BACHELOR THESIS

THE TRACEABILITY OF MEDICAL EQUIPMENT THROUGH HOSPITALS AND RETIREMENT HOMES

Dennis Rieffe
S1239619

CREATIVE TECHNOLOGY & INDES B.V.

EXAMINATION COMMITTEE:

CHAIRMAN: O. BANOS LEGRAN, PHD
CRITICAL OBSERVER: DR. IR. B.J.F. VAN BEIJNUM
EXTERNAL MEMBER: E. WOLDRING, MSC

18-01-2017

UNIVERSITY OF TWENTE.

“The wisest men follow their own direction”
Euripides (484 BC – 406 BC)

Abstract

Staff working in healthcare and retirement homes is facing high workloads and accordingly high stress levels are reported by healthcare personal. Healthcare staff has to work in an effective way to reduce their workload. In hospitals and retirement homes valuable time is lost when searching for medical devices and supporting aids like lifts, beds, etc. Another disadvantage of lost equipment is that the required maintenance is not conducted at the correct moment. A system that easily tracks and finds such devices would save valuable time and accordingly would reduce the work stress of healthcare staff and improve the quality of work. This thesis describes the feasibility study of a simple and low cost tracking system for medical devices based on existing IT infrastructure available in every hospital and retirement homes nowadays: A Wi-Fi network.

An indoor Wi-Fi fingerprint system was developed, tested and evaluated. This study has shown that a tracking system based on Wi-Fi position is feasible.

Even though the current system is functional, it is recommended that the interface, hardware and implementation method are improved before commercialisation.

Acknowledgements

Thank you very much to Oresti Banos for being my supervisor and helping me to the finish line. The support was very helpful and appreciated. I enjoyed the freedom and the trust he gave me to explore my own possibilities but also the extensive and helpful feedback he provided when asked for.

I would also like to thank Bert-Jan van Beijnum for his guidance and for reviewing my work. His welcome advice kept me on to the right path.

A special thanks to Erik Woldring and the personnel from Indes. I am very glad for the chances and opportunities Indes gave me. The way the project was set up and the given freedom allowed me to learn a lot from the experience. Moreover, advice was always given when solicited.

Table of contents

Abstract v
Acknowledgements vii
Table of contents viii
List of figures ix
List of tables x
List of boxes xi
List of abbreviations and acronyms xii
1. Introduction 1
1.1. Defining scope 1
1.2. Research questions 2
2. State of the art 3
2.1. Measurement principles 3
2.2. Time of flight 3
2.3. Angle of arrival 5
2.4. Signal strength 6
2.5. Signal behaviour 7
2.6. Filter 10
2.7. Wireless communication technologies 11
3. Requirement analysis 14
3.1. Introduction of stakeholders 14
3.2. Overview of interview findings 14
3.3. List of requirements 16
4. System design 18
4.1. Choice of technique 18
4.2. Advantages and disadvantages 19
4.3. Architecture 20
5. Implementation 24
5.1. Tag 24
5.2. Server 28
6. Evaluation 40
6.1. Setup and method for searching times experiment 40
6.2. Experimental results: searching times 42
6.3. Setup and method for accuracy experiment 42
6.4. Experimental results: accuracy 43
6.5. Discussion 45
6.6. Evaluation of requirements 47
7. Conclusion and future work 49

7.1. Conclusion 49
7.2. Future work 50
8. References 51
Appendix A 52
Appendix B 53

List of figures

Figure 1: Schematic of the TOA localization in two dimensions. The two sided arrows indicate the distance from the transmitters (at (x1,y1) and (x2,y2)) to the object (red diamond). The object could theoretically be at both intersections of the object; further analysis of the available data or additional information should determine which location to eliminate [3]. 4
Figure 2: Schematic of the AoA localization in two dimensions. (x1, y1) and (x2, y2) are two transmitters and the red diamond at (x,y) is the object. The two sided arrows indicate the distance between the transmitters and the object. The angles θ1 and θ2 are shown by the circular arcs [3]. 5
Figure 3: Schematic of the propagation of a signal from a directional antenna [3]. 6
Figure 4: Schematic illustration of the structure of an antenna array. The distance d indicates the distance between each antenna and θ indicates the angle of arrival [5]. 6
Figure 5: A example of how signals travel through a building. The blue dots represent the transmitters. The further away from the transmitters the weaker the signal becomes. Red represents the strongest signal and blue the weakest signal. The RSSI value from different transmitters are collected at different locations and stored in a database [8]. 7
Figure 6: Schematic of the attenuation principle. The amplitude of a signal decreases over time. 8
Figure 7: Schematic of the principle of absorption. The undulating lines indicate where the signal was absorbed into a materials [10]. The horizontal line indicates the change in material. 8
Figure 8: Schematic of the principle of reflection [10]. The horizontal line indicates the change in material. 8
Figure 9: Schematic of the principle of Scattering. A signal encounters an object (dot) and scatters in all directions [10]. 9
Figure 10: Schematic of the principle of refraction. When encountering a new material, the signal bends to another, new direction. The horizontal line indicates the change in material [10]. 9
Figure 11: Schematic of the principle of diffraction, the signal encounters an object and travels around the object [10]. The vertical interrupted line represents an obstacle that causes the signal to bend. 10
Figure 12: Schematic illustration of the multipath effect. Two or more signals originating from the same transmitter arrive simultaneously at the same receiver. Both waves may have travelled different paths. [11]. 10
Figure 13: Flow chart of the different subsystems and how the different subsystems are related to each other. The dotted arrows represent the location at which the subsystem is based. 21
Figure 14: Flow chart giving the steps the user can perform when using the system to define their search. 23
Figure 15: Photograph of the used Raspberry Pi connected to the powerbank. 24
Figure 16: Class diagram of the classes from the software package which runs on the Raspberry PI. 25
Figure 17: Flow chart of all the steps the server performs once a search is initiated. 28
Figure 18: Class diagram generated by Eclipse of all the classes and their relations on the server side. 29
Figure 19: Map of the ground (left) and first (right) floor of the Zilverling building at the University of Twente [26]. 38
Figure 20: Map of the ground (left) and first (right) floor of the Zilverling building with all collected fingerprints (each represented by a bright red dot) The map itself is the Google Maps map [26]. The red dots were added to map by having the program display every single fingerprint. The exact location of each fingerprint is given by its x and y coordinates from the .csv file. 38

Figure 21: Map of the first floor of the Zilverling building illustrating the fingerprint nearest to the location of the stool, represented by the bright red dot [26]. 40
Figure 22: Photograph of the searching times experiment setup. The stool was placed in the middle of the lunch break area on the first floor of the Zilverling building. The red square indicates the Raspberry Pi attached to the powerbank. 41
Figure 23: Map of the ground floor of the Zilverling building giving the location of the experimental starting point, represented by the blue dot [26]. 41
Figure 24: Map of the ground floor of the Zilverling building giving the actual position of the Raspberry Pi in the first and second position. The locations are respectively indicated with the blue dots labelled 1 and 2. Two sets of measurements were performed at the second location. 43
Figure 25: Map of the ground floor of the Zilverling building. The actual location of the Raspberry Pi is indicated by the blue dot. The fingerprints that were returned for the experiment with the Raspberry Pi placed in the hallway are displayed by red dots and labeled with their fingerprint ID. The distances between the actual location and fingerprints 15 and 24 are respectively 5 and 7 meters. 44
Figure 26: Map of the ground floor of the Zilverling building. The actual location of the Raspberry Pi is indicated by the blue dot. The fingerprints that were returned for the experiment with the Raspberry Pi placed in the SmartXp are displayed by red dots and labeled with their fingerprint ID. The distances between the actual location and fingerprints 1, 2, 3 and 4 are respectively 7, 6, 2 and 5 meters. The fingerprint added for the last part of the experiment, fingerprint 65, was added at the exact same location as the blue dot. Since the location of the Raspberry pi and fingerprint 65 coincide, the distance between those two is 0 meters. 45

List of tables

Table 1: Overview of the different transmitter techniques and their respective ranges [1], [15] 13
Table 2: A Strength Weaknesses Opportunities Threats (SWOT) analysis of Wi-Fi fingerprinting, taken from Justin Stook, Planning an indoor Navigation service for a smartphone with Wi-Fi fingerprinting localization, 2011 [15]. 18
Table 3: Overview of the different major elements of the system and which elements use which subsystem. 21
Table 4: Summery of the different elements in the system and their purposes 22
Table 5: Summary of all classes in the program on the server side. For every class a small explanation is given about their role in the system. 30
Table 6: Example of the database. The BSSID's, RSSI's and (x, y) coordinates are given for each fingerprint. 33
Table 7: Experimental data of 20 subjects. The table gives the time (in seconds) in which the subjects were able to find the object with or without the software. The location given by the algorithm was accurate for all 10 subjects that used the software. Some of the control subjects were unable to find the object within the time limits, and their time was set to 300s. 42
Table 8: Measurements at the hallway locations (location 1). The result is a fingerprint ID. The distance between the actual location and the returned fingerprint was determined and is also given in the table. 43
Table 9: Measurements at the Smart XP locations (location 2) before adding an extra fingerprint. The result is a fingerprint ID. The distance between the actual location and the returned fingerprint was determined and is also given in the table. 43
Table 10: Measurements at the Smart XP locations (location 2) after adding an extra fingerprint. The result is a fingerprint ID. The distance between the actual location and the returned fingerprint was determined and is also given in the table. The previously returned fingerprints are also included in the table. 44
Table 11: Statistical analysis of the searching time data processed in SPSS. The data consist of the sample size, mean, standard deviation and standard error mean for the data of subjects with or without the use of the IPS software. 45

Table 12: Statistical analysis of the searching times data processed in SPSS. The results include, but are not limited to the significance, and mean difference. A T-Test with a 95% confidence interval was performed on the data to acquire the statistical analysis of the results of the experiment for the influence of the software to the searching times. 46

List of boxes

Box 1: Class AP, a struct based implementation of the values. 25
Box 2: Method onReceive, responsible for collecting the values 26
Box 3: Method startServer, responsible for sending the values 27
Box 4: Class Search responsible for collecting the user input 31
Box 5: Methods Start and filterList responsible for receiving and filtering the data 32
Box 6: Class Database, the struct setup for one unique fingerprint 33
Box 7: Method addPoints, responsible for loading the data from the .csv file to the memory 34
Box 8: Method findValue, resposible for adding fingerprints to the .csv file 35
Box 9: Method startAlgorithm resposible for collection all the values 36
Box 10: Method startComparing responsible. for comparing all the values in the database with the measured value 37
Box 11: Method sortList, responsible for sorting the resultList on the lowest value from the result of the Euclidean distance 37
Box 12: Method startLoading, responsible for loading and drawing the Image and the red dot at the specific location 39

List of abbreviations and acronyms

A-GPS	Assisted GPS (see GPS)
AOA	Angle of Arrival
AP	Access Point
BSSID	Basic Service Set Identifier
dBm	Decibel milliWatt
GPS	Global Positioning System
IEEE 802.11	See WLAN
IPS	Indoor Positioning System
INS	Inertial Navigation System
LAN	Local Area Network
LOS	Line of Sight
MAC	Medium/Media Access Control
RFID	Radio Frequency Identifier
RSSI	Received Signal Strength Indication
SSID	Service Set Identifier
TDOA	Time Difference of Arrival
TOA	Time of Arrival
TOF	Time of Flight
UWB	Ultra-Wide Band
Wi-Fi	Wireless Fidelity
WLAN	Wireless LAN (see LAN)
XML	eXtendible Markup Language

1. Introduction

In front of you lies the Bachelor Thesis report of Dennis Rieffe. This thesis is conducted at Indes B.V., based at the Kennispark Twente next to the University of Twente. Indes promote themselves with: *"Creating product people can rely on"*. Indes started out designing various products for a diverse customer base. During the years, the focus of Indes shifted and they started broadening their pool of employees with engineers. The company now works on the implementation and development phase as well as the design phase. An example of the projects Indes worked on in the past, is the development of the Sparta ION.
During this project Indes collected extensive e-bikes knowledge. E-bikes have a special motor that does not work independently but partially takes over the work the cyclist has to perform. This technology reduces the amplitude of problems like hills and headwind and makes them less challenging. Based on this idea, the addition of a supporting motor to existing technology would create new opportunities in a variety of applications.
The use of this idea for medical purposes is elaborated in this report since moving hospital beds can be a difficult and awkward task. A caster wheel could be placed under different medical appliances to make their handling easier. Through pressure sensors in the handle, this technology can help hospital or retirement home personnel in their daily chores. It could be used in combination with hospital beds, patient lift and food trolleys. In practice it is too expensive to implement this technology on every piece of equipment. Combining bed movers (which can be easily added to a hospital bed) and casters should avoid this problem.
An increased implementation of technology can already be witnessed in e.g. hospitals and fully developed and functional technologies make the use of the above mentioned technology promising.
Unfortunately, another totally different problem arises in practice. Because there are only a few bed movers and patient lifts available, and several staff members using them, they tend to get lost within the premises. This causes the personnel to search for the equipment. This is a time-consuming and costly flaw. This problem concerns on the one hand medical personnel and on the other hand maintenance staff. In the second case, the worst case scenario is when the equipment cannot be found, it is not checked and this may lead to malfunctioning of said equipment.
This is where the idea of Indes comes in. To assist medical personnel and maintenance staff, it would be helpful to be able to track the equipment within the building. This would reduce searching time, improve workload and result in more reliable medical equipment. With a trackable product, Indes has a unique opportunity to distinguish themselves from the competition.

1.1. Defining scope

Indes has multiple healthcare products on the market. To keep this thesis feasible and to establish its scope, initial boundaries were set by Indes. The first prototype should focus on two of Indes devices: the patient lift and the bedmover. Those products are used in two environments. The bedmover is mostly used in hospitals and the patient lift mainly in retirement homes. The implementation of the system should work in both environments. The products are used in the same way in both cases. They are available for all personnel on the work floor. Personnel share the equipment and no one has their dedicated to one.
The bedmover is developed by Indes to change work flow. In the current situation, most hospitals have logistics specialists that are only responsible for moving beds around. With its bedmover, Indes has a marketable product that enables everyone to use the equipment. Every hospital member is able to easily locate hospital beds.
In the current scenario, the logistic department is sometimes too late or they do not show up at all and the nurses are left to do the work anyway. Personnel use the product but do not return it to its supposed position. Therefore, the equipment gets lost within the building and the staff has to go through the entire building in order to find the bedmover.

When it comes to the patient lifts, the problem mostly occurs with the yearly check-up by the maintenance engineers. Patient lifts are mainly used in retirement homes to lift elderly people from their bed and to move them. During the check-up visits, the maintenance engineers aim to do all the maintenance at once to save time, but the lifts are in use all over the building. The engineers spend hours locating the lifts and may even miss a few. Since they cannot find them, the necessary maintenance is not performed. Therefore, some patient lifts will be used even though the maintenance is outdated.
Similarly, to bed movers, patient lifts get lost because personnel do not put them back after use. Inside the retirement homes, personnel make use of the lifts and do not place it back at its supposed position.

1.2. Research questions

To solve the above mentioned problem, Indes came up with a solution: an indoor positioning system should be created. The main research question was formulated as follows:

What is the best way for Indes to locate their bedmovers and patient lifts within a retirement home or hospital?

To answer this question, several sub-questions were drafted.

1. *What indoor positing systems are already do already exist?*
2. *What requirements should the system meet for a viable product?*
3. *Can an indoor positioning system reduce the searching time as compared to manual search?*
4. *Are Indes' clients willing to use this technique?*

The answers to these questions can be found in this thesis. A literature study was conducted to answer the first question. The knowledge gathered from the study can be found in chapter 'State of the art'. The requirements were setup after conducting interviews with different stakeholders and potential users of the products. The findings from the interviews can be read in 'Requirement analysis'. These interviews also allowed to answer the last sub-question. According to the requirements, an indoor positioning system was developed. The system design and implementation are visible in chapters 'System design' and 'Implementation'. Experiments where performed with this indoor positioning system to answer the third question. The setup and findings of these experiments can be found in the chapter 'Evaluation'.

2. State of the art

Tracing is something that has been done for a while. For thousands of years', people have been curious as to where they are and what distance they have travelled. Especially at sea, mankind based their navigation on stars and the sun.
In the sixties, the American army started with a more professional approach. They launched the satellite based TRANSIT system, which was a global positioning system primarily used by the navy to determine their exact location.
In the seventies, GPS (Global Positioning System) was develop by the American army. This made the use of position systems popular on a greater scale. GPS can be used worldwide for location determination. Nonetheless, a problem with GPS is that it does not work inside buildings because of the low accuracy within buildings. It can only be precise up until 5-50 meters inside [1]. This limitation makes GPS unsuited for tracking medical equipment in retiring homes and hospitals. Thus, other technologies were needed.
The technology for determining the location of people or objects inside a building is called an Indoor Positioning System (IPS). Radio waves, magnetic fields, acoustic signals or sensory information connected to mobile tags are the main methods used for IPS. Since there is not a standard system for IPS, there are several systems on the market. The feedback from the system can be on the device itself, like a mobile phone or on a workstation that can follow assets through a building. Most IPS's consist of three different parts. Multiple transmitters emit signals that can be picked up by receivers. The receiver's measurements can then be sent to a server that calculates and determines the location. To summarize, the three essential parts of a IPS are the server, the transmitters and the receivers [2].

2.1. Measurement principles

There is wide range of different technologies available on the market. In the following chapter, the various measurement principles and their mechanisms will be explained. First, an explanation will be given upon all the different kinds of techniques that can be used for location determination. Those techniques are Time of Flight, Angle of Arrival and Radio Signal Strength [3]. Most of the IPS that are now on the market use one of those techniques. Second, some wireless communication technologies are explained. Wireless communication technologies are methods to transfer information via a signal, e.g. Wi-Fi or Bluetooth.

2.2. Time of flight

The technique 'Time of Flight' (TOF) is highly applied for determining distances using radio waves. It measures the time that electromagnetic waves (e.g. light, radio) take to travel a given distance. The speed of light in vacuum is used as a reference, and measurements are in the magnitude of nanoseconds. There are two main derivatives of TOF: Time of Arrival (TOA) and Time Differential of Arrival (TDOA).
TOF has one important advantage in comparison to other methods for IPS's: the variance and thus the inaccuracy do not change over distance. This means that the precision of the location does not decrease over long distances [3].

