2 Comparison of trilateration methods and average

As next step the progression of results was analyzed. The test case was again “cooking” (see 2.2.4) which was run for 100s. The result is shown in Figure 3.20. All of the graphs represent the according Euclidean distance to the actual device location.

In general this result is a comparison of a linear solution which is already implemented in the GitHub library. This solution is only used for comparison and is not feasible for productive use. Also an average over the last ten values is plotted. As said earlier in this chapter, the average delivers a good solution for almost all cases.

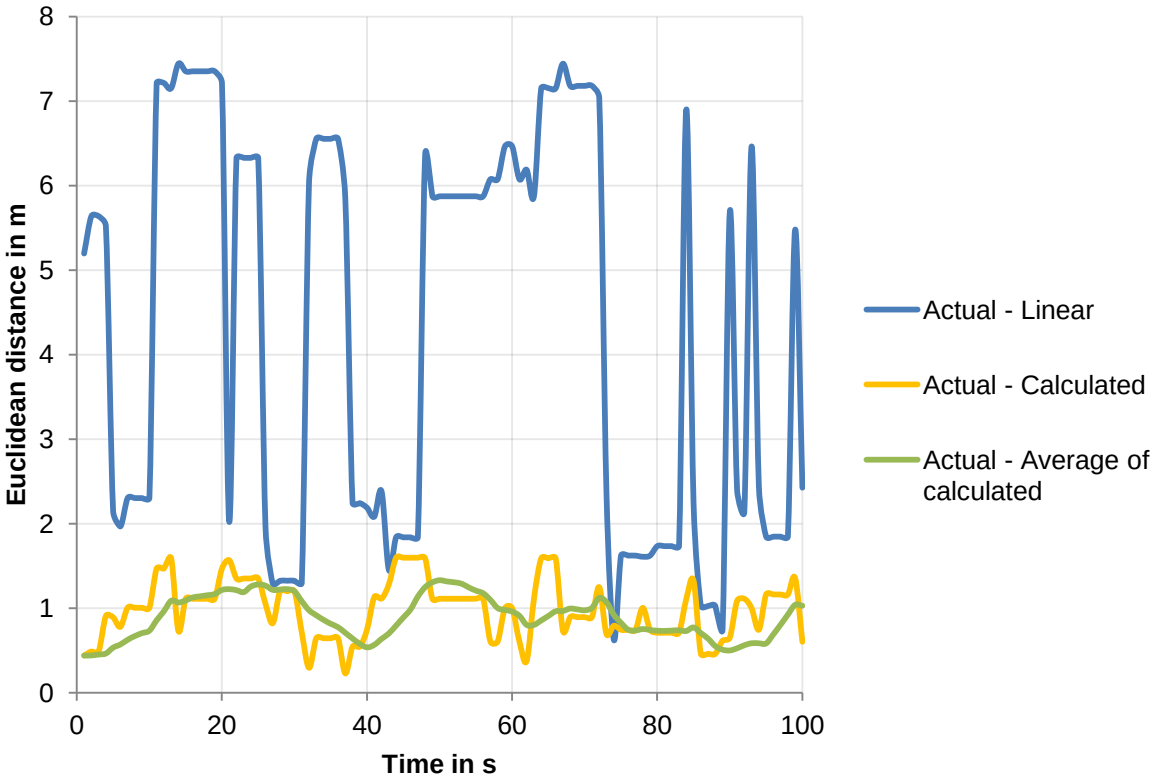

Figure 3.20: Euclidean distances to actual location 2D

3.2.3 Evaluation of explicit location tests

This part is about evaluating the explicit test cases defined in 2.2.4 as:

Case 1: Cooking

Case 2: Lying in front of television

Case 3: Lying in hallway

Each was tested for two- and three-dimensional localization with 100 test results per case. The results are shown in Table 3.1. The most obvious result is that using three-dimensional localization increases the error in almost every case independently of the test device and test case. Another result is that the three sigma rule is fulfilled for almost all test cases. Hence the distribution of results seems to be quite good.

Because using the third dimension includes using four instead of three beacons, the question arose if the error budget is growing simply because more beacons were used.