2.2.1. Time of Arrival

In the TOA approach, the transmitter sends a signal at a specific time to a known receiver. The clocks of the receiver and the transmitter are synchronized. The receiver knows the time the signal was sent and the time it received the signal. The elapsed time multiplied by the speed of light, gives the distance between the two objects. In a 2D situation, knowing the distance of two transmitter points is sufficient to arrive at the location point of the object, see Figure 1 below. Based on this idea, the combination of several distances can also locate the object in a 3D environment.

There is also an alternative, an unsynchronized method which is part of the TOA method to calculate the distance between sender and receiver. In this scenario, the round trip is measured at the receiver using information exchanged between the receiver and the transmitter.

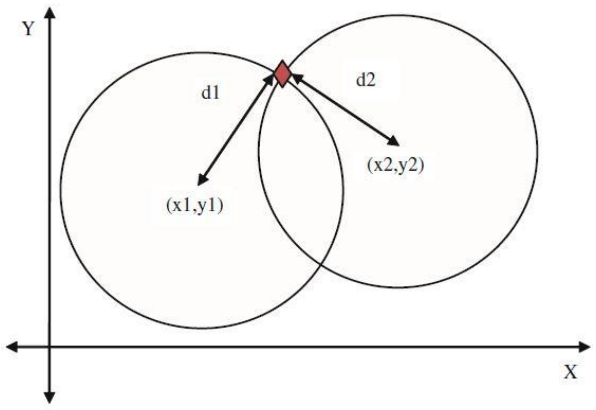


Figure 1: Schematic of the TOA localization in two dimensions. The two sided arrows indicate the distance from the transmitters (at (x1,y1) and (x2,y2)) to the object (red diamond). The object could theoretically be at both intersections of the object; further analysis of the available data or additional information should determine which location to eliminate [3].

2.2.2. Measuring Time of Arrival

Mathematics are used to determine the location of an object with a TOA solution. For an easy understanding, a 2D solution will be explained. The mathematics behind a 3D solution are similar but more complex.
Theoretically, to determine the position of an object, at least two transmitters are needed. The measured distance gives the possible locations of the object as a circle of a radius equal to the measured distance. Superposition of the circles of multiple transmitters gives at the most two intersections. Those give the possible locations of the object, as can be seen in Figure 1.
There is a possibility that one of the intersection lies in an impossible location. This way it will automatically fall off. If that is not the case, a third transmitter can be used to eliminate one of the intersections.
In a three dimensional space, it is a little more complicated. Here, at least three anchor points are needed to determine the location, because the measured distances result in spheres. Three equations can be formed, also resulting in two possible locations for the object. Additional information and knowledge of the layout can be used to eliminate one of the points.
In reality, more than three transmitters are necessary. Using more transmitters could lead to better accuracy. This is called over-determination. In an over-determination situation, different mathematical techniques are used to determine the location.
Time measurement of one radio wave allows the determination of the exact distance between the object and the transmitter. For indoor navigation systems, an accuracy of one meter is quite common. Light or radio waves take approximately 0.33ns to travel the said one meter, so a very accurate and precise clock is required to determine the time differences of a specific wave.
The relation between received and transmitted signal is given in equation 1.

$$r(t) = \alpha \sin(t - \tau) + n(t) \qquad (1)$$

The received signal, $r(t)$ is an amplified version of the transmitted signal, with a delay of τ. The noise component is $n(t)$. The propagation delay τ is determined through the Maximum Likelihood (ML) estimate of the correlation of the received signal and the transmitted signal [3].

2.2.3. Time Difference of Arrival

Because of the need for expensive and very good equipment with TOA, another method was devolved. For the TDOA method there is no need for synchronizations between the transmitter and receiver. For TDOA only the different transmitters are required to by synchronised. GPS is based on a TDOA method, since multiple transmitters send signals to the receiver. Those signals arrive at the receiver at different times. Based on those time differences between the received signals, a location can be determined [4].

2.3. Angle of arrival

The Angle of Arrival (AoA) method estimates the angle between the transmitter and the receiver. Figure 2 shows the AoA method in a two dimensional setup. The angles between the sensor and the positive x-axis at the two transmitters are θ_1 and θ_2 and the transmitter locations are (x1, y1) and (x2, y2).

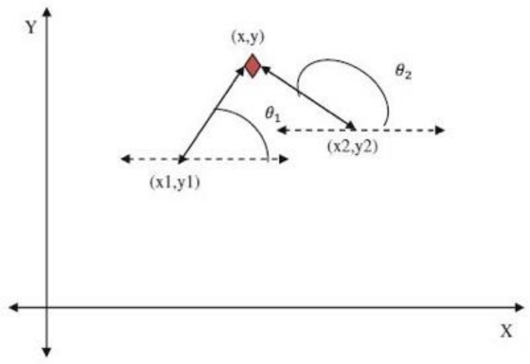


Figure 2: Schematic of the AoA localization in two dimensions. (x1, y1) and (x2, y2) are two transmitters and the red diamond at (x,y) is the object. The two sided arrows indicate the distance between the transmitters and the object. The angles ϑ1 and ϑ2 are shown by the circular arcs [3].

Once the angles are determined, the location can be determined using the following formulas (equations 2 and 3).

$$\tan(\theta_1) = \frac{y-y_1}{x-x_1}, \tan(\theta_2) = \frac{y-y_2}{x-x_2} \qquad (2)$$

$$y_1 - x_1 \tan(\theta_1) = y - x \tan(\theta_1) \qquad (3)$$

For a 3D space, a similar equation can be used, this time projecting into X, Y and Z directions. The projections onto Y-Z plane are considered and the angle with the Z axis defined as ϕ. The formulas for three dimensional AoA are as follows:

$$y_i - x_i \tan(\theta_i) = y - x \tan(\theta_i) \qquad (4)$$

$$y_i - z_i \tan(\phi_1) = y - z \tan(\phi_i) \qquad (5)$$

Measuring the angle in a multipath (see Multipath under Signal behaviour) environment can be difficult. Also, the greater the distance between the sensor and receiver, the larger the error.

2.3.1. Measuring the angle

There are two basic ways of measuring the angle. One of the methods uses a directional antenna with a known beam pattern. The other solution is an antenna array. An important note beforehand is that the directional/array antenna can be located either at the receiver or at the transmitter.
The first method consists of a directional antenna that emits radio signals in a particular direction. In Figure 3, an example is given of how a directional antenna emits in a particular direction. It shows the signal strength at an angle, when the distance between the transmitter and the receiver is kept constant. It is not possible to determine the direction with only one antenna. The combination of the data from the multiple antennas result in the angle of arrival.

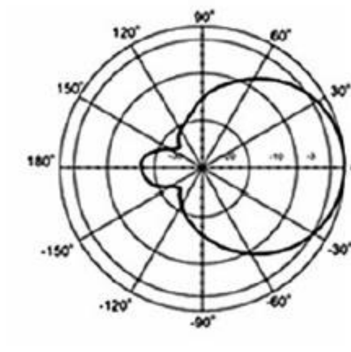


Figure 3: Schematic of the propagation of a signal from a directional antenna [3].

The second method uses an antenna array. In such an array, the elements of an antenna are separated by a fixed distance. In the example in Figure 4 this distance is called *d*. The object is compared with multiple single elements inside the array. The final angle is shown as θ in the figure.

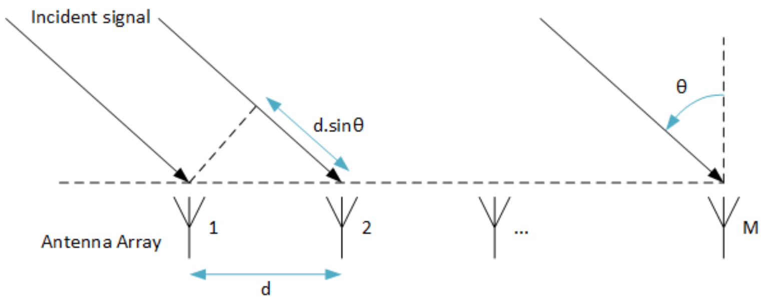


Figure 4: Schematic illustration of the structure of an antenna array. The distance d indicates the distance between each antenna and ϑ indicates the angle of arrival [5].

2.4. Signal strength

Another method developed to determine the distance between receiver and transmitter is to calculate the propagation related loss of signal. Propagation is the way waves behave in the air. Signals are sent in a specific direction and spread out over the area. The signal strength in an ideal area can be determined with the equation 6. P_0 is the signal strength at distance r_0:

$$P(r) = \frac{{r_0}^2 P_0}{r^2} \tag{6}$$

This formula provides the distance between the receiver and the transmitter. When the distance from three receivers is known, a method like the TOA can be used. The problem is that, most of the time, the signal will not be in an ideal area, and will have problems with the environment. Different factors can influence the signal. Walls, people and objects can be in the propagation line of wave. An explanation about different factors that can influence a signal will be given in the chapter 'Signal behaviour'.
The strength of a signal is expressed in dBm, even though the RSSI value is mostly used for signal strength. The difference is that RSSI is a relative index, while dBm is an absolute number representing power levels in mW (milliwatts). The RSSI value is a measurement of how well a receiver can 'hear' the transmitter and it depends on the receiver's properties which range is used to determine the RSSI value.

2.4.1. *Fingerprinting*

Fingerprinting is a specific signal strength method. This method does not calculate the length from the receiver to the transmitter but stores certain RSS values in a database and links those to a certain grid

of location points in an area. The advantage of this technique is that most of the environments' interference is stored in the database. All the transmitters are sending their signal over the grid. For this method to work, the entire database should be mapped manually (calibration of offline phase). When the database is filled with data points, the system can go into online phase. These data points are called fingerprints. There are multiple algorithms available that can calculate the location [6]. For a normal triangulation method, three transmitters are used. The main advantage of fingerprinting is that more transmitters can be used.
A popular and frequently used algorithm for an IPS is the k-nearest-neighbour (kNN) algorithm. The kNN algorithm is an algorithm that selects the nearest fingerprint around a device to determine its own location. When the algorithm can determine the nearest fingerprint, it can determine with a small error its location within the grid. The higher the amount of stored fingerprints, the more precise the location becomes. The Euclidean distance or Pythagorean metric can be used to compare every fingerprint with a measured value within the database:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{7}$$

When the closest fingerprint is found, the location linked to that fingerprint can be requested from the database [7].
There are some negative sides to a fingerprinting system. Filling this database is an expensive and time consuming step because of the calibration. It also means that the manufacturer of the system needs to enter every area in the building. This can be a problem if there are some private rooms or when the site is very large. A great advantage though, is that the exact location of all transmitters is unimportant. Once the system is operational, filling a new database is all that is required to set up a new tracking environment.

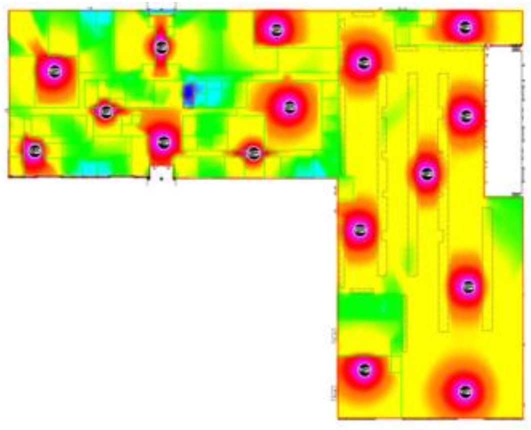


Figure 5: A example of how signals travel through a building. The blue dots represent the transmitters. The further away from the transmitters the weaker the signal becomes. Red represents the strongest signal and blue the weakest signal. The RSSI value from different transmitters are collected at different locations and stored in a database [8].

2.5. Signal behaviour

A signal travels through the environment it is broadcasted in. Because of influences from the environment, changes will be made to the signal. All those influences interfere with the accuracy and precision of the measurements that will be done to determine the correct value [9]. A few different behaviours that can occur are described below.

2.5.1. Attenuation

Attenuation is described as the decrease of signal strength, see Figure 6. This phenomenon occurs when the amplitude of a signal decreases over its propagation. A signal can lose its strength when traveling through a material (air, metal…). Attenuation is a natural behaviour of signals and occurs because of the following reasons: absorption and the negative effects of the multipath effect [9].

Attenuation

Figure 6: Schematic of the attenuation principle. The amplitude of a signal decreases over time.

2.5.2. Absorption

One of the most common radio signal behaviour is absorption. When a signal passes through an object, absorption occurs, see Figure 7. Materials will absorb some amount of the radio signal. Absorption can be a leading cause of attenuation. Larger objects with high water content are a challenge for signals within buildings since they result in high absorption. Paper, cardboard, people and so forth can absorb significant amounts of signal [9].

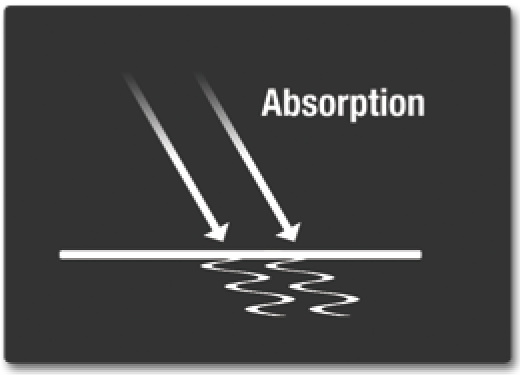


Figure 7: Schematic of the principle of absorption. The undulating lines indicate where the signal was absorbed into a materials [10]*. The horizontal line indicates the change in material.*

2.5.3. Reflection

When a wave encounters a surface, that is greater than the wavelength itself, the wave may bounce into another direction. This signal behaviour is called reflection, see Figure 8. In an indoor environment signals reflect from surfaces like wall and doors. Metal and glass are greatly known for their reflecting properties. Whether or not a wave reflects, depends on the physical properties of both surroundings (the present and the encountered e.g.: the air and the wood of the door) and the angle at which the wave hits said surface [9].

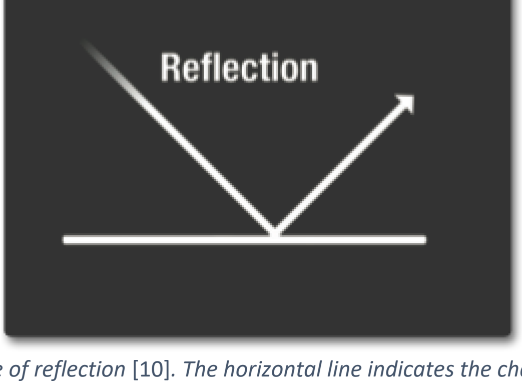


Figure 8: Schematic of the principle of reflection [10]*. The horizontal line indicates the change in material.*

2.5.4. Scattering

Scattering is a behaviour that is caused by signal propagation and multiple reflections, see Figure 9. The presence of multiple reflections means that instead of one concentrated ray of signal, that ray splits into several less concentrated ones with different directions. These multiple reflections occur when the signal's wavelength is larger than the material the signal encounters [9].

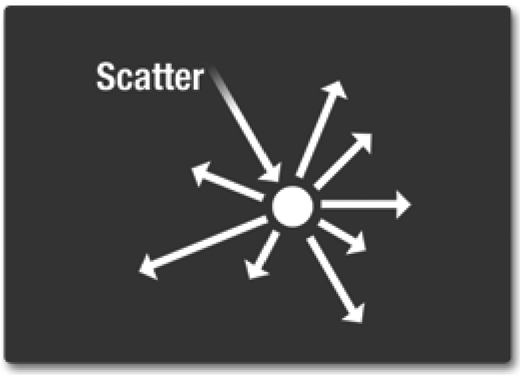


Figure 9: Schematic of the principle of Scattering. A signal encounters an object (dot) and scatters in all directions [10].

2.5.5. Refraction

If the right conditions are met, a signal can actually bend to another, new direction. This phenomenon is called refraction, see Figure 10. An example is when light passes from air to water the direction the angle from input is different than the angle of output [9]. This example is very well known as the phenomenon that seems to 'bend' or 'cut' a straw at the surface of water.

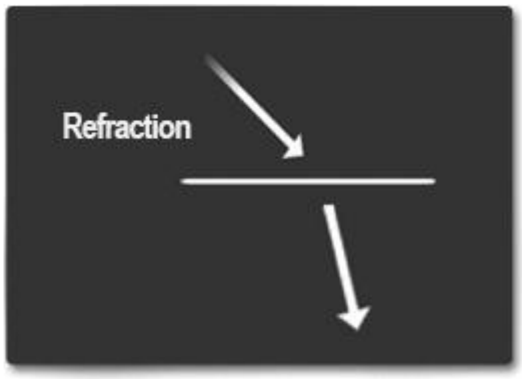


Figure 10: Schematic of the principle of refraction. When encountering a new material, the signal bends to another, new direction. The horizontal line indicates the change in material [10].

2.5.6. Diffraction

Diffraction is when the signal is bent. It occurs when a signal encounters an object, see Figure 11. The difference with refraction is that with refraction, the signal passes through the object and with diffraction, the signal travels around the object. There are multiple conditions to be met before diffraction will occur. Those conditions are that the object must have a certain shape, size and material that meet the characteristics of the signal (polarization, phase and amplitude) [9].

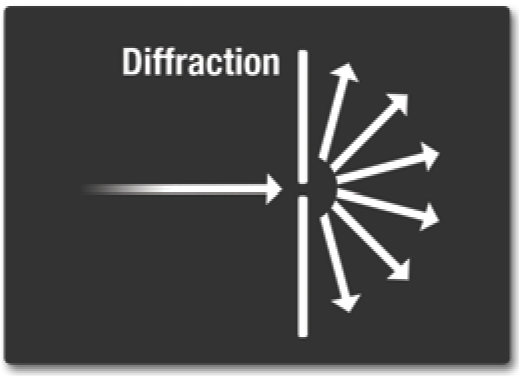


Figure 11: Schematic of the principle of diffraction, the signal encounters an object and travels around the object [10]. *The vertical interrupted line represents an obstacle that causes the signal to bend.*

2.5.7. Multipath

As explained before, signals do not move directly from the transmitter to the receiver. Because of the above mentioned behaviours, signals move almost randomly through the room. This behaviour can lead to the multipath effect. Multipath is a phenomenon that results in two or more path of signals arriving at the receiver simultaneously or within nanoseconds of each other, see Figure 12. The receiver cannot determine which signal travelled the shortest way from transmitter to receiver. This incorrect signals can interfere with correct measurements [9].

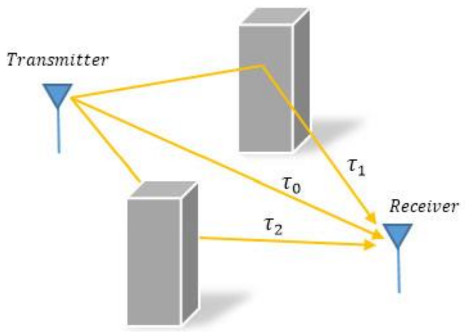


Figure 12: Schematic illustration of the multipath effect. Two or more signals originating from the same transmitter arrive simultaneously at the same receiver. Both waves may have travelled different paths. [11].

All these behaviours can affect the accuracy of the measurements. To determine the correct measurement, a filter must be applied to the incoming signal.

2.6. Filter

As briefly mentioned in the previous chapter, filters are a way to clear a signal of its interference. A physical filter is used in several areas to remove particles you do not want in your substance. Applying a filter to a signal will have the same effect. There is a wide range of filters that can reduce the signal noise. The filter that could be implemented for an indoor positioning system is described below.

The Kalman filter is an algorithm that uses a series of measurements over time. It is used to get rid of the noise within the measurements. When a Kalman filter is used, the result is more precise than one single value measurement. It uses a Bayesian inference and estimates a joint probability distribution over time for the variables over multiple timeframes. The Kalman filter is an iterative mathematical process that uses a set of equations and a consecutive data input. It continues to receive new data and keep calculation a new estimate. The advantage of using a Kalman filter in comparison to e.g. taking an average is that the true value can quickly be determined and there is no need to wait for a large set of data points [12].

2.7. Wireless communication technologies

In the previous chapters, different techniques for signal interpretation were given. In this chapter the different transmitters are described. The properties of the different wireless communication technologies are summed up in Table 1.

2.7.1. GPS based

Global Positioning System (GPS) is one of the most successful positioning systems for outdoor environments. Nevertheless, GPS has poor coverage of satellite signals in indoor environments. This decreases its accuracy and makes the method unsuitable for indoor location estimations. Although GPS is not useful for indoor location tracking there are possibilities where a GPS solution can be used for tracking. With assisted GPS (A-GPS) some limitations of GPS can be overcome. A-GPS a is a GPS based method that was enhanced by adding another tracking method to increase the accuracy. With A-GPS, some companies were able to determine the location of an object within 5-50 meters. The accuracy depended on the different indoor environments. They were able to combine the GPS signals with the signals from a mobile station. The wireless handset collects measurements from both signals and combines them to a possible location [1].

2.7.2. RFID

Radio Frequency Identification (RFID) is a system that can store and retrieve data through electromagnetic transmission. RFID systems have several basic components: a RFID reader, RFID tags and the communication between them. A reader is able to read the data emitted by the tag. There is a difference in protocol to transmit or receive data. RFID tags can be categorized into active and passive tags.
A RFID tag is passive when it operates without a battery. They are mainly used to replace the traditional barcode. Passive tags are small, light and not very expensive compared to the active tags.
They operate as follows: the receivers broadcast a signal and passive tags reflect that signal with additional information. This information can be unique for every tag. However, the range is limited at around 1-2 meters. Besides, the cost of the readers is relatively high.
Active RFID tags operate with a battery and are small transceivers. They can actively transmit their ID (or any other additional data) in reply to an interrogator. The main advantage of active tags in comparison to passive tags is that a small antenna can be placed on the tag to greatly increase the range by up to tens of meters. In practice, active tags are being used for high-unit-value products moving through a harsh assembly process. Both active and passive RFID system are based on a received RSS value to determine the distance from the receiver [1], [13].

2.7.3. Cellular based

As mentioned before (A-GPS), an easy method to determine an outdoor location for mobile clients is with the mobile network. This can also work as a standalone solution. Another name for this method is cell-ID, but the precision is very low. Depending on the cell size the range can spread between 50 until 200 meters. In more urban areas the precision is higher because the higher amount of cell towers. Indoor positioning based on a mobile cellular network is possible, if the building is covered by several base stations or one base station with strong RSS which can be received by indoor mobile clients.
A possible solution for localization is the use of wide signal-strength fingerprints. A system has a wide fingerprint database from for example six strong GSM cells. The advantage is that it can detect signals that are too weak to have a stable connection for a phone call, but that are strong enough to use in a fingerprint database. Inside a building with multiple stories it was able to track an object with an accuracy of 2.5 meters. The algorithm mostly used in such a context is the kNN algorithm. In theory the same method can be applied to the 3G mobile network [1], [9]

2.7.4. UWB

UWB stands for Ultra-Wide Band. UWB is based on sending ultrashort pulses. Normally those pulses are less than one nanosecond long. UWB location determining has multiple advantages. In comparison to RFID systems which operate on single band of the radio spectrum, UWB transmits a signal over multiple bands of frequencies at the same time. The range of 3.1 to 10.6 GHz decreases the chance of interference. UWB signals are also transmitted for a much shorter duration than those used in conventional RFID. Also, UWB tags use less power in comparison to RFID tags. Another great advantage is that UWB can be used in close proximity to other RF signals without causing or suffering from interference because of the differences in signal types and radio spectrum. Because UWB has short pulses is it easier to filter which signals are correct and which are generated because of multipath. Moreover, UWB can easily travel through walls and equipment. Nonetheless, metallic and liquid materials can cause interference. The problem can be overcome by using multiple UWB's. Because of the short pulses, UWB allows for an accurate determination with TOA. The time synchronization of UWB communication devices allows for a very accurate indoor localization. This can be scaled down until 20 centimetres accurate. When an accurate 3D location is needed TDOA and AoA can be combined to achieve a location [1], [9].

2.7.5. WLAN

WLAN stands for wireless local area network. It operates on a 2.4-GHz band. WLAN has become very popular for hotspots and home- and industrial networks. The typical range for a WLAN signal depends on the transmitter but varies at around 50-100 meters. Its official name is IEEE 802.11 and it is currently the dominant local wireless network. The popular brand name is Wi-Fi. Because of these properties, it is appealing to use an existing WLAN network for an indoor positioning system. Since in most environments a WLAN network is already available, only a server should be added to the network.
A WLAN networks works as follows: an access point transmits an available signal with information about the transmitter itself (e.g. its SSID and MAC address). A receiver can detect this signal and choose to make a stable connection. To determine the signal strength, a stable connection is not required.
The most common technique when using WLAN for determining a location is with the use of RSSI. The accuracy can then be determined at 3 to 30 meter [1], [9].

2.7.6. Bluetooth

Bluetooth operates on the 2.4 GHZ ISM band. With a typical range of 10-15 meters, the range is shorter than that of WLAN. Despite that, Bluetooth a highly used technique. It is implemented in most mobile phones and other mobile devices. Bluetooth tags are small size transceivers and any Bluetooth device has a unique ID. This ID can be used for locating the Bluetooth tag. A Bluetooth location determining system is based on three types of elements: the positioning server, wireless access points and wireless tags. With Bluetooth it is possible to track people or assets within an accuracy of 2 meters. Bluetooth also uses RSSI values [1], [9].

2.7.7. INS

Inertial Navigation Systems are some of the most widely used dead-reckoning systems. They can provide continuous position, velocity and also orientation estimates which are accurate for a short term. INS are quickly subject to drift due to noise of the sensors. Because of this, filtering is important in an INS. The Kalman filter is widely used in GPS and INS applications to reduce the noise effect on the measurements. The exact accuracy of INS system cannot be given, because the error margin increases over time. For the use of an INS, only an exact starting position is needed after which the new position can be determined with accelerometers and gyroscopes. Since the system needs to be readjusted over time, INS is not useful as a standalone indoor positioning system, but it can be useful in combination with another indoor position system [14].

Table 1: Overview of the different transmitter techniques and their respective ranges [1], [15]

Technique	Range (m)	Receiver	Cost
GPS	5 – 50	Active	Medium
RFID	1.5 – 2	Active/passive	Low
Cellular based	50 – 200	Active	Medium
UWB	0.2	Active	High
WLAN	3-5	Active	Low
Bluetooth	2	Active	Medium
INS	n/a	n/a	n/a

3. Requirement analysis

To determine the requirements, interviews were conducted with several people from different professional expertise areas. First, the different experts will be introduced. Second, the different findings from the interviews are elaborated. Third, a list sums up all the requirements.

3.1. Introduction of stakeholders

To acquire more knowledge about the requirements the system should meet, interviews were conducted. Examples of question asked during the interviews are added in Appendix A. Because the bedmover and patient lift are primarily used in two distinct environments, both user groups were consulted. For the bedmover, which is mainly used in hospitals, an interview was conducted with a former nurse at the Medisch Spectrum Twente (MST), Cindy van Goor. The MST is the main hospital in Twente. Today, she holds a managing function within the Cardiology department. Annemieke van Dijk and Robert Blokzijl were interviewed about the use of the patient lift used in retirement homes. Both are employees of LevelUpAssist (previously known as Active4Care). LevelUpAssit is a company that retails patient lifts to customers [16]. Annemieke van Dijk is the operational director and Robert Blokzijl is the commercial director.
For a better understanding of technologies used today, Jan Freerk Popma, Marcel Lamers, Roy de Jager and Frank Wopereis were also interviewed.
A global insight into the possible technologies was acquired through the answers of Jan Freerk Popma. He is a Wi-Fi specialist at the University of Twente, and is the main administrator of the Cisco Wi-Fi network at the University.
The interview with Marcel Lamers, the co-founder and CEO of Lone Rooftop [17] clarified the concept of determining locations through Wi-Fi. Lone Rooftop develops cutting-edge technologies that enhance office buildings on a technological level to improve efficiency and sustainability.
The last interview was with Roy de Jager, a security specialist at SecureLink Nederland [18] and Frank Wopereis, an application administrator for the MST with extensive knowledge about the tracing project at the MST. Roy de Jager did a project for the MST to set up the current network infrastructure within the new location on Koningsplein. Some additional information about requirements was gathered from colleagues at Indes.

3.2. Overview of interview findings

In this section, the requirements that were based on the interviews are described. All parties recognised the problem of losing equipment.
Cindy van Goor elaborated about the problems within the MST. Numerous maintenance engineers work on the different pieces of equipment. Because products get lost inside the MST, there are now 5000 devices that are past the maintenance date and still in use. The organization of the medical equipment at the MST is organized per level, with three depots per floor. Personnel can lend the equipment from a depot and are supposed to bring it back. Nevertheless, not everything is brought back. It also occurs that equipment is moved from floor to floor, disorganizing the entire structure. From her point of view, solving the issue through the use of the software could increase the quality of the provided healthcare.
The experience of LevelUpAssist was more specific to the patient lifts, but they described the same problem of overdue maintenance because of misplaced appliances. As there is not an elaborated structure in the establishments of LevelUpAssist's clients, all patient lifts move through the building. The only structure retirement homes use is that in the evening, all equipment should be returned to its respective power socket.
There is also a difference between apartment and residential group based retirement homes. The residential group retirement homes are located close to each other but not necessarily in the same building. In that case, maintenance engineers have to go from one building to another. If a piece of

equipment is in another building than it is supposed to be, the maintenance engineer spends a lot of time trying to locate it.
During the interviews, an interesting finding was that, even though it was presumed that there were only two stakeholders: caregivers and maintenance engineers, managers can also benefit from the technology. Through the finding, the pool of stakeholders was broadened with the managements of both, LevelUpAssist and the retirement home. These two new stakeholders could implement the technology for acquiring knowledge about the use of the equipment. This knowledge may enable for a more efficient use and distribution of the equipment, and through the knowledge about the usage intensity provide information about the necessity (or lack thereof) of maintenance. Together, those improvements could have a positive effect on work ethics and efficacy.

3.2.1. System feedback

A feedback system was discussed with Cindy van Goor, Robert Blokzijl and Annemieke van Dijk. With their help and in order to produce a fitting system, several requirements where set up.
The system should be adapted to having three types of users: the maintenance engineers, medical personnel and both managements. Because of their different uses of the software, the different users can favour either a portable or a desktop device or both: the system should therefore be able to run on both.
The manner in which the information is displayed is crucial as well. All the stakeholders agreed that it should be an easy to master and simple interface. From the interviews, three possible suggestions came forward. The first option is a 2D map with dots representing the required devices. The second a 3D model of the building again with dots to represent the devices. The last option was a list of devices linked with room numbers. Both parties agreed that the best feedback from the system was with the display of a 2D map.
Two different kinds of search should be possible. The first possibility is mainly for the maintenance engineers, and should give them the possibility to locate specific equipment or all appliances of one type. This would enable maintenance staff to locate the equipment in need of maintenance. A second possibility would be to allow the user to find the nearest appliance by using the user's location. This should take into consideration the location of the stairwells and elevators and travelling times.
Last but not least, the system should be able to tell whether the equipment is in use. This makes sure that members of the personnel directly go to an available appliance.

3.2.2. Tracking and tracing

There are different ways to track a device. When a device is tracked continually, the system is constantly probing and sending information to the server. It is also possible to track a device only at the exact moment a request is done. Indes made the decision to go for the second option. Continuous tracking would be a heavier burden for the battery. After requesting the location, it should only be updated as long as the user wants to know the location of the device. This way, the system will save energy.
In order to keep an eye on all equipment at any time, the location of any object should be requested and stored automatically whenever the battery level drops too low, even though this does not occur often.
The management of LevelUpAssist saw purpose in saving the historical locations of their equipment. This would contradict the choice for non-continuous tracking but enable them to determine the efficacy of the usage. This could also be of interest to the management of other companies.

3.2.3. Standalone system.

A conclusion that could be drawn from the interviews with Cindy van Goor, Robert Blokzijl and Annemieke van Dijk was that tracking would only be effective if more equipment could be tracked. The variety of medical equipment in retirement homes and hospitals comes from different suppliers, so only being able to track a fraction would not allow for the technology to settle into the daily routine and forms a significant limitation. Being able to implement the tracking technology on other large and power supplied appliances would increase the overall attractiveness of the technology. The system should be standalone and it should be possible to add it to certain devices. It would also be useful to enable the use of the power supply of the equipment. Even if the system is standalone but powered by the equipment's power supply, the tracker should somehow also function when the device's power is switched off since the system could request its location at any given moment.

3.2.4. Wireless communication technology

For choosing the most adequate system, the existing network infrastructure should be taken into account. Choosing for existing technology considerably lowers the expenses related to the implementation of the system. Some smart buildings are already equipped with a Wi-Fi and/or Bluetooth network.
The MST has a full covered Wi-Fi network through the entire hospital. There are 960 access points, which makes the building fit for tracking over Wi-Fi since most places will have around fifteen different available access points.
If there is no or too little coverage in a building, it can be expensive to install or expand a network of transmitters through an entire building. When the choice to expand the existing Wi-Fi network in a building is made, it has another benefit: not only is the network infrastructure then capable of implementing an IPS, it also has a better internet coverage through it.

3.2.5. Cost constraint

A constraint from the system is the implementation cost. The implementation of the system should not be too expensive in time and money. The searching time is correlated to the accuracy of the system: the more accurate the system, the lower the searching time. This means that the degree of searching time reduction depends on the desired precision of the system. Some IPS's have a precision of up to 20 cm. The downside of those setups is the price: the implementation is expensive. Keeping in mind that the objects Indes wants to track are rather large objects, such a high precision is not required, and localization per room (accuracy of 5 meters) would be sufficient. A precision of this magnitude would imply that a localisation could occasionally be off by one room. Nonetheless, searching times would be greatly reduced. Unfortunately, errors of the same magnitude in vertical direction are a bigger issue. This means that the system should avoid these errors at all costs.

3.3. List of requirements

The final requirements are written in a complete list, is presented below. The requirements are sorted according to the MoSCoW principle. For a viable product these requirements should be met.

3.3.1. Must-requirements

- **The system must reduce searching times was opposed to manual search.**
  Using the system should have an important positive effect on the time medical and maintenance personnel spend on looking for the equipment.
- **The system must be combinable with two Indes devices: the bedmover and patient lift.**
  The first prototype must be developed to track the bedmover and the patient lift.
- **The system must be implementable in two environments (hospitals and retirement homes) as well as in different kinds of setups (premises consisting of one or several buildings).**
  Because the system is supposed be used in multiple environments it must be developed in such a way that it can be used in different environments.
- **The system must be intuitive.**
  This requirement is added for the ease of use of the system. It must be easy to use and easy to learn.
- **The system must be able to find a specific product by entering its ID.**
  For maintenance engineers this is a must. In practice it happens often that they cannot find one specific device, resulting in lacking maintenance.
- **The system must be able to search for products close to the user.**

This is a requirement for the medical personnel. They need to find the available device closest to them.

- **The system must be standalone and applicable on multiple devices.**
  Multiple users are going to use the system. They must be able to simultaneously send requests to the server. The feedback must be accessible on different platforms (computer, smartphone, tablet)
- **The system must determine a location with an accuracy of 10 meters or less.**
  The error of the given feedback must not be larger than 10 meters.

3.3.2. Should-requirements

- **The system should change work ethics and be a marketable product that enables everybody to use the equipment.**
  To acquire the maximum result from the implementation of the system, the work ethics should be changed.
- **The system should be accessible for all personnel on the work floor for whom it is useful to use.**
  This requirement focuses on which personnel should use the system. The more personnel have access to use the system, the more the total searching time is reduced.
- **The system should also be accessible for the management of the institutions.**
  If the management has access to the data of the usage of the equipment, it could help in improving the work ethics.
- **The costs of the implementation of the system should be profitable.**
  If the costs for implementation are too high, it is not worth to invest in the system.
- **The system should be given on a 2D map of the building floor.**
  According to the different stakeholders this is the easiest way to interpret the given information from the system.
- **The system should only search for a specific piece of equipment when it is asked to do so.**
  To keep the system active, power is required. To save power, the system should only be active when needed.
- **The system should keep the error in the vertical direction to a minimum.**
  An error margin with one room difference still greatly reduces the searching times. When the mistake in the vertical direction (incorrect floor) is made to often, the system would not be effective.
- **If possible, the system should use of an already existing wireless communication technology as a network environment for an IPS.**
  For cost reduction, should be considered if the available wireless communication technology could be used for an IPS.