Device	Test Case	Average Euclidean distance to Actual location (m)	Sample σ	% in 1σ	% in 2σ	% in 3σ
Galaxy A3	1 (2D)	1,48	0,45	74,00%	92,00%	100,00%
Galaxy A3	1 (3D)	1,75	0,27	68,00%	97,00%	100,00%
Watch	1 (2D)	1,84	0,51	53,54%	94,95%	100,00%
Watch	1 (3D)	0,83	0,34	84,04%	93,62%	96,81%
Galaxy A3	2 (2D)	1,46	0,24	64,95%	98,97%	100,00%
Galaxy A3	2 (3D)	2,31	0,98	60,00%	100,00%	100,00%
Watch	2 (2D)	2,46	0,20	67,74%	95,70%	100,00%
Watch	2 (3D)	2,48	0,44	78,00%	93,00%	98,00%
Galaxy A3	3 (2D)	1,78	0,26	77,55%	91,84%	100,00%
Galaxy A3	3 (3D)	2,35	0,45	72,92%	94,79%	100,00%
Watch	3 (2D)	0,91	0,15	61,70%	95,74%	100,00%
Watch	3 (3D)	1,39	0,36	66,00%	100,00%	100,00%
Galaxy A3	4 Beacons 2D	0,76	0,47	78,79%	93,94%	96,97%

Table 3.1: Accuracy test results

Another test case was defined with four beacons and two-dimensional localization. It is an extension of the case “cooking”. The result is called “4 Beacons 2D” and it can be seen that the error significantly went down. To test this further another room was equipped with 10 beacons. The result is shown in Figure 3.21 where the red circle is the actual location. It was observed that the result was very unstable and almost never was in the vicinity of the real location.

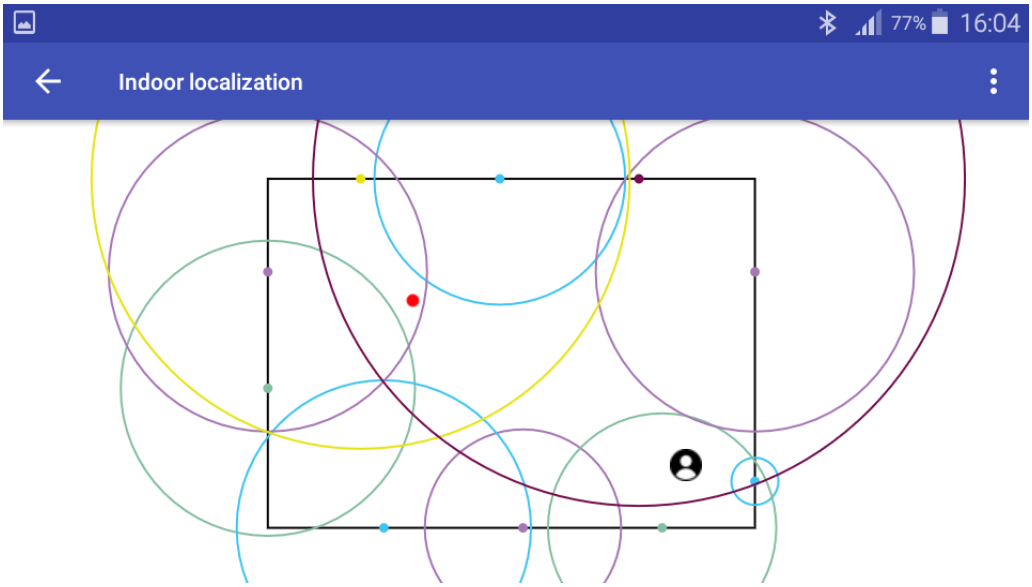

Figure 3.21: Test with ten beacons

3.2.4 Battery runtime comparison

The battery consumption was tested with the three tests “battery runtime”, “always ranging” and “movement monitoring” designed in 2.2.1.

The result is shown in Table 3.2. It must be kept in mind that the results are only referring to runtime without mobile perception activated.

Device	Test case	Runtime in dd:hh:mm
Samsung Galaxy A3	General runtime	03:21:17
simvalley AW-414.GO	General runtime	04:10:40
Samsung Galaxy A3	Always ranging	00:21:41
simvalley AW-414.GO	Always ranging	00:18:06
Samsung Galaxy A3	Movement monitoring	02:01:10
simvalley AW-414.GO	Movement monitoring	02:03:42

Table 3.2: Battery runtime comparison

Chapter 4

Discussion

This chapter is about interpreting the results of this thesis. Firstly the implementation of the smartwach client will be discussed. Afterwards the results of the new indoor localization approach are examined. The final part is reviewing the energy efficiency of indoor localization.

The leading questions of this will be answered in this chapter. They were:

- Can the accuracy of indoor localization be improved with least squares error correction?
- What is the distribution of the results? E.g. does the distribution suggest that taking the mean of several results improves accuracy?
- What is the impact on battery runtime when running indoor localization?
- Is it possible to use the results for AAL use cases?

4.1 Smartwatch optimization

The smartwatch optimization was successful in implementing a new view to the Confidence project. It was possible to implement the required functionality for such small displays without restrictions. ilogs used the view developed in this thesis to develop a new smartwatch client. The resulting product is already in sale at time of handing in this thesis. However this client does not include indoor localization at this time.

4.2 Indoor localization

The indoor localization app Estimote locate was developed successfully. It delivers good results between 1.5 and 2 meters deviation from the actual position. Also the results are reproducible and stable proving that least squares error correction is indeed improving the results.