3.3.3. Could-requirements

- **The system could be able to determine which product is in use or not.**
  This is an additional extra feature which could be implemented. If the system displays whether or not a product is in use by another care
  giver, it would improve the searching times even more.
- **The system could help by analyzing the distribution and usage of the equipment.**
  When a certain piece of equipment is used much in a certain department than the other, the system could contribute in good distribution of equipment throughout the premises.
- **The system could monitor the battery level and when the battery level drops below 10% it should automatically store the location.**
  When this precaution is taken, the system is still partly operational without power.

4. System design

In this chapter, an explanation will be given upon the different design choices. They were based on the requirements and literature research and the market availability together with information about the various implementations were also taken into consideration. This process resulted in the choice for a fingerprinting algorithm in a Wi-Fi environment [1], [3], [7], [15]. The used algorithm was based on the kNN algorithm. In Table 2, a SWOT analysis of the use of Wi-Fi for this purpose is given.

Table 2: A Strength Weaknesses Opportunities Threats (SWOT) analysis of Wi-Fi fingerprinting, taken from Justin Stook, Planning an indoor Navigation service for a smartphone with Wi-Fi fingerprinting localization, 2011 [15].

SWOT	Aspect	Remark
Strengths	Low-cost and low-entry	Widely available at affordable prices
	Able to penetrate walls where GPS fails	Effect is depleted after thick layers, such as thick concrete walls
	In available spaces, fairly good available signal strengths	Due to multi-path, good signal differentiation; up to 100 m
	Specific location fingerprints available	Coverage of entire building, if access points are well placed
Weaknesses	Susceptible to variations in signal strength over time	A recorded fingerprint cannot exactly be reproduced
	Earthbound based, requires more infrastructure	For fingerprinting, more access points are needed, unlike satellite based
	Multi-path influenced by present objects	The more objects there are, the more differentiated the fingerprints will be
	Might interfere with other appliances in the 2.4 GHz ISM	For example, Bluetooth and microwaves
	Site surveying and registering time consuming	Must be repeated for interior and movement changes, and for each building
	MAC address related – prone to changes	If system fails, or when access point fails, MAC cannot be used.
	Speed decrease with traffic	Only applicable to data transfer
Opportunities	Fingerprinting does not require geometric surveys	Time and effort can be saved on mapping.
	Fingerprinting only necessary at selected places	It is not necessary to measure every n meter; only at places with important topological meaning or at least where fingerprints are different.
Threats	Bluetooth (BT) or Ultra-Wideband (UWB) might overtake Wi-Fi for positioning	BT rather stable, UWB powerful in better catering multi-path + better range.

4.1. Choice of technique

There are several reasons why Wi-Fi is chosen as the main technology for further research and testing. The main reason is because it is a relative popular technique that is already used in modern or modernized buildings: most buildings already have an existing wireless communication technology. Literature and the interviews suggest that it is advisable to use Wi-Fi as a basic structure [3]. When a more accurate system is required, extra methods and techniques could be used in addition to the Wi-Fi basis.

Since most modern buildings already have an extensive Wi-Fi network, the necessary hardware investments are limited. When this is not the case, extra access points can be purchased, which is cheaper than an entire new wireless communication technology and it has the advantage that also increases the building's Wi-Fi coverage.

4.1.1. Fingerprinting

The choice to use the fingerprinting technique in combination with the kNN algorithm was made based on literature and current available products [1], [3], [19]. According to literature, establishing a triangulation system in an indoor environment is difficult [15]. The negative effects of signal behaviours like multipath could influence the measurements. The technique for fingerprinting is the same for each building as long as enough access points are available. After establishing an offline phase fingerprint database, no additional calculations need to be done during the installation of the system.
At present, the MST already has a Wi-Fi based indoor positioning system. Their IPS is also based on a fingerprinting method. The system was developed by Aruba networks and is named Analytics and Location Engine [19]. In the MST, all personnel have a Ascom Myco, an Android based work cell phone. All phones can be tracked by the system administrators. In the hospital, this system is used for emergency situations. On the Myco, there is a so-called 'stress button' that can be pressed once or twice. When it is pressed once, another caregiver receives a notification that a colleague needs help. The message automatically includes the location of the caregiver in need. When pressed twice, the message goes to security. The success of this method only emphasized the choice for fingerprinting.

4.1.2. Feedback on 2D map

Literature [15] and interviews suggest that the best way to display the location of an Indes object is on a 2D map. This can be done by a mobile app or a web application.
Depending on the request of the user, the correct floor map is showed. If the user requests a specific object, the system should project the location of that exact object on the floor it is located at. If e.g. a caregiver needs to know all the available objects on a specific floor, the system should answer the specific floor with all the available objects.

4.2. Advantages and disadvantages

As can be seen in the SWOT analysis in Table 2, there are multiple advantages and disadvantages of using a Wi-Fi based fingerprinting IPS. First, the main advantages will be discussed.

4.2.1. Advantages

The greatest advantage of using Wi-Fi is that most buildings already have an extensive Wi-Fi network. Most of the time, when a hospital or retirement home is interested in having an IPS, the network environment could already be sufficient for the implementation of the IPS. When it is not sufficient, the investment in a better network environment could also be beneficial for other purposes, e.g. a better coverage of the Wi-Fi signal throughout the building. This makes this system low cost and low entry. Another advantage is that Wi-Fi signals are also ideal in indoor environments. Wi-Fi signals are able to penetrate walls easily. Because of this, Wi-Fi is able to travel large distances through closed environments whereas GPS is not.
Another great advantage of using Wi-Fi based fingerprinting is that it does not require geometric surveys (no calculations are needed according to room size and distances). The idea behind the system is that it is independent of the building lay-out. When using fingerprinting, the size of the rooms and location of the AP's are not relevant. The only important fact is what the RSSI values are on that specific location. Filling the database is relatively easy job in comparison to doing geometric surveys. For doing geometric surveys, specialists are needed to install the system. This is not the case for filling a database.

4.2.2. Disadvantages

Using a Wi-Fi based fingerprinting indoor positioning system also has its disadvantages. The first disadvantage is that, with the use of the fingerprinting technique, the location cannot be determined very accurately and it will always be an estimate of the nearest fingerprint. This means that the accuracy of the system is correlated to the amount of fingerprints in the database. Filling a database with enough fingerprints is a tedious and time consuming job. The system only works if there is a recorded fingerprint close to wherever the measurement is done. It is essential to have fingerprints at all locations the equipment could possibly be at. This could be a problem, since some locations may be off limits.

Another flaw when it comes to a fingerprinting system is that the system is sensitive for changes in the environment. These changes can come from two different sources, the first being that changes in the lay out (accidental or on purpose). When a certain AP is moved from one position to another, this influences the system greatly. The same goes for when an AP breaks down. Every change in lay out influences the measurements. Additionally, changes to the building layout influence measurements. The other source comes from Wi-Fi signals being sensible to traveling through water. Human bodies exist for a great deal of water. The presence of human bodies in a room can therefore interfere with signal strength.

A large amount of AP's is needed for the implementation of this system since multiple AP's need to be visible at any location. Moreover, the 2.4 GHz wavelength is also used by microwaves and Bluetooth. When there are a lot of Bluetooth senders and receivers this could interfere with the Wi-Fi network and influence the measurements.

4.3. Architecture

To develop an IPS based on the mentioned system design in 'Choice of technique', the system must consist of multiple elements: there are three major elements in an IPS. These three elements are the tag, server and network environment. Most elements consist of subsystems which operate on different major elements. In Table 3, a list of the three main elements and the different subsystem is visible. For a Wi-Fi fingerprinting IPS with feedback on a 2D map, the components in this paragraph must be developed. An overview of all the different subsystems and their relation between each other can be found in Figure 13.

As can be read above, the system has an offline and an online phase. During the offline phase, the database is filled with fingerprints and during the online phase, the measured points are compared to the database. Thus, an IPS consists of two different programs: one to fill the database and one to compare and determine the location. The database filler application is an addition to the normal software package to fill the database. A summary of all elements in the system is given in Table 4.

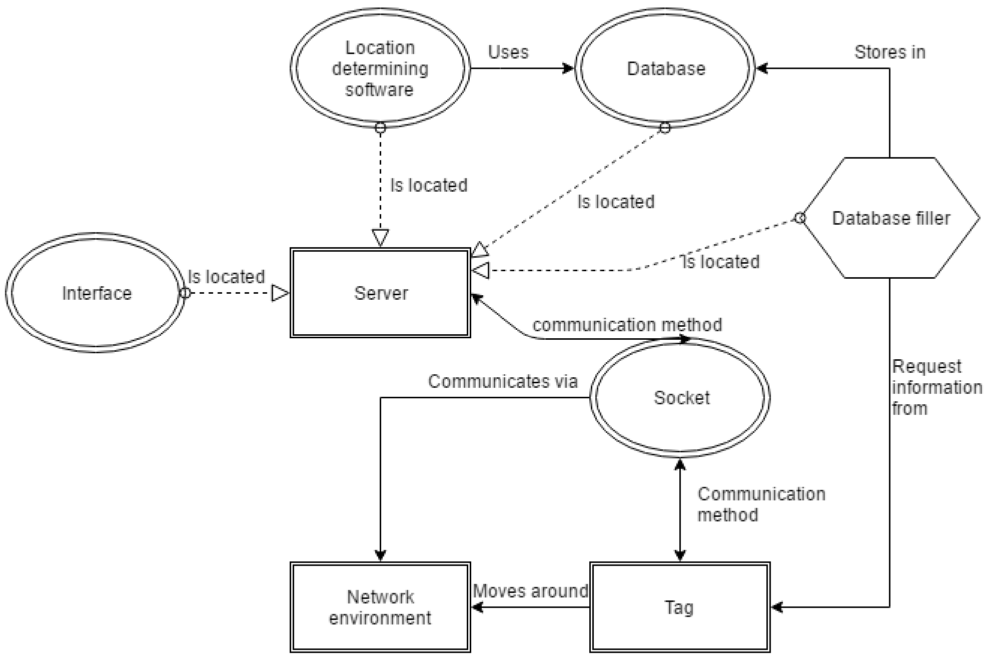


Figure 13: Flow chart of the different subsystems and how the different subsystems are related to each other. The dotted arrows represent the location at which the subsystem is based.

Table 3: Overview of the different major elements of the system and which elements use which subsystem.

Major element	**Subsystem**
Server	Socket, Location determining software, Database, Database filler, Interface
Tag	Socket, Database filler
Network environment	Socket

4.3.1. Server

The server is the main controller of the system. The server is responsible for linking all the different parts of the system together. Different users can connect to the server and send requests to the server about the different locations they want the server to determine. The server is stable and secure.

4.3.2. Tag

The tag is the physical object that gets tracked. A tag can be placed on the specified object that needs to be tracked. A tag consists of a piece of hardware with a Wi-Fi chip. The Wi-Fi chip enables the tag to connect to the network. The purpose of a tag in the system is to collect the RSSI values, MAC addresses, and SSID that surround it and send the information to the server over the socket connection. Every tag in the system is unique. Tags in a Wi-Fi environment will require an external power source.

4.3.1. Network environment

For the tag to gather information about its location, it will require information from the network environment (the network in which the tag moves around). The network consists of access points (AP). Every AP has a unique MAC address that the tag is able to record. The AP's are placed logically throughout the building, so that there is enough spatial coverage. Multiple AP's should be visible for the tag at every position in the building.

4.3.2. Socket

The socket is the connection that is established between the tag and the server. The connection can be made via the existing network using Wi-Fi. Information that was collected by the tag must be sent through the socket to the server.

4.3.3. Location determining software

The location determining software is an algorithm which is responsible for determining the best match between the measured values and the values in the database. Most of the time, an algorithm is the heart of the source code that determines the conclusion. This algorithm compares the MAC addresses in the database to the MAC addresses it got from the tag. When a matching MAC address is found, the RSSI values of those MAC addresses can be compared and a distance can be concluded.

4.3.4. Database and database filler

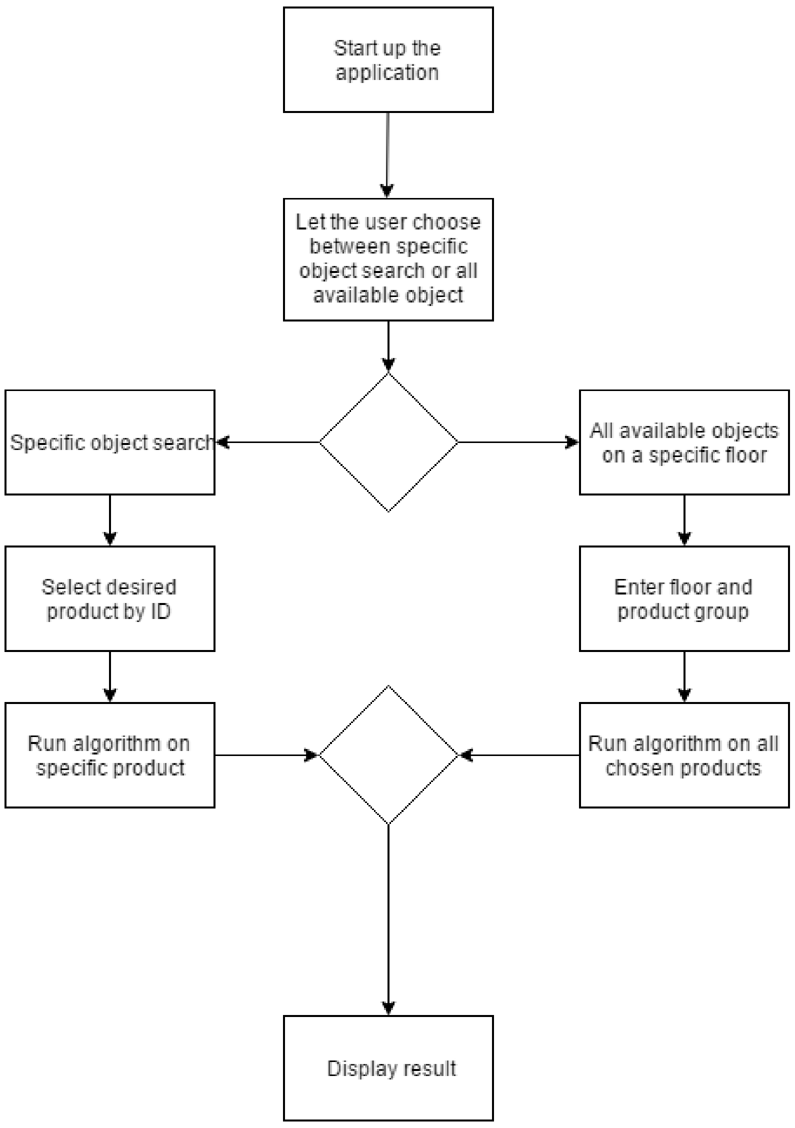
The database is an important part of the system. The database is responsible for storing the unique fingerprints. A fingerprint is a specific location within a building. It consists of the x and y coordinates on a map and multiple MAC addresses and RSSI values on that specific location. They all makes each fingerprint unique. Because a fingerprint must be unique, an ID number should be added.
To fill the database, a separate piece of software (the database filler) is needed. The database filler is responsible for easily adding MAC addresses and RSSI values to the database. While executing the program, the user should be able to easily add fingerprints with the specific data to the database: the program should generate a random unique number and add that to the specific database. Hereafter, the user should be able to add the coordinates of the fingerprint on the map by selecting the location.

4.3.5. Interface

The system has an interface that consists of multiple screens. When the user starts up the application, the first step the user can perform is to choose between whether they want to search for a specific object or if they want to see all available objects on a specific floor. If the user decides to choose for a specific object, a next screen should display a list of all products and their ID's. When the input is given, the server should search for that specific object and the system should determine the location. The location could be given on the matching floor map.
If the user decides to search for all objects on a specific floor, the system should give another screen. Now, the system should give the user the opportunity to select the floor they want to visualize. Again, the server should collect the information and determine the locations. Now the feedback should be one or several objects on the specific floor. A visualisation is provided in Figure 14.

Table 4: Summery of the different elements in the system and their purposes

Element	**Purpose**
Server	Main controller of the system
Tag	Wireless trackable object
Network environment	Hardware that emits signals that can be gathered by the tag.
Socket	Connection between the server and tag
Location determining software	Math application for calculating the best decision
Database	Place were fingerprints are stored
Database filler	Software application to fill the database
Interface	Application to communicate easily between user and server


Start up the application
Let the user choose between specific object search or all available object
Specific object search
All available objects on a specific floor
Select desired product by ID
Enter floor and product group
Run algorithm on specific product
Run algorithm on all chosen products
Display result
Figure 14: Flow chart giving the steps the user can perform when using the system to define their search.

5. Implementation

In this chapter, an explanation will be given upon the implementation of the architecture. For testing and experimenting purposes, a prototype has been created. In this chapter, a step by step explanation is given about how the prototype was set up. First the hardware of the tag is elaborated. A class diagram is given and important methods of the tag are also explained. Secondly, the same is done for the server. For creating a prototype, the steps that are mentioned in the previous chapter were followed. The abstract description in the previous chapter has been implemented in this chapter. The source code of the software package can be found in 'Appendix B'.

5.1. Tag

For the prototype it was important that an easily accessible and easily changeable piece of hardware was used, so that a lot of changes could be made to the hardware and software during the prototype phase. Therefore, a Raspberry Pi 3B was used [20]. A Raspberry Pi is a microcomputer developed for educational purposes at the University of Cambridge. The advantage of using a Raspberry Pi is that it is small and it has an 802.11n Wireless LAN chip which made Wi-Fi connection easy. It has four available USB slots. Two of those slots were attached to a keyboard and mouse. Via the HDMI cable, the Raspberry Pi was connected to a monitor. During the programming phase, it was connected to a power line. During the experiments, it was possible to power the Raspberry Pi with a powerbank. The powerbank was a solution to cover the power problem during the prototyping phase. The powerbank used was a TP-Link TL-PB10400 Powerbank 10400 mAh [21]. The powerbank had the capacity to power the Raspberry Pi for approximately 48 hours non-stop. The setup is visible in Figure 15.