The results are very close to being normal distributed. Using an average of ten measurements further improved the quality.

The following sections will discuss the results for the AAL use cases from 1.5.

4.2.1 Movement monitoring

The results are good enough for movement monitoring as long as only two dimensional localization is used. It is not possible to make assumptions about the depth of a device inside of a room. The three dimensional localization creates higher errors than the two dimensional localization. This means that with Estimote beacons it was not possible to make assumptions about the device being on the ground or not. Hence fall detection is not possible by directly localizing the device. Instead as suggested in 1.5 predefined use cases for rooms must be defined in order to recognize possibly critical situations.

4.2.2 Automatic assistance

The achieved accuracy is enough for offering automatic assistance based on the room the device is in. However recognizing explicit positions in a room for example if the user is exactly in front of a refrigerator is not possible. Also the high localization interval required for recognizing explicit locations cannot be maintained. This aspect will be discussed further in the battery efficiency section.

4.2.3 Activity monitoring

For activity monitoring the same is true as it is for automatic assistance. The accuracy of localization is enough to recognize if the owner is in the vicinity of a sensor recognizing an activity. Also the same trade off has to be made. Activity

recognition requires higher localization intervals than 15 minutes. This is certainly a problem for battery runtime.

4.3 Energy efficiency

The energy efficiency is the major factor defining which indoor localization approaches can be used and which cannot. All battery runtime tests were done without mobile perception activated. It can be seen that even short ranging for every 15 minutes nearly cuts the battery runtime in half. This is about the same amount of energy required for active GPS localization. This is not unexpected because the basic principles of triangulation and trilateration are the same.

Mobile perception has also a quite high impact on battery runtime and would therefore further reduce it.

Also these tests did not include normal usage of mobile devices during the tests. This means that the display was not used. This is the component which has the highest energy consumption by far.

In summary it is currently definitely impossible to run ranging all the time. Even the 15 minute interval of movement monitoring would possibly be too high for real world use.

Chapter 5

Outlook

In general this thesis has shown that indoor localization is possible with BLE beacons. It was greatly improved by applying least squares error correction. Further development is required to apply the achieved accuracy to real world use cases in AAL. It was proven that inside a room it is possible to differentiate the location within 1.5 to 2 meters.

Differentiation between rooms will be the next step that is required to achieve an AAL system. This can possibly be done with beacon regions which were not considered in this thesis.

In terms of visualization it is possible to expand the current implementation with canvas to not only draw a single room but a whole flat.

Finally a major aspect of mobile devices in general is currently a limiting factor for indoor localization. It has a major impact on battery runtime. Batteries need to be improved or devices need to become more efficient in order to have enough energy to run ranging for longer periods of time.

Bibliography

[1] Active and Assisted Living Programme, „Active and Assited Living Programme - About,“ 2015. [Online]. Available: http://www.aal-europe.eu/about/. [Zugriff am 07 03 2016].

[2] European Union, „eurostat,“ 27 11 2015. [Online]. Available: http://ec.europa.eu/eurostat/statistics-explained/index.php/People_in_the_EU_%E2%80%93_population_projections. [Zugriff am 07 03 2016].

[3] S. Meyer, H. Mollenkopf und B. Eberhardt, AAL in der alternden Gesellschaft - Anforderungen, Akzeptanz und Perspektiven - Analyse und Planungshilfe, Berlin und Offenbach: VDE Verlag GmbH, 2010.

[4] „Insurance firm to replace human workers with AI system,“ The Mainichi Newspapers Co., 30 12 2016. [Online]. Available: http://mainichi.jp/english/articles/20161230/p2a/00m/0na/005000c. [Zugriff am 14 03 2017].

[5] S. McLeod, „Maslow's Hierarchy of Needs,“ 2016. [Online]. Available: https://www.simplypsychology.org/maslow.html. [Zugriff am 27 05 2017].

[6] „GPS Accuracy,“ National Coordination Office for Space-Based Positioning, Navigation, and Timing, 10 02 2017. [Online]. Available: http://www.gps.gov/systems/gps/performance/accuracy/. [Zugriff am 20 03 2017].

[7] H. Karl und A. Willig, Protocols and Architectures for Wireless Sensor Networks, Chichester, West Sussex: John Wiley & Sons Ltd, The Atrium, Southern Gate, 2007.

[8] K. Plößl, Mehrseitig sichere Ad-hoc-Vernetzung von Fahrzeugen, Wiesbaden: Gabler Verlag / GWV Fachverlage GmbH, 2009.

[9] Å. Björck, Numerical methods for least square problems, Linköping: Society for Industrial and Applied Mathematics, 1996.

[10] Universität Zürich, „Einfache lineare Regression,“ 08 01 2017. [Online]. Available: http://www.methodenberatung.uzh.ch/de/

datenanalyse/zusammenhaenge/ereg.html. [Zugriff am 13 04 2017].

[11] C. L. Lawson und R. J. Hanson, Solving Least Sqaures Problems, New Jersey: Society for Industrial and Applied Mathematics, 1995.

[12] H. Yu und B. M. Wilamowski, „Levenberg–Marquardt Training," in s *The Industrial Handbook - INTELLIGENT SYSTEMS*, Boca Raton, Taylor and Francis Group, LLC, 2011, pp. 12.1-12.16.

[13] J. Plattner, *Implementation of a Non-Invasive Sensor Network for ADL/iADL Recognition in the Context of Active and Assisted Living,* Klagenfurt, 2016.

[14] S. Katz, A. B. Ford, R. W. Moskowitz, B. A. Jackson und M. W. Jaffe, Studies of illness in the aged: The index of adl: astandardized measure of biological and psychosocial function, Cleveland: JAMA, 1963, pp. 914-919.

[15] M. P. Lawton und E. M. Brody, „Assessment of older people: self-maintaining," Bd. 9, Gerontologist, p. 179–186.

[16] Estimote Inc., „Estimote," 2017. [Online]. Available: https://estimote.com/. [Zugriff am 11 06 2017].

[17] Estimote Inc., „Estimote - Our Product," 2017. [Online]. Available: https://estimote.com/press-kit/. [Zugriff am 24 05 2017].

[18] „Developer Docs - Part 3: Ranging beacons," Estimote Inc., 2014. [Online]. Available: http://developer.estimote.com/android/tutorial/part-3-ranging-beacons/. [Zugriff am 24 01 2017].

[19] Google Inc., „Android Wear," [Online]. Available: https://www.android.com/wear/. [Zugriff am 01 051 2017].

[20] PEARL.GmbH Deutschland, „simvalley MOBILE Produkte ANDROID-SMART-WATCH," 07 06 2017. [Online]. Available: http://www.simvalley-mobile.de/android-smart-watch-mtrkw-2794.html. [Zugriff am 11 06 2017].

[21] Salzburg Research, „CONFIDENCE – Mobility Safeguarding Assistance Service for People with Dementia," 2017. [Online]. Available: https://www.salzburgresearch.at/en/projekt/confidence_en/. [Zugriff am 01 05 2017].

[22] Pierre Schaschl, *Implementing an Ambient Assisted Living System Utilizing the Android Platform,* Klagenfurt, 2013.

[23] Dollinger & Junginger Unternehmensgesellschaft, „greenDAO: Android ORM for your SQLite database,“ 2016. [Online]. Available: http://greenrobot.org/greendao/. [Zugriff am 06 09 2017].

[24] Dollinger & Junginger Unternehmensgesellschaft, „Android ORM Performance 2016,“ 2016. [Online]. Available: http://greenrobot.org/android/android-orm-performance-2016/. [Zugriff am 09 06 2017].

[25] „Estimote Cloud - What are App ID and App Token and what do I need them for?,“ Estimote Inc., 17 05 2016. [Online]. Available: https://community.estimote.com/hc/en-us/articles/203607313-What-are-App-ID-and-App-Token-and-what-do-I-need-them-for-. [Zugriff am 15 02 2017].

[26] M. Auer, J. Poelz, A. Fuernweger, L. Meyer und T. Tschurtschenthaler, „UMLet 14.2 Free UML Tool for Fast UML Diagrams,“ [Online]. Available: http://www.umlet.com/. [Zugriff am 11 06 2017].

[27] Beijing Easyhome Investment Holding Group Co.,Ltd , „EASYHOME - HOMESTYLER,“ 2017. [Online]. Available: https://www.homestyler.com/home. [Zugriff am 06 06 2017].

[28] E. W. Weisstein, „Variance,“ From MathWorld--A Wolfram Web Resource, 2017. [Online]. Available: http://mathworld.wolfram.com/Variance.html. [Zugriff am 31 05 2017].

[29] E. W. Weisstein, „Standard Deviation,“ From MathWorld--A Wolfram Web Resource, 2017. [Online]. Available: http://mathworld.wolfram.com/StandardDeviation.html. [Zugriff am 31 05 2017].

[30] Andale, „Statistics How To - Normal Distributions,“ 16 04 2017. [Online]. Available: http://www.statisticshowto.com/probability-and-statistics/normal-distributions/. [Zugriff am 08 06 2017].

[31] E. W. Weisstein, „Distance,“ From MathWorld--A Wolfram Web Resource, 2017. [Online]. [Zugriff am 03 06 2017].

[32] „Developers - Animation and Graphics Overview,“ [Online]. Available: http://developer.android.com/guide/topics/graphics/overview.html. [Zugriff am 05 02 2017].