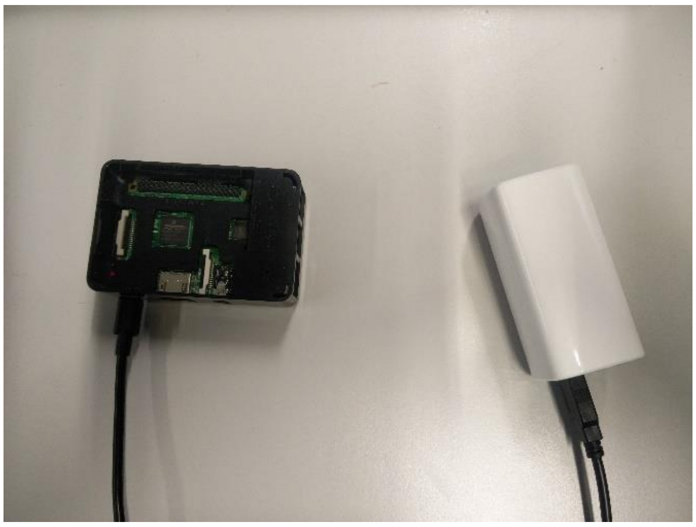


Figure 15: Photograph of the used Raspberry Pi connected to the powerbank.

For collecting the RSSI values, MAC addresses and SSID's from the available network, software had to be written. For collecting these values, an Android app was developed. Android has extensive libraries for connection management. In Android, the libraries inside *android.net.wifi* are available and these libraries are very useful for requesting the needed information for connection management. Android applications are written in at least two languages: Extensible Markup Language (XML) for lay out and interface, and Java for the functionality. For the prototype on the Raspberry Pi, no interface is needed, because it is only used as mobile tag in the system. When creating an Android app, the output has an .apk extension. Normally, a Raspberry Pi runs a Linux OS. In this case a special version of Android Marshmallow 6.0.1 based on a tablet interface was loaded on the Raspberry Pi. This way it was possible to run the .apk files on the Raspberry Pi. The operating system of a Raspberry Pi runs on an external Kingston 32 GB microSD card [22]. Win32 Disk Imager successfully flashed an Android Marshmallow .img on the microSD card. The app was developed in the IDE Android Studio. It was possible to push software updates over a wireless local network to the Raspberry Pi with the use of the Android Debug Bridge (ADB).

The software on the Raspberry Pi is built up into three main parts: a struct, a Wi-Fi Manager and a socket. A class diagram is given in Figure 16. All the classes used on the tag are visible. The class *WifiListActivity* is the main class responsible for collecting and sending the values to the server and the class AP is a struct which holds the three important values an AP has. Further in this chapter some important classes and methods will be explained further.

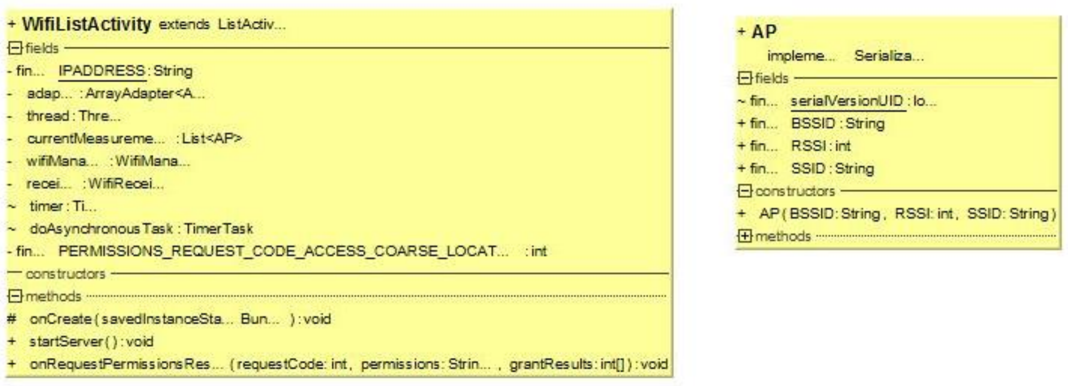

+ WifiListActivity extends ListActiv...
fields
- fin... IPADDRESS : String
- adap... : ArrayAdapter<A...
- thread : Thre...
- currentMeasureme... : List<AP>
- wifiMana... : WifiMana...
- recei... : WifiRecei...
~ timer : Ti...
~ doAsynchronousTask : TimerTask
- fin... PERMISSIONS_REQUEST_CODE_ACCESS_COARSE_LOCAT... : int
constructors
methods
# onCreate (savedInstanceSta... Bun...) : void
+ startServer ( ) : void
+ onRequestPermissionsRes... (requestCode: int , permissions: Strin... , grantResults : int[]) : void
+ AP
impleme... Serializa...
fields
~ fin... serialVersionUID : lo...
+ fin... BSSID : String
+ fin... RSSI : int
+ fin... SSID : String
constructors
+ AP (BSSID: String , RSSI: int , SSID: String)
methods

Figure 16: Class diagram of the classes from the software package which runs on the Raspberry PI.

5.1.1. Struct

A struct-like class was created. A struct is an old complex data type from the C programming language. Originally, structs are not supported in Java. However, creating one is possible by implementing a class with a constructor that passes variables. It is possible to add different groups of variables under one name in the memory. The struct was named AP and consisted out of three variables: the RSSI, SSID and BSSID. The BSSID is the basic service set identifier and is the same as the MAC address of the AP. A single access point could thus be represented with three variables. A list of AP's named *currentMeasurements* was created. This list represented a list of all available RSSI, SSID and BSSID values at one specific location. This location is the basic of one fingerprint. The class AP is visible in Box 1.

```
public class AP implements Serializable{
    static final long serialVersionUID = 1L;
    private final String BSSID;
    private final int RSSI;
    private final String SSID;

    public AP(String BSSID, int RSSI, String SSID) {
        this.BSSID = BSSID;
        this.RSSI = RSSI;
        this.SSID = SSID;
    }
```

Box 1: Class AP, a struct based implementation of the values.

5.1.2. WifiManager

Two methods are responsible for collecting the required values. Those two methods are *onReceive* and *startServer*, visible in Box 2 and Box 3. A *WifiManger* is declared and it then requests the right information. After this, two lists are declared. While the list *results* is declared, it is filled with all available information about the current network data around the Raspberry Pi. This information is filtered and only the BSSID, RSSI and SSID are added to the list 'temp'. The next step is that the list *temp* is sorted on the highest RSSI value. The list is now ready to be sent to the server.

```
        @Override
        public void onReceive(Context context, Intent intent) {
            WifiManager manager = (WifiManager)
context.getSystemService(Context.WIFI_SERVICE);
            Log.v("Tag", "Scan results received");
            List<ScanResult> results = manager.getScanResults();
            List<AP> temp = new ArrayList<>();
            for (ScanResult res : results) {
                AP ap = new AP(res.BSSID, res.level, res.SSID);
                temp.add(ap);
            }
            Collections.sort(temp, new Comparator<AP>() {
                @Override
                public int compare(AP ap, AP t1) {
                    return t1.RSSI - ap.RSSI;
                }
            });
            Log.d("Lijst" ,Integer.valueOf(temp.size()).toString());
            currentMeasurements.clear();
            currentMeasurements.addAll(temp);
            synchronized (thread) {
                thread.notify();
            }
            adapter.clear();
            adapter.addAll(temp);
            adapter.notifyDataSetChanged();
        }
    }
```
Box 2: Method onReceive, responsible for collecting the values

```
public void startServer(){
        try {

            WifiManager wm = (WifiManager)
getSystemService(WIFI_SERVICE);
            String ip =
Formatter.formatIpAddress(wm.getConnectionInfo().getIpAddress());
            InetAddress address = InetAddress.getByName(ip);

            ServerSocket serverSocket = new ServerSocket(4444, 50,
address);
            while (true) {
                Socket skt = serverSocket.accept();
                registerReceiver(receiver, new IntentFilter(
                        WiFiManager.SCAN_RESULTS_AVAILABLE_ACTION));
                WifiManager.startScan();
                synchronized (thread) {
                    if (currentMeasurements.size() == 0) {
                        thread.wait();
                    }
                }
                ObjectOutputStream objectOutputStream = new
ObjectOutputStream(skt.getOutputStream());
                objectOutputStream.writeObject(currentMeasurements);
                objectOutputStream.flush();
                skt.close();
            }
            }catch(IOException e){
                e.printStackTrace();
            }catch(InterruptedException e){
                e.printStackTrace();
        }
    }
```
Box 3: Method startServer, responsible for sending the values

5.1.3. Socket

The last part of the system is the socket. The implementation of the socket can be found in Box 3. When a series of AP's is collected at a specific location, it should be sent to the main server. To make the implementation more feasible, it was decided to declare a *ServerSocket* on the Raspberry Pi and a *Socket* on the server.

The setup is as follows: every Raspberry Pi has a *ServerSocket* awaiting a connection request from the computer. The computer requests a connection over the Wi-Fi network to the Raspberry Pi via a socket. The request can be done by entering the specific IP-Address and port the Raspberry Pi is listening on. The Raspberry Pi waits until a connection request is done. As soon as a connection request arrives at the Raspberry Pi, it accepts the connection and starts scanning. The results are then sent over the socket to the server. The method used to send the list over the network is the *objectOutputStream* method. With this method, it is possible to send entire lists over the socket as an object. As soon as the list is successfully sent to the server, the Raspberry Pi closes the socket connection and starts waiting for a new request.

5.2. Server

The server is the main controller of the system. A server is normally a stable and strong computer that is able to handle all requests of the system. In this case, a Dell Precision M2800 (RAM: 8GB DDR3; CPU: i7-4710MQ 2.50GHz; OS: Windows 10 Pro 64-bit) [23] was used to run the software on the server side. The software was written in Java with JDK 1.7 in Eclipse Mars.1 [24]. The location information provided by the tag is interpreted on the server. The server performs the following actions (1) it asks the user for their choice (a choice the user can make is which search they want to perform: the location of a nearby device or of a specific device), (2) it requests the RSSI, SSID, BSSID values from that tag, (3) it filters the collected information it received, (4) it compares the filtered information with the database information, (5) it determines the nearest location and (6) it plots the result on a map. A flow chart of the steps is given in Figure 17.

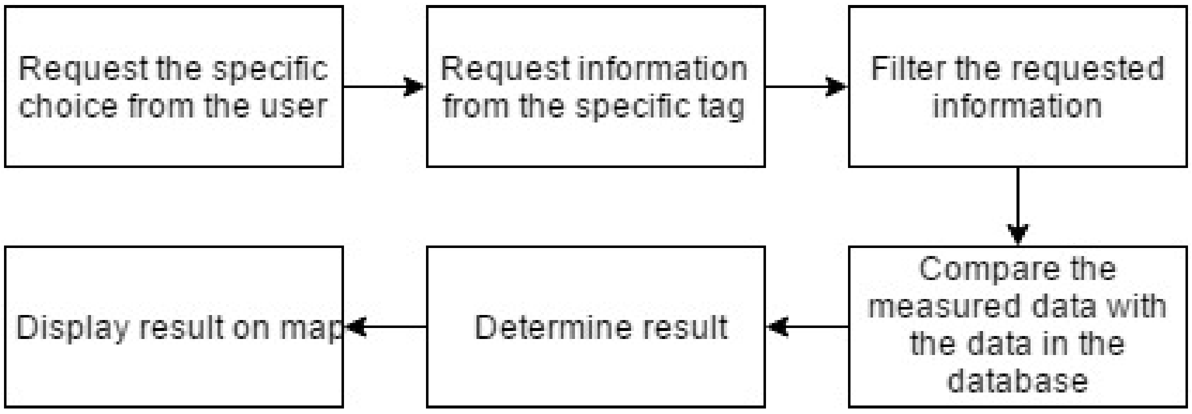

Request the specific choice from the user
Request information from the specific tag
Filter the requested information
Compare the measured data with the data in the database
Determine result
Display result on map

Figure 17: Flow chart of all the steps the server performs once a search is initiated.

The system on the server side is built up from different classes and methods. Every method or class is responsible for a different part of the system. In Figure 18, a class diagram with all the different classes and methods on the server side is given. In Table 5, all different classes are summarized and a brief explanation is given as to what the responsibility of every class is.

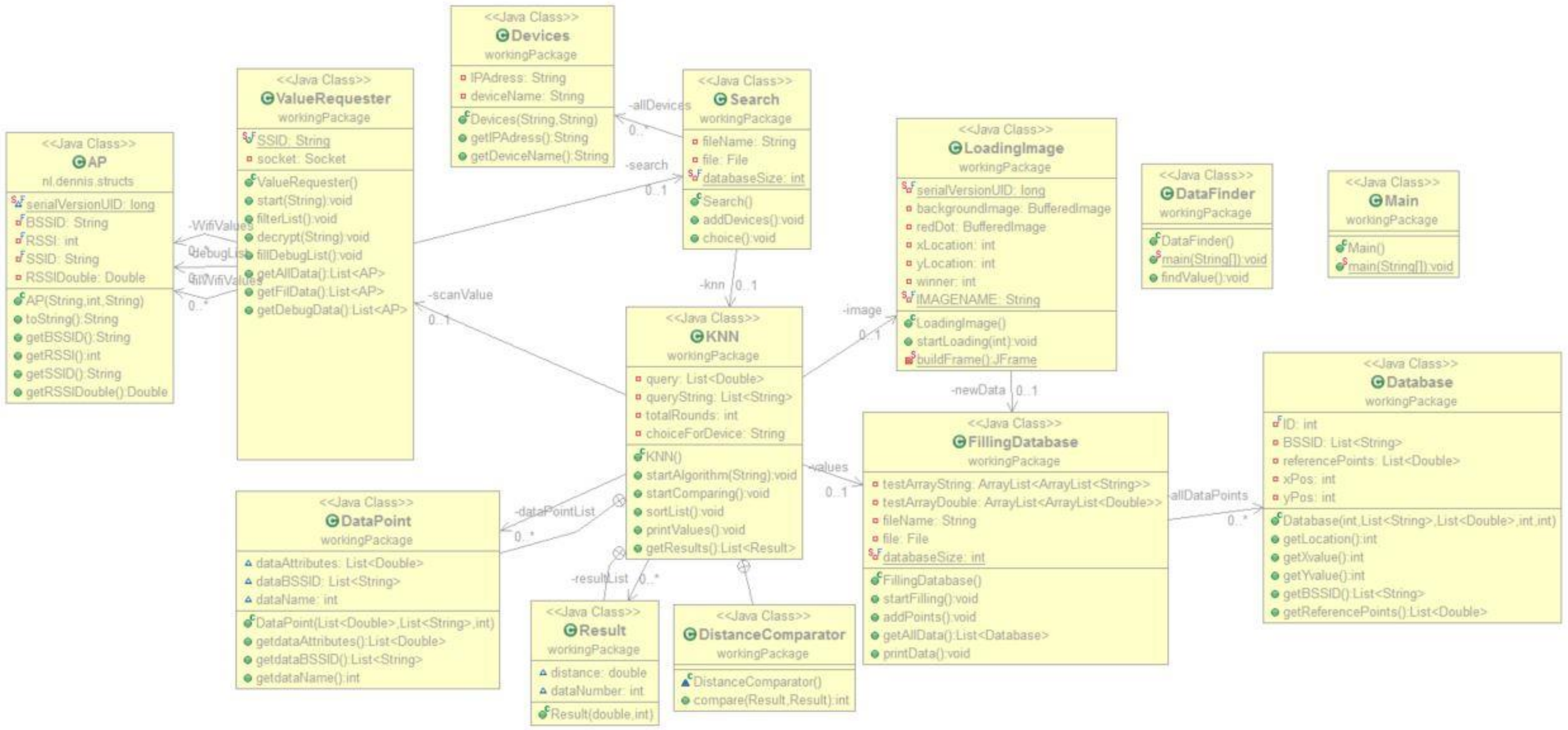


Figure 18: Class diagram generated by Eclipse of all the classes and their relations on the server side.

Table 5: Summary of all classes in the program on the server side. For every class a small explanation is given about their role in the system.

Class	Responsibility
Main	The first class that starts the program, to determine a location.
DataFinder	The first class that start the program when a new fingerprint must be added to the database. When this class is run a new fingerprint is added to the database.
Search	The class responsible for the search based on the users input. It prompts the user to select the desired device(s).
ValueRequester	ValueRequester is responsible for collecting the values from the selected device. It uses a socket connection to connect to the device and collects the sent data from the tag.
KNN	The main part of the program. This class is responsible for comparing the measured values with all fingerprints in the database.
FillingDatabase	This class collects all the information for fingerprints from a .csv file and loads them into the program.
LoadingImage	The class responsible for displaying a map of the building and a dot at the nearest location.
AP	A struct-like class used on both the server and Raspberry Pi to store the important information of one AP.
Database	A struct-like class for the fingerprints.
Devices	A struct-like class for the different devices.
Result	An inner class to model the results.
DistanceComperator	An inner class used to sort the matched fingerprints.
DataPoint	An inner struct-like class for the fingerprints that are needed for the calculation done by the KNN class.