[33] J. Altmann, „Motorola Milestone | Android,“ 16 05 2010. [Online]. Available: http://www.milestone-blog.de/android-development/einfaches-zeichnen-canvas/. [Zugriff am 13 09 2012].

[34] Ideum Inc., „Gestureworks Version 2,“ 2017. [Online]. Available: http://gestureworks.com/. [Zugriff am 11 06 2017].

[35] A. Powell, „Android Developers Blog,“ 09 06 2010. [Online]. Available: http://android-developers.blogspot.co.at/2010/06/making-sense-of-multitouch.html. [Zugriff am 05 02 2017].

[36] K. Petrov, „GitHub - ColorPicker,“ 28 09 2016. [Online]. Available: https://github.com/kristiyanP/colorpicker. [Zugriff am 19 02 2017].

[37] lemmingapex, „GitHub - trilateration,,“ 09 05 2017. [Online]. Available: https://github.com/lemmingapex/trilateration. [Zugriff am 24 05 2017].

[38] Treer, „openclipart - abstract user icon 3,“ 2016 04 23. [Online]. Available: https://openclipart.org/detail/247319/abstract-user-icon-3. [Zugriff am 2017 02 04].

[39] C. R. Nave, „HyperPhysics - Inverse Sqaure Law,“ 2012. [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/Forces/isq.html. [Zugriff am 17 05 2017].

[40] Android Open Source Project, „Developers - Swipe Views,“ [Online]. Available: http://developer.android.com/design/patterns/swipe-views.html. [Zugriff am 04 06 2016].

[41] Android Open Source Project, „Developers - Creating Swipe Views with Tabs,“ [Online]. Available: http://developer.android.com/training/implementing-navigation/lateral.html#horizontal-paging. [Zugriff am 11 04 2016].

[42] Mobiletuxedo, „flaticon,“ 2016. [Online]. Available: http://www.flaticon.com/free-icon/tactile-left-movement_14416. [Zugriff am 16 03 2016].

[43] Hayri Bakici, „InfiniteViewPager - an infinite paging ViewPager,“ 26 09 2013. [Online]. Available: http://thehayro.blogspot.co.at/2013/09/infiniteviewpager-infinite-paging.html. [Zugriff am 02 06 2016].

[44] Android Open Source Project, „Providing Proper Back Navigation,“ [Online]. Available: https://developer.android.com/training/implementing-navigation/temporal.html. [Zugriff am 05 06 2016].

[45] Alexander, „stackoverflow; Android: associate a method to home button of smartphone,“ 10 07 2015. [Online]. Available: http://stackoverflow.com/questions/31340715/android-associate-a-method-to-

home-button-of-smartphone/31340960#31340960. [Zugriff am 05 06 2016].

[46] Apple Inc., „Take Advantage of 3D Touch,“ 2017. [Online]. Available: https://developer.apple.com/ios/3d-touch/. [Zugriff am 24 04 2017].

[47] Android Open Source Project, „Developers - MotionEvent,“ [Online]. Available: https://developer.android.com/reference/android/view/MotionEvent.html. [Zugriff am 20 06 2016].

[48] Android Open Source Project, „Developers - TextToSpeech,“ [Online]. Available: https://developer.android.com/reference/android/speech/tts/TextToSpeech.html. [Zugriff am 22 08 2016].

[49] Android Open Source Project, „Developers - Scheduling Repeating Alarms,“ [Online]. Available: https://developer.android.com/training/scheduling/alarms.html. [Zugriff am 12 06 2017].

Appendix A

A.1 Localization unit tests

Because the iLocate localization implementation was unclear unit tests were written. The original class from iLocate handling triangulation is called *TriangulationCalculation* (Figure A.1). It was only adapted to automatically calculate the Euclidean distance. The Euclidean distance was implemented in the holder class *Coordinate* (Figure A.3) so that each instance of *Coordinate* can calculate the distance to another instance.

The unit tests are located in the class *LocalizationUnitTests* (Figure A.2). They test localization with both the iLocate and the new Estimote locate implementation. Input data for both methods were the original values from iLocate (Figure 3.1). The "measured" distances to the device are given as Euclidean distance to the Coordinate *"expectedResult = new Coordinate(120, 150)".* Afterwards the result of each method is compared to the expected result. This means that this is an ideal test case where there is a single solution. Any localization approach should find this solution with a distance margin of under 0.1 pixels error. Pixels are used because iLocate calculates with pixel locations and Estimote locate does not care about the size of input values.