5.2.1. Choice of information

The class Search, visible in Box 4 is responsible for registering the user's choice. First, it displays a list of all available devices within the system. In this case, the device is the Raspberry Pi. All devices can be added to the struct *Devices*. To add a device to the struct, a unique IP address and name must be given. This happens in the method *addDevices*. The system prints a list of all available devices. The user can select the specific device by pressing a number on the keyboard. The requested number is read and the correlating IP address is used to request the proper values. For time reasons, only the implementation of the specific device search was done.

```
public class Search {

ArrayList<Devices> allDevices = new ArrayList<Devices>();
KNN knn = new KNN();

Public void addDevices() {
    try {
        Scanner inputStream = new Scanner(file);
        inputStream.next();
        for (int i = 0; i < databaseSize; i++) {
            String data = inputStream.next();
            String[] values = data.split(",");
            Devices device = new Devices(values[0], values[1]);
            allDevices.add(device);
            }
        inputStream.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}

public void choice() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("What do you want to find? \n Availible are:");
    for (int i = 0; i < allDevices.size(); i++) {
        System.out.println((i + 1) + ": Name: "
+allDevices.get(i).getDeviceName()+ " IPAddress: "
+allDevices.get(i).getIPAdress());
    }
    System.out.println("Make your choice?");
    int choice = scanner.nextInt();
    knn.startAlgorithm(allDevices.get(choice - 1).getIPAdress());
    scanner.close();
}
}
```

Box 4: Class Search responsible for collecting the user input

5.2.2. Request and process values

The next step of the system is that, when a specific tag is chosen, the information from that tag needs to be requested. The *start* and *filterList* method, visible in Box 5 are responsible for requesting and sorting the values that are sent over the socket. The socket on this side tries to connect to the selected IP address through port 4444. An *objectInputStream* that can receive an object over the socket connection is created. As soon as a connection is established, the Raspberry Pi sends its list of AP's over the socket. This list is sent as an object. By casting the object to a list of AP's, it can be stored in *WifiValues*. *WifiValues* is a list of AP's, which are built up the same way as the struct on the Raspberry Pi. The method *filterList* is responsible for filtering all the collected data and only keep the data with a specific SSID.

```
public void start() {
try {
    InetAddress adress = InetAddress.getByName(IPADRESS);
    socket = new Socket(adress, 4444);
    ObjectInputStream objectInput = new
ObjectInputStream(socket.getInputStream());
    try {
        Object object = objectInput.readObject();
        //Cast them to a list;
        WifiValues = (List<AP>) object;
    } catch (ClassNotFoundException e) {
        e.printStackTrace();
    }
} catch (UnknownHostException e) {
    e.printStackTrace();
} catch (IOException e) {
    e.printStackTrace();
try {
    socket.close();
} catch (IOException e) {
    e.printStackTrace();
}
filterList();
}

public void filterList() {
for (AP value : WifiValues) {
    if (value.SSID.equals(SSID)) {
        filWifiValues.add(value);
    }
}
```

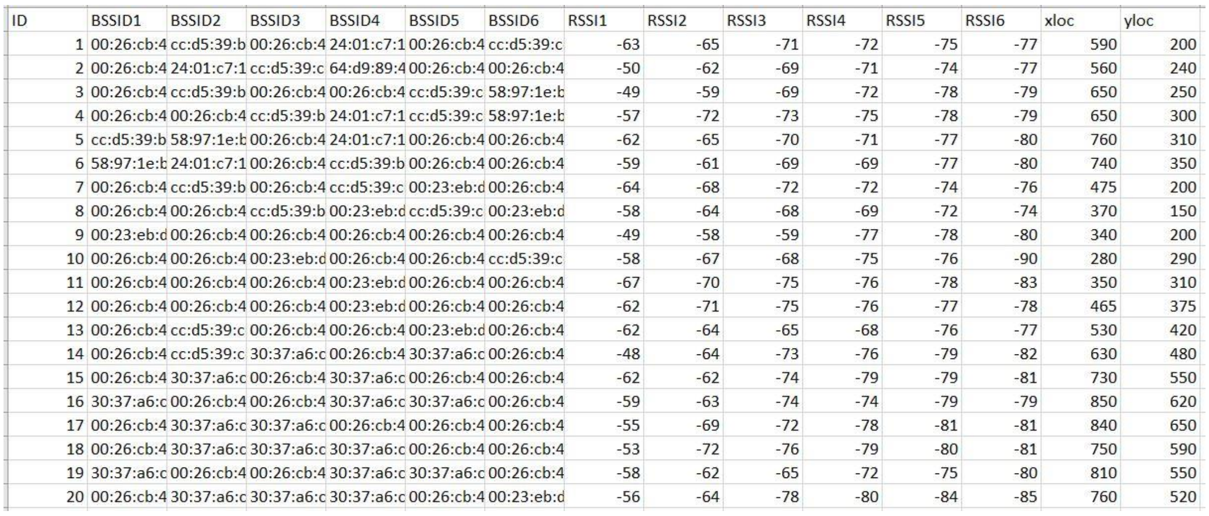
Box 5: Methods Start and filterList responsible for receiving and filtering the data

5.2.3. Database and Database Filler

The database is located on the server and stores all the unique fingerprints. The same structure as the AP's was used. The class Database is also based on the struct principle. In Box 6, the class database is given. This struct contains the following data: a unique ID, a list of BSSID, a list of *referencePoints* (RSSI values) and a set of x and y coordinates. An example of the database can be found in Table 6. A .csv file is created in Microsoft Excel. The BSSID and RSSI values of each fingerprint are automatically added to the file when the *databaseFiller* is run. The coordinates are then added manually to the same file. The struct is filled with information from the file: when the program is executed, the data from the .csv file are read by the method *addPoints*, visible in Box 7. A .csv file is a comma separated values file that can easily be imported. Every line of the file is read and put into the string data. This string is split into different parts on the delimiter ",". This new array string can now add the correct values to the corresponding parts in the struct. The RSSI first need to be casted from a String to a double, before they can be properly added to the struct. In the end, every fingerprint is added to the list *allDataPoints*.

```
public class Database {
private final int ID;
private List<String> BSSID = new ArrayList<String>();
private List<Double> referencePoints = new ArrayList<Double>();
private int xPos;
private int yPos;

public Database(int newLocation, List<String> newBSSID, List<Double>
newRefrencePoint, int newxPos, int newyPos) {
    ID = newLocation;
    BSSID = newBSSID;
    referencePoints = newRefrencePoint;
    xPos = newxPos;
    yPos = newyPos;
    }
}
```
Box 6: Class Database, the struct setup for one unique fingerprint



ID	BSSID1	BSSID2	BSSID3	BSSID4	BSSID5	BSSID6	RSSI1	RSSI2	RSSI3	RSSI4	RSSI5	RSSI6	xloc	yloc
1	00:26:cb:4	cc:d5:39:b	00:26:cb:4	24:01:c7:1	00:26:cb:4	cc:d5:39:c	-63	-65	-71	-72	-75	-77	590	200
2	00:26:cb:4	24:01:c7:1	cc:d5:39:c	64:d9:89:4	00:26:cb:4	00:26:cb:4	-50	-62	-69	-71	-74	-77	560	240
3	00:26:cb:4	cc:d5:39:b	00:26:cb:4	00:26:cb:4	cc:d5:39:c	58:97:1e:b	-49	-59	-69	-72	-78	-79	650	250
4	00:26:cb:4	00:26:cb:4	cc:d5:39:b	24:01:c7:1	cc:d5:39:c	58:97:1e:b	-57	-72	-73	-75	-78	-79	650	300
5	cc:d5:39:b	58:97:1e:b	00:26:cb:4	24:01:c7:1	00:26:cb:4	00:26:cb:4	-62	-65	-70	-71	-77	-80	760	310
6	58:97:1e:b	24:01:c7:1	00:26:cb:4	cc:d5:39:b	00:26:cb:4	00:26:cb:4	-59	-61	-69	-69	-77	-80	740	350
7	00:26:cb:4	cc:d5:39:b	00:26:cb:4	cc:d5:39:c	00:23:eb:d	00:26:cb:4	-64	-68	-72	-72	-74	-76	475	200
8	00:26:cb:4	00:26:cb:4	cc:d5:39:b	00:23:eb:d	cc:d5:39:c	00:23:eb:d	-58	-64	-68	-69	-72	-74	370	150
9	00:23:eb:d	00:26:cb:4	00:26:cb:4	00:26:cb:4	00:26:cb:4	00:26:cb:4	-49	-58	-59	-77	-78	-80	340	200
10	00:26:cb:4	00:26:cb:4	00:23:eb:d	00:26:cb:4	00:26:cb:4	cc:d5:39:c	-58	-67	-68	-75	-76	-90	280	290
11	00:26:cb:4	00:26:cb:4	00:26:cb:4	00:23:eb:d	00:26:cb:4	00:26:cb:4	-67	-70	-75	-76	-78	-83	350	310
12	00:26:cb:4	00:26:cb:4	00:26:cb:4	00:23:eb:d	00:26:cb:4	00:26:cb:4	-62	-71	-75	-76	-77	-78	465	375
13	00:26:cb:4	cc:d5:39:c	00:26:cb:4	00:26:cb:4	00:23:eb:d	00:26:cb:4	-62	-64	-65	-68	-76	-77	530	420
14	00:26:cb:4	cc:d5:39:c	30:37:a6:c	00:26:cb:4	30:37:a6:c	00:26:cb:4	-48	-64	-73	-76	-79	-82	630	480
15	00:26:cb:4	30:37:a6:c	00:26:cb:4	30:37:a6:c	00:26:cb:4	00:26:cb:4	-62	-62	-74	-79	-79	-81	730	550
16	30:37:a6:c	00:26:cb:4	00:26:cb:4	30:37:a6:c	30:37:a6:c	00:26:cb:4	-59	-63	-74	-74	-79	-79	850	620
17	00:26:cb:4	30:37:a6:c	30:37:a6:c	00:26:cb:4	00:26:cb:4	00:26:cb:4	-55	-69	-72	-78	-81	-81	840	650
18	00:26:cb:4	30:37:a6:c	30:37:a6:c	30:37:a6:c	00:26:cb:4	00:26:cb:4	-53	-72	-76	-79	-80	-81	750	590
19	30:37:a6:c	00:26:cb:4	00:26:cb:4	30:37:a6:c	30:37:a6:c	00:26:cb:4	-58	-62	-65	-72	-75	-80	810	550
20	00:26:cb:4	30:37:a6:c	30:37:a6:c	30:37:a6:c	00:26:cb:4	00:23:eb:d	-56	-64	-78	-80	-84	-85	760	520

Table 6: Example of the database. The BSSID's, RSSI's and (x, y) coordinates are given for each fingerprint.

Besides loading the stored data to the physical memory, the server is also responsible for filling the database with the fingerprints. Fingerprints are collected during the offline phase. As mentioned before, a fingerprint consists of a unique ID, a List of RSSI values and a List of BSSID values. In Box 8 the method *findValue* can be found. The method *findValue* uses the same method as *valueRequester* (Box 5). This method collects all the available information from all AP's around that position. The limit in the database for each fingerprint is six AP's. Six was chosen because this was the lowest number of AP's at multiple locations through the building. This can easily be scaled up or down if there are more or less AP's available. After *valueRequester* has been executed, the method checks if at least six datapoints are collected. If the number is lower than six, it will return that the required amount of available AP's is not met. After the check, the six strongest values are added to two ArrayList's. The RSSI values are casted from Integer to String. All the values are collected into one string and separated with a comma. A unique ID and the x and y coordinates are also added. Last, the newly collected fingerprint is written to the .csv file.

```
public void addPoints() {

try {
    Scanner inputStream = new Scanner(file);
    inputStream.next();
    for (int i = 0; i < databaseSize; i++) {
        String data = inputStream.next();
        String[] values = data.split(",");
        ArrayList<String> nSM1 = new ArrayList<String>();
        testArrayString.add(nSM1);
        ArrayList<Double> nSD1 = new ArrayList<Double>();
        testArrayDouble.add(nSD1);
        nSM1.add(values[1]);
        nSM1.add(values[2]);
        nSM1.add(values[3]);
        nSM1.add(values[4]);
        nSM1.add(values[5]);
        nSM1.add(values[6]);
        Double dRSSI1 = Double.parseDouble(values[7]);
        Double dRSSI2 = Double.parseDouble(values[8]);
        Double dRSSI3 = Double.parseDouble(values[9]);
        Double dRSSI4 = Double.parseDouble(values[10]);
        Double dRSSI5 = Double.parseDouble(values[11]);
        Double dRSSI6 = Double.parseDouble(values[12]);
        nSD1.add(dRSSI1);
        nSD1.add(dRSSI2);
        nSD1.add(dRSSI3);
        nSD1.add(dRSSI4);
        nSD1.add(dRSSI5);
        nSD1.add(dRSSI6);
        int index = Integer.parseInt(values[0]);
        int xpos = Integer.parseInt(values[13]);
        int ypos = Integer.parseInt(values[14]);
        Database database = new Database(index,
testArrayString.get(i),testArrayDouble.get(i), xpos, ypos);
        allDataPoints.add(database);
        inputStream.close();
    }
} catch (FileNotFoundException e) {
    e.printStackTrace();
}
```

Box 7: Method addPoints, responsible for loading the data from the .csv file to the memory

```
public void findValue() throws FileNotFoundException, IOException {

valueRequester.start(IPAdress);
if (valueRequester.getFilData().size() < amountOfReferencePoints) {
    System.out.println("Niet genoeg values gevonden op deze
locatie...");
} else {
    for (int i = 0; i < amountOfReferencePoints; i++) {
        BSSID.add(valueRequester.getFilData().get(i).getBSSID());
        RSSI.add(Integer.toString(valueRequester.getFilData().get(i).
getRSSI()));
    }
    StringBuilder sb = new StringBuilder();
    sb.append(ID);
    sb.append(",");
    for (int i = 0; i < BSSID.size(); i++) {
        sb.append(BSSID.get(i));
        sb.append(",");
    }
    for (int i = 0; i < RSSI.size(); i++) {
        sb.append(RSSI.get(i));
        sb.append(",");
    }
    sb.append(xLoc);
    sb.append(",");
    sb.append(yLoc);
    sb.append("\n");
    pw.write(sb.toString());
    pw.close();
}
}
```

Box 8: Method findValue, resposible for adding fingerprints to the .csv file

5.2.4. kNN algorithm

The algorithm is the heart of the program. This part is where the measured value and the values in the database are compared. This class consists of three major methods: *startAlgorithm*, *startComparing* and *sortList*. All three methods are based on an implementation of a kNN algorithm in Java [25]. The method *startAlgorithm* (in Box 9) is responsible for collecting all available data and makes sure every variable is a local variable. It first collects the list of all database points and stores it into a new struct named *dataPointList*. At that point, all values are local variables and it is not necessary anymore to use 'get' functions to request the required data. The next step is to request the measured value. It is added to two ArrayLists of RSSI values and MAC addresses. All the required values for determining the location are then available.
Next, *startComparing*, presented in Box 10, compares the values of the measurement with the database. The method first iterates through every point in the complete database. Two numbers that are needed later for calculations are declared. Next, it iterates through every MAC address for one specific fingerprint. The last step is that it iterates through every measured MAC address from the Raspberry Pi. In this three double 'for' loop, all MAC addresses are requested. If two MAC addresses are identical, both relating RSSI values are request and compared to each other. The result of this comparison is stored in the double *dist*. For every matching MAC address between one fingerprint and the measurement, the Euclidean distance is calculated. The result is added to the *resultList*. The dexterity of this method is that it first tries to compare all six MAC addresses with each other. In practice, it can occur that one or two MAC addresses cannot be determined because of e.g.

interference. When the method fails to find a match with all six AP's, it will try again with five and so on. The system can print the value of the amount of matching MAC addresses. If the value is six, one can be quite certain that the measurement is correct. If the value is lower, the measurement can get quite incorrect. If this happens often at a location, it could be useful to add more fingerprints to the database in that area.
The last method, *sortList*, is responsible for sorting the list. Two RSSI values are compared using the Euclidean distance. The closer the RSSI values are together, the closer the measured point should be to a specific fingerprint. So the higher the value of *dist*, the further away it is from that specific fingerprint. This means that the lowest value of *dist* is the closest fingerprint to the measured value. *SortList* is responsible to sort the list on *dist* values. The fingerprint that ends up highest on the list is thus the closest fingerprint. The location of this fingerprint is then displayed on the map.

```
public void startAlgorithm(String choice) {
for (int i = 0; i < values.getAllData().size(); i++) {
    dataPointList.add(new DataPoint(
    values.getAllData().get(i).getReferencePoints(),
    values.getAllData().get(i).getBSSID(),
    values.getAllData().get(i).getLocation()));
}

query = new ArrayList<Double>();
queryString = new ArrayList<String>();
for (int i = 0; i < scanValue.getFilData().size(); i++) {
    Double tempDouble;
    String tempString;
    tempDouble = scanValue.getFilData().get(i).getRSSIDouble();
    tempString = scanValue.getFilData().get(i).getBSSID();
    query.add(tempDouble);
    queryString.add(tempString);
}
}
```

Box 9: Method startAlgorithm resposible for collection all the values

```
public void startComparing() {
for (int i = 0; i < dataPointList.size(); i++) {
    double dist = Double.POSITIVE_INFINITY;
    int counter = 0;
        for (int j = 0; j < dataPointList.get(i).getdataBSSID().size();
j++) {
        for (int h = 0; h < queryString.size(); h++) {
            if (dataPointList.get(i).getdataBSSID().get(j).equals(
queryString.get(h))) {
                dist = dist == Double.POSITIVE_INFINITY ? 0.0 : dist;
                Double refRSSI =
dataPointList.get(i).getdataAttributes().get(j);
                Double measRSSI = query.get(h);
                Dist += Math.pow(refRSSI - measRSSI, 2);
                counter++;
            }
        }
}
    if (counter == totalRounds) {
        double distance = Math.sqrt(dist);
        resultList.add(new Result(distance,
dataPointList.get(i).getdataName()));
        System.out.println("Matching Mac adress: " + totalRounds);
    }
}
if (resultList.isEmpty()) {
    totalRounds--;
    startComparing();
} else {
    sortList();
}
}
```

Box 10: Method startComparing responsible. for comparing all the values in the database with the measured value

```
public void sortList() {
Collections.sort(resultList, new DistanceComparator());
List<Integer> ss = new ArrayList<Integer>();
for (int x = 0; x < resultList.size(); x++) {
    ss.add(resultList.get(x).dataNumber);
}

for (Result results : resultList) {
    System.out.println("Closest Access point: " + results.dataNumber);
    System.out.println("Total Distance: " + results.distance + "\n");
}
int winner = resultList.get(0).dataNumber;
image.startLoading(winner - 1);
}
```

Box 11: Method sortList, responsible for sorting the resultList on the lowest value from the result of the Euclidean distance

5.2.5. Feedback

A 2D map by Google Maps was chosen to visualize the feedback, since Google Maps already mapped the insides of the University of Twente buildings. The maps of the ground and first floor of the Zilverling building at the University of Twente were juxtaposed, as it can be seen in Figure 19. This class starts with the method *startLoading* which receives the fingerprint which is on top of the list from the *sortList* method. The method *startLoading* is visible in Box 12. First, it loads both images, the map of the building and a small red dot. Hereafter, it requests the x and y coordinate of the specific fingerprint. Last, the method *paintComponent* returns the map of the building and the red dot at the specific location. For illustrative purposes, Figure 20 gives a map all collected fingerprints within the database.