The iLocate implementation failed this test with an actual distance of 101 pixel (=~1m). The linear Estimote implementation delivers a distance of $5.7 * 10^{-14}$. The non-linear implementation delivers a result of $1.4 * 10^{-14}$. These results did not change for different test locations, the numbers behaved similarly. Multiple runs of the same test deliver exactly the same results.

```
public class TriangulationCalculation {

    // Berechnen des Punktes X mit Koordinaten von Beacon 1,2,3, Distanzen
zwischen B1, B2 und B3 und Distanz zu Device

    public static Coordinate getPointX(Coordinate Beacon1, Coordinate
Beacon2, Coordinate Beacon3,
                                       double distanceB1, double
distanceB2, double distanceB3) {
```

```
        double distB1B2 = Beacon1.getEuclideanDistance(Beacon2);
        double distB2B3 = Beacon2.getEuclideanDistance(Beacon3);
        double distB1B3 = Beacon1.getEuclideanDistance(Beacon3);

        // Berechnung der Winkel f?r das Dreieck von Beacon1 und Beacon2
(Cosinussatz)

        double alphaB1B2 = StrictMath.cos((Square(distanceB2) +
Square(distB1B2) - Square(distanceB1)) / (2 * distanceB2 * distB1B2));
        double betaB1B2 = StrictMath.cos((Square(distanceB1) +
Square(distB1B2) - Square(distanceB2)) / (2 * distanceB1 * distB1B2));
        double gammaB1B2 = StrictMath.cos((Square(distanceB1) +
Square(distanceB2) - Square(distB1B2)) / (2 * distanceB1 * distanceB2));

        //double alphaB1B2 = StrictMath.acos((Square(distanceB2) +
Square(distB1B2) - Square(distanceB1))/(2*distanceB2*distB1B2));
        //double betaB1B2 = StrictMath.acos((Square(distanceB1) +
Square(distB1B2) - Square(distanceB2))/(2*distanceB1*distB1B2));
        //double gammaB1B2 = StrictMath.acos((Square(distanceB1) +
Square(distanceB2) - Square(distB1B2))/(2*distanceB1*distanceB2));

        // Gamma1 und Gamma 2 ausrechnen
        double gamma1B1B2 = 180 - 90 - betaB1B2;
        double gamma2B1B2 = 180 - 90 - alphaB1B2;

        // c1 und c2 ausrechnen, c1 = Distanz von B1 bis H?lfte c2 =
Distanz von B2 bis H?lfte
        double c1B1B2 = distanceB1 * StrictMath.sin(gamma1B1B2);
        double c2B1B2 = distanceB2 * StrictMath.sin(gamma2B1B2);

        //x1 = a*sin(Beta), x2 = b*sin(Alpha)
        double x1B1B2 = distanceB1 * StrictMath.sin(betaB1B2);
        double x2B1B2 = distanceB2 * StrictMath.sin(alphaB1B2);

        // c1 und c2 umrechhnen in Pixel. C1(px) und c2(px) sind die
Abst?nde von den Beacons aus gesehen
        // 0.583 und 0,6 kommt von zb ImageView Room ist width 290 = 300cm
, height 307px = 200 cm.
        double c1InPxB1B2 = c1B1B2 * 1.03;
        double c2InPxB1B2 = c2B1B2 * 1.03;

        double x1InPxB1B2 = x1B1B2 * 0.65;
        double x2InPxB1B2 = x2B1B2 * 0.65;

        //Neue x,y Koordinaten von User (B1.x +c1, Beacon1. y + x1) c1, x1
in Pixel
        Beacon1.X = Beacon1.X + c1InPxB1B2;
        Beacon1.Y = Beacon1.Y + x1InPxB1B2;
        Beacon2.X = Beacon2.X - c2InPxB1B2;
        Beacon2.Y = Beacon2.Y + x2InPxB1B2;


        // F?r Beacon 2 und Beacon3
        //double alphaB2B3 = StrictMath.acos((Square(distanceB3) +
Square(distB2B3) - Square(distanceB2))/(2*distanceB3*distB2B3));
        //double betaB2B3 = StrictMath.acos((Square(distanceB2) +
Square(distB2B3) - Square(distanceB3))/(2*distanceB2*distB2B3));
        //double gammaB2B3 = StrictMath.acos((Square(distanceB2) +
Square(distanceB3) - Square(distB2B3))/(2*distanceB2*distanceB3));
```

```
        double alphaB2B3 = StrictMath.cos((Square(distanceB3) +
Square(distB2B3) - Square(distanceB2)) / (2 * distanceB3 * distB2B3));
        double betaB2B3 = StrictMath.cos((Square(distanceB2) +
Square(distB2B3) - Square(distanceB3)) / (2 * distanceB2 * distB2B3));
        double gammaB2B3 = StrictMath.cos((Square(distanceB2) +
Square(distanceB3) - Square(distB2B3)) / (2 * distanceB2 * distanceB3));

        double gamma1B2B3 = 180 - 90 - betaB2B3;
        double gamma2B2B3 = 180 - 90 - alphaB2B3;
        double c1B2B3 = distanceB2 * StrictMath.sin(gamma1B2B3);
        double c2B2B3 = distanceB3 * StrictMath.sin(gamma2B2B3);

        double x1B2B3 = distanceB2 * StrictMath.sin(betaB2B3);
        double x2B2B3 = distanceB3 * StrictMath.sin(alphaB2B3);

        double c1InPxB2B3 = c1B2B3 * 1.03;
        double c2InPxB2B3 = c2B2B3 * 1.03;

        double x1InPxB2B3 = x1B2B3 * 0.65;
        double x2InPxB2B3 = x2B2B3 * 0.65;

        Beacon2.X = Beacon2.X + c1InPxB2B3;
        Beacon2.Y = Beacon2.Y + x1InPxB2B3;
        Beacon3.X = Beacon3.X - c2InPxB2B3;
        Beacon3.Y = Beacon3.Y + x2InPxB2B3;

        // F?r Beacon 1 und Beacon3
        //double alphaB1B3 = StrictMath.acos((Square(distanceB1) +
Square(distB1B3) - Square(distanceB3))/(2*distanceB1*distB1B3));
        //double betaB1B3 = StrictMath.acos((Square(distanceB3) +
Square(distB1B3) - Square(distanceB1))/(2*distanceB3*distB1B3));
        //double gammaB1B3 = StrictMath.acos((Square(distanceB3) +
Square(distanceB1) - Square(distB1B3))/(2*distanceB3*distanceB1));

        double alphaB1B3 = StrictMath.cos((Square(distanceB1) +
Square(distB1B3) - Square(distanceB3)) / (2 * distanceB1 * distB1B3));
        double betaB1B3 = StrictMath.cos((Square(distanceB3) +
Square(distB1B3) - Square(distanceB1)) / (2 * distanceB3 * distB1B3));
        double gammaB1B3 = StrictMath.cos((Square(distanceB3) +
Square(distanceB1) - Square(distB1B3)) / (2 * distanceB3 * distanceB1));

        double gamma1B1B3 = 180 - 90 - betaB1B2;
        double gamma2B1B3 = 180 - 90 - alphaB1B2;

        double c1B1B3 = distanceB3 * StrictMath.sin(gamma1B1B3);
        double c2B1B3 = distanceB1 * StrictMath.sin(gamma2B1B3);

        double x1B1B3 = distanceB3 * StrictMath.sin(betaB1B3);
        double x2B1B3 = distanceB1 * StrictMath.sin(alphaB1B3);

        double c1InPxB1B3 = c1B1B3 * 1.03;
        double c2InPxB1B3 = c2B1B3 * 1.03;
        double x1InPxB1B3 = x1B1B3 * 0.65;
        double x2InPxB1B3 = x2B1B3 * 0.65;

        Beacon1.X = Beacon1.X - c1InPxB1B3;
        Beacon1.Y = Beacon1.Y + x1InPxB1B3;
        Beacon3.X = Beacon3.X + c2InPxB1B3;
        Beacon3.Y = Beacon3.Y + x2InPxB1B3;

        // Mittelwert berechnen
```

```
        double x = (Beacon1.X + Beacon2.X + Beacon3.X) / 3;
        double y = (Beacon1.Y + Beacon2.Y + Beacon3.Y) / 3;


        // Neue Coordinaten mit Position vom User
        Coordinate result = new Coordinate(x, y);

        return result;
    }

    public static double Square(double x) {
        return Math.pow(x, 2);
    }
}
```