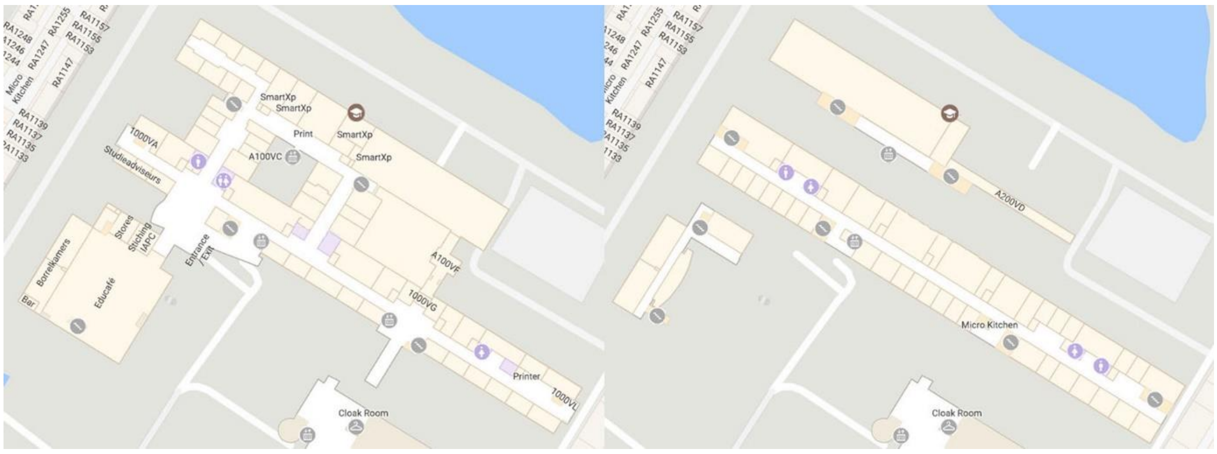


Figure 19: Map of the ground (left) and first (right) floor of the Zilverling building at the University of Twente [26].

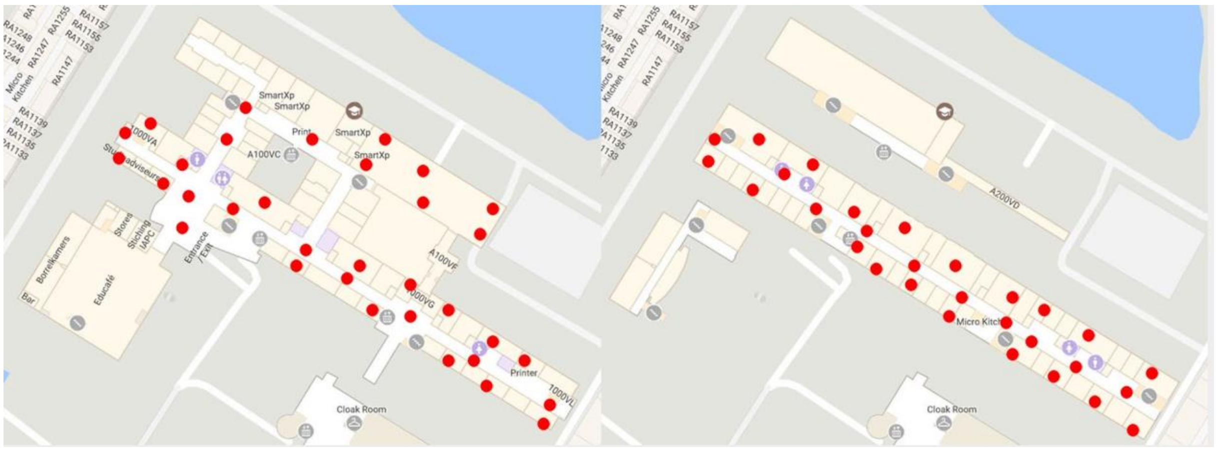


Figure 20: Map of the ground (left) and first (right) floor of the Zilverling building with all collected fingerprints (each represented by a bright red dot) The map itself is the Google Maps map [26]. *The red dots were added to map by having the program display every single fingerprint. The exact location of each fingerprint is given by its x and y coordinates from the .csv file.*

```
public void startLoading(int newWinner) {
winner = newWinner;
JFrame frame = buildFrame();
try {
    backgroundImage = ImageIO.read(new File(IMAGENAME));
    redDot = ImageIO.read(new File("reddot.png"));
} catch (IOException e) {
    e.printStackTrace();
}
xLocation = newData.getAllData().get(winner).getXvalue();
yLocation = newData.getAllData().get(winner).getYvalue();

JPanel pane = new JPanel() {
private static final long serialVersionUID = 1L;

@Override
protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(backgroundImage, 0, 0, null);
    g.drawImage(redDot, xLocation, yLocation, null);
//  for (int i = 0; i < 65; i++) {
//      g.drawImage(redDot, newData.getAllData().get(i).getXvalue(),
newData.getAllData().get(i).getYvalue(), null);
//  }
```

Box 12: Method startLoading, responsible for loading and drawing the Image and the red dot at the specific location

6. Evaluation

In this chapter, the test setup and the experimental results are explained. Two experiments were performed to determine if the prototype functioned the way it was supposed to. The goal of the first experiment was to see if a Wi-Fi fingerprinting based indoor positioning system reduces the searching time in comparison to manual search. The goal of the second experiment was to determine the accuracy of the software that was used. The following paragraphs will elaborate upon the experimental proceedings.

6.1. Setup and method for searching times experiment

The experiment was done at the University of Twente, inside the Zilverling building at Hallenweg 19, 7522 NH Enschede. The ground floor and first floor were used as the general area where the object could be located. A stool with a distinctive yellow top was chosen as object to be found. The stool represented a patient lift or bedmover. A Raspberry Pi attached to the powerbank was placed on the stool. The object was then placed in a lunch break area at the end of the hallway on the first floor. The setup is visible in Figure 22. The location is given in Figure 21. Two groups were formed. The first group was asked to find the object while using the software. The second group was asked to look for the object without the software and had to search manually. The time the subjects took to find the objects was measured.
The experiment started at the SmartXp (red dot in Figure 23), on the ground floor of the Zilverling building, at the University of Twente. A subject was approached and asked if they were interested in participating in an experiment. If the subject agreed, a brief explanation was given about the experiment. They were told that searching times of manual search and indoor positioning system assisted search were being compared. The subjects were chosen at random and they were randomly placed in the experimental group or control group. Both conditions are described below.

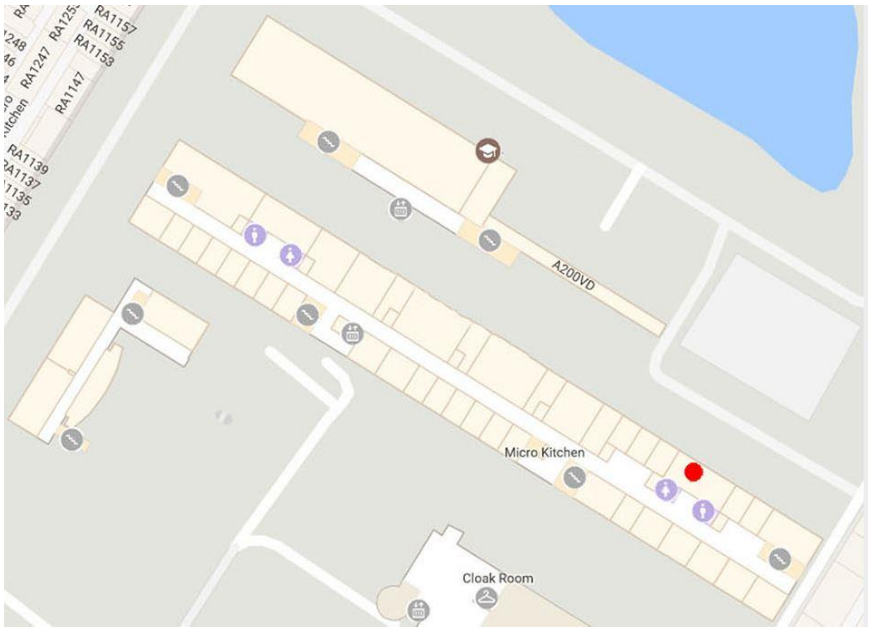


Figure 21: Map of the first floor of the Zilverling building illustrating the fingerprint nearest to the location of the stool, represented by the bright red dot [26].

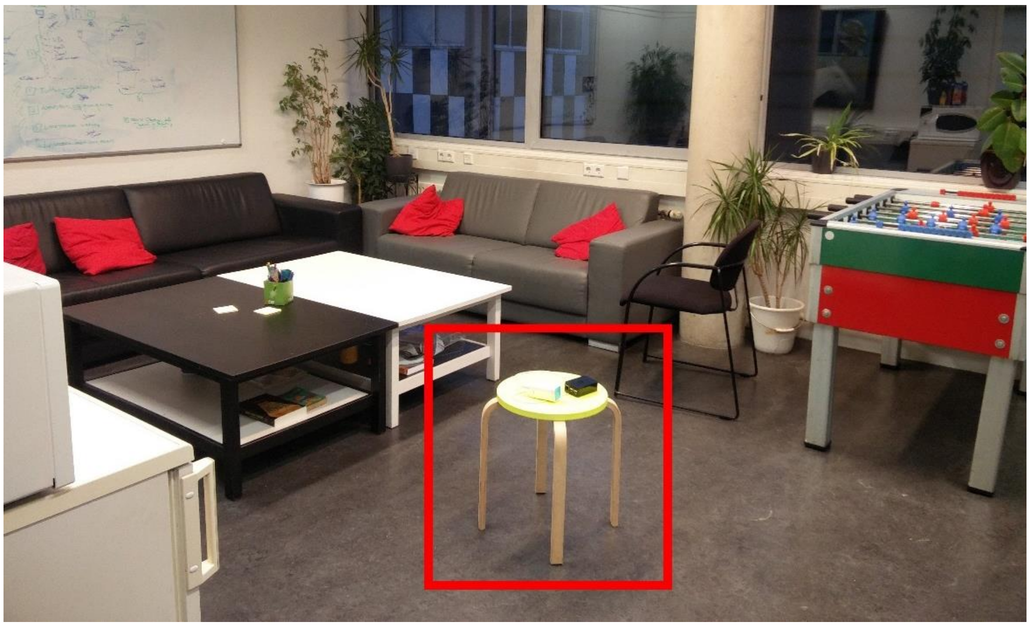


Figure 22: Photograph of the searching times experiment setup. The stool was placed in the middle of the lunch break area on the first floor of the Zilverling building. The red square indicates the Raspberry Pi attached to the powerbank.

6.1.1. Control group

The control group participants were shown a map of the ground and first floor of the Zilverling building (Figure 19). The subjects were told that they had to find a stool with a Raspberry Pi on it, and that it was hidden somewhere in that area shown on the map. The moment the subject walked away and started searching, a timer was started. As soon as the subject found the object, the timer was stopped. If the subject was interested, an explanation was given upon the system and the reason for the experiment.

In some cases, the subject was unable to find the object within the set time limit of five minutes. If this was the case, the experiment was stopped and the time was recorded as five minutes. The reason behind this was that the five minutes were a long enough time to witness whether or not the system worked. If this time limit was set higher, the effect of the use of the software would be even greater.

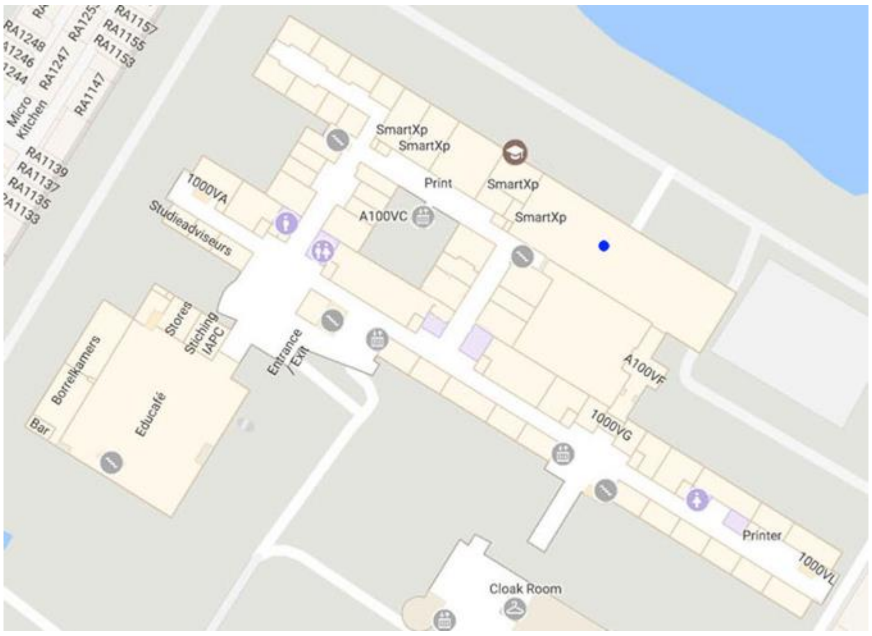


Figure 23: Map of the ground floor of the Zilverling building giving the location of the experimental starting point, represented by the blue dot [26].

6.1.2. Experimental group

The experimental group had to find the object but, as opposed to the other group, they used the software. It was explained to them that a stool with a Raspberry Pi on it was hidden somewhere within the ground and first floor of the building. As they ran the program to start their search, the timer was started. The program was run and the location was determined by the system. The observer registered whether the correct location was returned. A map like the map in Figure 21 was shown to the subject. The location of the stool was now known to the subject and they were able to walk to the object. As soon as the stool was found by the subject, the timer was stopped and the time registered. If the subject was interested an explanation was given about the system, its purpose and how it worked.

6.2. Experimental results: searching times

The results were split up into two parts: the results from the experimental group and from the control group. The data collected are shown in Table 7. A total of 20 subjects completed the experiment. They were evenly divided between both groups. In normal conditions, walking from the starting point to the location of the device would take approximately 55 seconds at a normal waking pace. The amount of steps between the two points is approximately 100.

Table 7: Experimental data of 20 subjects. The table gives the time (in seconds) in which the subjects were able to find the object with or without the software. The location given by the algorithm was accurate for all 10 subjects that used the software. Some of the control subjects were unable to find the object within the time limits, and their time was set to 300s.

Subject number	Searching time with software (in seconds)	Searching time without software (in seconds)
1	84	300
2	155	183
3	100	300
4	97	243
5	83	210
6	80	91
7	105	300
8	80	250
9	113	261
10	98	237
Average	**99.50**	**237.50**

6.3. Setup and method for accuracy experiment

The same experimental environment as before was used once again, but no participants were needed. The goal of this experiment was to determine the precision and consistency of the IPS. Insights into these properties helped to estimate the correctness and effectiveness of the software, and the effects of adding a single fingerprint to the database.

The experiment was conducted at two locations at the Zilverling. The first location was a position chosen in the hallway on the ground floor. The second position was in the SmartXp. Both locations are displayed in Figure 24, and respectably labelled 1 and 2. While the Raspberry Pi was at both locations, the program was run 25 times with a time delay between the runs of approximately 30 seconds. The ID of fingerprint that was returned by the system was recorded each time. The distance between the actual location of the Raspberry Pi and the given fingerprint was estimated.

After the experiment at the second location, the SmartXp, an additional fingerprint was added to the database at the location of the Raspberry Pi. The program was once again run 25 times and the returned fingerprint ID was recorded.

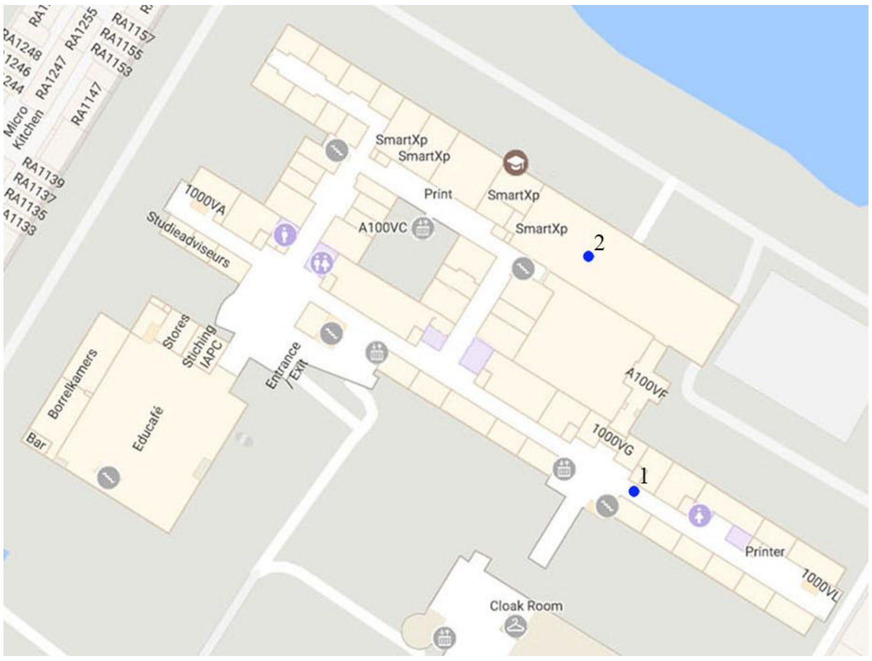


Figure 24: Map of the ground floor of the Zilverling building giving the actual position of the Raspberry Pi in the first and second position. The locations are respectively indicated with the blue dots labelled 1 and 2. Two sets of measurements were performed at the second location.

6.4. Experimental results: accuracy

The collected results for the accuracy experiment are given in three tables. The results of the hallway measurements are given in Table 8. Table 9 gives the results of the SmartXp measurements. Table 10 gives the results of the SmartXp location after adding an additional fingerprint. In the tables below, the ID's of the returned fingerprints are given. The actual location of these fingerprints are visible on a map in Figure 25 and Figure 26.

Table 8: Measurements at the hallway locations (location 1). The result is a fingerprint ID. The distance between the actual location and the returned fingerprint was determined and is also given in the table.

Fingerprint ID	Occurrence (absolute)	Occurrence (%)	Distance to actual position
15	21	84%	5 m
24	4	16%	7 m

Table 9: Measurements at the Smart XP locations (location 2) before adding an extra fingerprint. The result is a fingerprint ID. The distance between the actual location and the returned fingerprint was determined and is also given in the table.

Fingerprint ID	Occurrence (absolute)	Occurrence (%)	Distance to actual position
1	10	40%	7 m
2	3	12%	6 m
3	10	40%	2 m
4	2	8%	5 m

Table 10: Measurements at the Smart XP locations (location 2) after adding an extra fingerprint. The result is a fingerprint ID. The distance between the actual location and the returned fingerprint was determined and is also given in the table. The previously returned fingerprints are also included in the table.