Figure A.1: iLocalte class TriangulationCalculation

```
public class LocalizationUnitTests {

    private Coordinate b1;
    private Coordinate b2;
    private Coordinate b3;
    private List<Room2Beacon> beacons;
    private Coordinate expectedResult;

    @Test
    public void testExactCaseILocate() throws Exception {
        initializeLocation();
        Coordinate resultOfiLocate = TriangulationCalculation.getPointX(b1,
b2, b3,
                b1.getAccuracy(),
                b2.getAccuracy(),
                b3.getAccuracy());
        double distance =
expectedResult.getEuclideanDistance(resultOfiLocate);
        assertEquals(0, distance, 0.2d);
    }

    @Test
    public void testExactCaseEstimoteLocate() throws Exception {
        initializeLocation();
        TrilaterationHelper trilaterationHelper = new
TrilaterationHelper(TestType.ACCURACY_2D);
        if (!trilaterationHelper.trilaterationIsPossible(beacons)) {
            throw new Exception();
        }
        RealVector linear = trilaterationHelper.getLinearSolution();
        double distance = expectedResult.getEuclideanDistance(linear);
        outputAndTestResult(distance, "Result of linear: " + distance);

        RealVector result = trilaterationHelper.getOptimum().getPoint();
        distance = expectedResult.getEuclideanDistance(result);
        outputAndTestResult(distance, "Result with error correction: " +
distance);
    }

    private void outputAndTestResult(double distance, String x) {
        System.out.println(x);
        assertEquals(0, distance, 0.2d);
    }
```

```
    private void initializeLocation() {
        b1 = new Coordinate(10, 0);
        b2 = new Coordinate(260, 0);
        b3 = new Coordinate(130, 300);
        beacons = new ArrayList<>();
        beacons.add(b1);
        beacons.add(b2);
        beacons.add(b3);
        expectedResult = new Coordinate(120, 150);
        b1.setDistanceTo(expectedResult);
        b2.setDistanceTo(expectedResult);
        b3.setDistanceTo(expectedResult);
    }
}
```

Figure A.2: Localization unit tests

```
/**
 * Test object which works for both iLocate and EstimoteLocate
 */
public class Coordinate implements Room2Beacon {

    public double X;
    public double Y;

    private double distanceToDevice;

    public Coordinate(double x, double y) {
        X = x;
        Y = y;
    }

    public void setDistanceTo(Coordinate to) {
        this.distanceToDevice = getEuclideanDistance(to);
    }

    public double getEuclideanDistance(Coordinate to) {
        return calculateEuclideanDistance(to.X, to.Y);
    }

    public double getEuclideanDistance(RealVector to) {
        return calculateEuclideanDistance(to.getEntry(0), to.getEntry(1));
    }

    private double calculateEuclideanDistance(double x, double y) {
        return Math.sqrt(Math.pow(X - x, 2) + Math.pow(Y - y, 2));
    }

    @Override
    public int getColor() {
        return 0;
    }

    @Override
    public Double getAccuracy() {
        return distanceToDevice;
    }

    @Override
```

```
    public ThreeDimensionalVector getLocationInRoom() {
        return new ThreeDimensionalVector(-1L, X, Y, null);
    }
}
```

Figure A.3: Class Coordinate

A.2 ExtendedOnClickListener

```
public class ExtendedOnClickListener implements View.OnTouchListener {

    private static final float CLICK_ACTION_THRESHOLD = 12f;
    private static final long LONG_CLICK_DURATION = 700;

    private float startX;
    private float startY;
    private boolean actionFinished;
    private Timer holdTimer;

    private TouchActions registeredForTouch;

    public ExtendedOnClickListener(TouchActions t) {
        registeredForTouch = t;
    }

    public interface TouchActions {

        /**
         * Called when a view was clicked without cancelling the action
         *
         * @param view the view being clicked
         */
        void onClick(View view);

        /**
         * Called when a view was LongClicked without cancelling the action
         *
         * @param view the pressed view
         */
        void onLongClick(View view);
    }

    @Override
    public boolean onTouch(final View view, MotionEvent event) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                view.setPressed(true);
                actionFinished = false;
                startX = event.getX();
                startY = event.getY();
                instantiateTimer(view);
                return true;
            case MotionEvent.ACTION_CANCEL:
            case MotionEvent.ACTION_OUTSIDE:
                cancelTouchListener(view);
                actionFinished = true;
                break;
```

```
            case MotionEvent.ACTION_UP:
                cancelTouchListener(view);
                float endX = event.getX();
                float endY = event.getY();
                if (!actionFinished) {
                    if (isAClick(startX, endX, startY, endY))
                        registeredForTouch.onClick(view);
                }
                actionFinished = true;
                break;
        }
        return false;
    }

    public void instantiateTimer(final View view) {
        cancelTimer();
        holdTimer = new Timer();
        holdTimer.schedule(new TimerTask() {
            @Override
            public void run() {
                if (!actionFinished)
                    registeredForTouch.onLongClick(view);
                actionFinished = true;
                cancelTimer();
            }
        }, LONG_CLICK_DURATION);
    }

    private void cancelTimer() {
        if (holdTimer != null) {
            holdTimer.cancel();
            holdTimer = null;
        }
    }

    public void cancelTouchListener(View view) {
        view.setPressed(false);
        cancelTimer();
    }

    private boolean isAClick(float startX, float endX, float startY, float
endY) {
        float differenceX = Math.abs(startX - endX);
        float differenceY = Math.abs(startY - endY);
        return !(differenceX > CLICK_ACTION_THRESHOLD || differenceY >
CLICK_ACTION_THRESHOLD);
    }
}
```

Figure A.4: ExtendedOnClickListener

Table of abbreviation

AAL	Active and Assisted Living
EU	European Union
GPS	Global Positioning System
IPS	Indoor Positioning System
RSSI	Received Signal Strength Indication
ToA	Time of Arrival
TDoA	Time Difference of Arrival
ADL	Activities of Daily Living
iADL	instrumented Activities of Daily Living
BLE	Bluetooth Low Energy
SIM	Subscriber identity module
ORM	Object-relational mapping
SDK	Software Development Kit
SI	International System of Units
PC	Personal Computer

Table of Figures

List of Tables