Fingerprint ID	Occurrence (absolute)	Occurrence (%)	Distance to actual position
1	0	0%	7 m
2	0	0%	6 m
3	0	0%	2 m
4	0	0%	5 m
65	25	100%	0 m

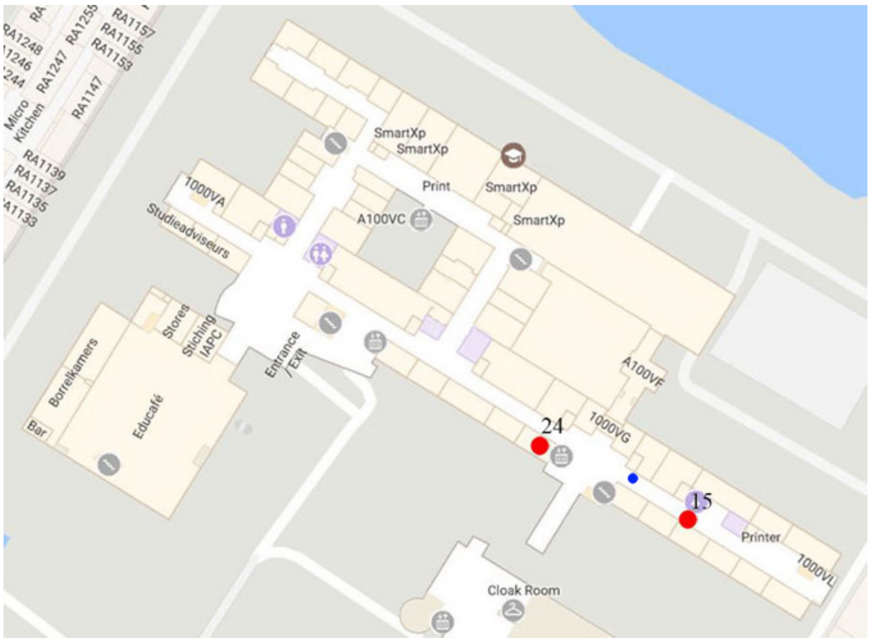


Figure 25: Map of the ground floor of the Zilverling building. The actual location of the Raspberry Pi is indicated by the blue dot. The fingerprints that were returned for the experiment with the Raspberry Pi placed in the hallway are displayed by red dots and labeled with their fingerprint ID. The distances between the actual location and fingerprints 15 and 24 are respectively 5 and 7 meters.

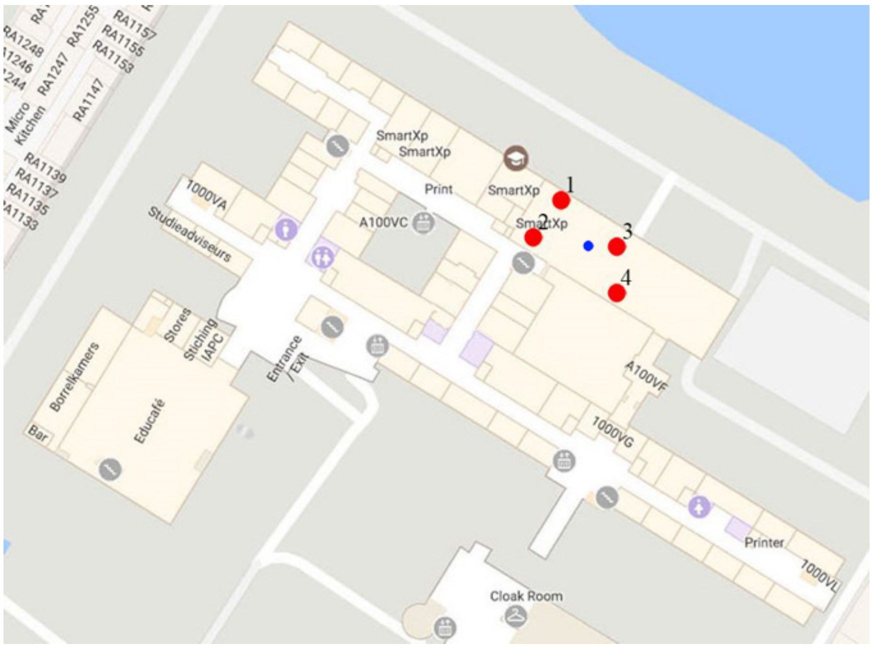


Figure 26: Map of the ground floor of the Zilverling building. The actual location of the Raspberry Pi is indicated by the blue dot. The fingerprints that were returned for the experiment with the Raspberry Pi placed in the SmartXp are displayed by red dots and labeled with their fingerprint ID. The distances between the actual location and fingerprints 1, 2, 3 and 4 are respectively 7, 6, 2 and 5 meters. The fingerprint added for the last part of the experiment, fingerprint 65, was added at the exact same location as the blue dot. Since the location of the Raspberry pi and fingerprint 65 coincide, the distance between those two is 0 meters.

6.5. Discussion

In this chapter, the data acquired during the experiments are discussed. At the end of the chapter, the limitations related to the experiments and the research are mentioned and explained.
All statistical analyses were performed using the software SPSS version 24. The mean, standard deviation and standard error mean where calculated.

6.5.1. Analysis of the searching times experiment

The results of the searching time experiment are visible in Table 11. To assess the statistical relevance of the experimental data, a statistical analyses was done with a 2 sample T-Test [27]. This test can help to determine whether there is a statistical difference between the means of two groups.

Table 11: Statistical analysis of the searching time data processed in SPSS. The data consist of the sample size, mean, standard deviation and standard error mean for the data of subjects with or without the use of the IPS software.

Experimental conditions	**Sample Size**	**Mean**	**Std. Deviation**	**Std. Error Mean**
Subjects with software	10	99.50	22.545	7.129
Subjects without software	10	237.5	64.676	20.452

For analyzing the data, a T-Test was performed. The output of the SPSS T-Test with a confidence interval of 95% was given in Table 12.

Table 12: Statistical analysis of the searching times data processed in SPSS. The results include, but are not limited to the significance, and mean difference. A T-Test with a 95% confidence interval was performed on the data to acquire the statistical analysis of the results of the experiment for the influence of the software to the searching times.

Equal Variance	F	Sig.	T	Df	Sig. (2-tailed)	Mean Difference	Std Error Difference	Lower	Upper
Assumed	4.495	0.048	6.371	18	0.000005	138.000	21.659	92.496	183.504
Not assumed			6.371	11.155	0.000050	138.000	21.659	90.409	185.591

The significance value is 0.048, which is below 0.05. This means that the 'equal variances not assumed' row should be used. The significance (2-tailed) value is 0.00050. Since the value is below 0.05, it can be concluded that there is a significant difference between the measured values. The acquired data allow a conclusion that the searching time was reduced of an average of 138 seconds when using the prototype. This is a time reduction of 58% when compared to when no software is used.

6.5.2. Analysis of the accuracy experiment.

The data from the accuracy experiment are analyzed in this paragraph. The first part of the experiment, when the Raspberry Pi was placed in the hallway, returned two locations, fingerprints 24 and 15, as could be seen in Table 8. The distance between the actual location and fingerprint 15 was shortest, and therefore fingerprint 15 was the qualitatively best feedback. From the data it can be seen that in 84% of the experiments the program determined the best available fingerprint. The remaining 16% of the measurements did however still return a quite correct location, since even though there was better location, the returned one was still the second best fingerprint. Nevertheless, Fingerprint 24 was not located in the hallway but in a room.

The second part of the experiment, when the Raspberry Pi was placed in the SmartXp, confirmed that walls do have an important effect on signal strength. All returned fingerprints were located inside the correct room. Since there are no walls inside the SmartXp, a higher difference in the returned fingerprint was expected. As can be seen from the results in Table 9, results are distributed. Since the most adequate fingerprint would have been fingerprint 3, only 40% of the data was determined correctly by the program. Nonetheless, since all fingerprints were in the correct room, and considering the intended use of the technology, any of the returned results would have resulted in the fast and correct localization of the equipment.

The third part of the experiment, when the experiment was run at the same location as before but with an added fingerprint, confirmed that increasing the amount of fingerprints in a specious room increases the accuracy of the software. As can be seen in Table 10, the distribution that could be depicted in Table 9 was eliminated. All measurements were as accurate as they could be.

Henceforth, it can be concluded that the system is effective in determining a location within the building but has troubles when too little fingerprints are added to the database.

6.5.3. Limitations

The results of this research are promising. It appears that using Wi-Fi fingerprinting for IPS can reduce searching times. Still, the results show some limitations. These are discussed in this paragraph.

The first limitation was the fact that the sample size for determining whether searching times were reduced was small. There were a total of 10 subjects in each group. This is a significant number but more subjects would increase the quality of the information and the conclusion.

The second limitation was that the system is only tested in one environment. In theory the system should work as long as there are enough AP's available on every position in the building. But this has not been tested, and problems may arise that did not in this environment.

A third limitation related to the test environment was that it was a relatively small environment, especially if compared to e.g. the size of the MST hospital. Moreover, only a small part of the building was indexed. It would be interesting to see the extent of the decrease in searching time with much larger environment sizes.

Another limitation is that both groups only consisted of students of the University of Twente. An aspect that characterize students from the University of Twente is that they are familiar with the surroundings and technology. The results could be different if people with less feeling for technology or another background would perform the search.
Additionally, the fingerprints database (see Figure 20) was small: only 65 fingerprints were added to the database. Not all rooms were indexed and in the hallway fingerprints were registered every 5-10 meters. Increased the database should make a more accurate system possible.

6.6. Evaluation of requirements

In this paragraph, all drafted requirements from 'Requirement analysis' are discussed. Moreover, it is assessed if the prototype meets the requirements that were drafted. For clarity purposes, the requirements are once again listed below and an explanation is given as to whether each of them was successfully implemented.

6.6.1. Must requirements

- **The system must reduce searching times was opposed to manual search.**
  The experiments proved that the searching time can be reduced.
- **The system must be combinable with two Indes devices: the bedmover and patient lift.**
  The prototype was a stand-alone tag, but the system could be applied to the two devices.
- **The system must be implementable in two environments (hospitals and retirement homes) as well as in different kinds of setups (premises consisting of one or several buildings).**
  The system was not tested in in both environments but they were both visited and analyzed. It was concluded that the system should work in both environments.
- **The system must be intuitive.**
  During experiments the subject's behavior was analyzed and no understanding problems were registered.
- **The system must be able to find a specific product by entering its ID.**
  This feature was implemented in the system. The ID was the IP-address of the specific tag.
- **The system must be able to search for products close to the user.**
  This was not implemented in the system. Further development of the software could implement this feature.
- **The system must be standalone and applicable on multiple devices.**
  The feedback was now given on the laptop on which the server was running. Further development of the system could make the server accessible for multiple clients.
- **The system must determine a location with an accuracy of 10 meters or less.**
  The experiment done with the prototype proved that the required accuracy of 10 meters or less can be achieved.

6.6.2. Should-requirements

- **The system should change work ethics and be a marketable product that enables everybody to use the equipment.**
  This requirement was not tested but information gathered from the interviews suggested that stakeholders are willing to change their work ethics.
- **The system should be accessible for all personnel on the work floor for whom it is useful to use.**
  The requirement was not tested, however, once the technology is in place and the interface is accessible on different platforms with different searching features, all personnel that need the technology should be able to access it.

- **The system should also be accessible for the management of the institutions.**
  This requirement was not met because it fell out of the scope of the initial boundaries set by Indes.
- **The costs of the implementation of the system should be profitable.**
  The solution to uses a Wi-Fi based fingerprinting implementation. It is one of the most low-prized IPS solutions.
- **The system should be given on a 2D map of the building floor.**
  The feedback from the system was developed according to this requirement and the location of the device was illustrated by a red dot on a 2D map.
- **The system should only search for a specific piece of equipment when it is asked to do so.**
  This requirement was achieved. The system does not operate until the server sends a request for information.
- **The system should keep the error in the vertical direction to a minimum.**
  This requirement was met. During all experiments, an error in the vertical direction was never observed.
- **If possible, the system should use of an already existing wireless communication technology as a network environment for an IPS.**
  This requirement was met. The system was created and tested in the already existing Wi-Fi wireless communication technology.

6.6.3. Could-requirements

- **The system could be able to determine which product is in use or not.**
  This requirement was not met because it fell out of the scope of the project.
- **The system could help by analyzing the distribution and usage of the equipment.**
  This requirement was not met because it fell out of the scope of the project.
- **The system could monitor the battery level and when the battery level drops below 10% it should automatically store the location.**
  This requirement was not met because it fell out of the scope of this project

7. Conclusion and future work

In this chapter, a conclusion is drawn and it is explained how the research should be continued in the future. The structure of the main research question and sub-questions will be maintained.

7.1. Conclusion

The research question was *'What is the best way for Indes to locate their bedmovers and patient lifts within a retirement home or hospital?'*. In order to answer this question, the sub-questions from the chapter Introduction were answered. Once the sub-questions are answered, the main research question will also be answered.

7.1.1. Sub question 1: what indoor positing systems do already exist?

The answer was found by looking at existing literature and observations of current applications. As opposed to outdoor localization where there is one standard, e.g. GPS, there is no standard method for IPS's. All different IPS's have different specifications and advantages and disadvantages. The researched measurement principles where Time of Flight, Angle of Arrival and Signal Strength. The wireless communication technologies that were analyzed were GPS, RFID, Cellular based, UWB, WLAN, Bluetooth and INS.

7.1.2. Sub-question 2: what requirements should the system meet for a viable product?

The chapter 'Requirement analysis' elaborates on the subject addressed in this question. First of all, initial boundaries and expectations for the first prototype were set up by Indes. Secondly, the preferences of the intended user influence the functions the system should offer. Last, some technical requirements originate from the way the system should be built and how it should handle requests. All the requirements where sorted according to the MoSCoW principle.

7.1.3. Sub- question 3: can an indoor positioning system reduce the searching time as compared to manual search?

In order to answer this sub-question, an experiment was performed. The experiment was described in 'Setup and method for searching times experiment'. The conclusion that could be drawn from the experiment was that the use of software could reduce searching times with 58% on a two floor building. It may be assumed that search area and time saving are positively correlated.

7.1.4. Sub-question 4: are Indes' clients willing to use this technique?

Stakeholders were given the possibility to answer this question themselves through interviews. All of them agreed they would use the system. If Indes develops an IPS for medical equipment that reduces searching times and increases the quality of working conditions, the stakeholders would be willing to use it. The readiness of the clients to use the technology would increase if Indes would offer the possibility to extend the applications by making the hardware standalone.

7.1.5. Main research question: What is the best way for Indes to locate their bedmovers and patient lifts within a building?

The answers given to the sub-questions above help to answer this main question. The system makes it possible to locate specific devices in large buildings. A Wi-Fi based IPS does not give a perfect positioning, but the provided location is precise enough for the objects to be found. The system is easy to implement in most buildings, because most buildings already have an existing Wi-Fi network. Hence, implementation costs can remain limited. The principle behind the software is applicable for most buildings and does not need geometric surveys. Most of the requirements that were drafted and presented in the chapter 'Requirement analysis' were met according to the analyses that was done in

the paragraph ‘Evaluation of requirements’. The unmet requirements can be met with additional time and research.
In conclusion, the Wi-Fi implementation is the best option for tracking the Indes medical devices inside buildings.

7.2. Future work

If the system should be implemented in the future, additional research should be done. There are a few shortcomings that need to be overcome.
At this moment, there is no security implemented in the system. The information sent over the Wi-Fi network is in plain text. It would be more secure to encrypt the information. In ‘The security of private information for Indes equipment’ [28], a research was done to start acquiring knowledge about this part of the system.
When the database was filled, the Raspberry Pi was placed inside the room and the program was run to add a fingerprint to the database. It was later found that for one same location, the RSSI value could vary over time. The database could be more accurate if every fingerprint was an average of 10 values taken at one location. A recommendation is that when the database is filled, a program is written that takes 10 measurements and stores the average in the database.
Every fingerprint consists of 6 MAC addresses. Over time, it was found that it could be possible to increase this number. As a recommendation, it could be very useful to extensively research the maximum number of MAC addresses stored for one fingerprint. The higher this number, the more accurate the system becomes.
For making the setup of the database easier, a program with an interface should be written. The interface should somehow replace the manual entry of x and y coordinates.
It would be useful if the system could determine if the product is in use. When a caregiver is looking for a specific object and does a search for the nearest device, knowing if it is already in use would further decrease searching times. In this case, it can be taken into consideration that the object is not available and suggest another device.
The experiment was done in a controlled and relatively small environment. It would be interesting to scale up the experimental area and see how much the decrease in searching times would increase on a larger scale. With that information, a better analysis can be done as to how much the real searching will be reduced. From there, a calculation can be made concerning how much the cost reduction would become if an IPS would really be used.
For making the system more useful and more versatile, the possibility of continuous tracking should be considered. The interviews led to the conclusion that certain stakeholders would be interested in this feature. Analyzing the collected data could be an advantageous extra feature for the management.
The last recommendation is to see if the tag could be improved. For prototyping purposes, a Raspberry Pi was used to collect the data. A Raspberry Pi is an overclassified device to be a tag in a real system. Another piece of hardware that can function as tag should be developed. It is important that a Wi-Fi tag is an active tag, so it requires power. If a standalone tag is developed, the scope of the technology will be broadened. It is not only implementable in the healthcare branch but in every branch where larger expensive equipment is shared through buildings.
The advice given to Indes is to first make a decision: there are two options to be considered. Should the system be with a built in tag or standalone tag? Both sides have advantages and disadvantages. To make a final decision two follow-up researches should be done. A solution for the power problem with a standalone tag should be found and the possibilities of the Internet of Things could be explored with a built in tag.

8. References

[1] H. Liu, H. Darabi, P. Banerjee, and J. Liu, “Survey of wireless indoor positioning techniques and systems,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 6, pp. 1067–1080, 2007.

[2] L. P. Wen, C. W. Nee, K. M. Chun, T. an Shiang-yen, and R. Idrus, “Application of WiFi-based Indoor Positioning System in Handheld Directory System,” *Proc. 5th Eur. Conf. Eur. Comput. Conf. World Sci. Eng. Acad. Soc.*, pp. 21–27, 2011.

[3] S. Goswami, *Indoor LocationTechnologies*, no. 1. 2014.

[4] M. Werner, *Indoor location-based services: Prerequisites and foundations*. 2014.

[5] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti, “SpotFi : Decimeter Level Localization Using WiFi,” *Sigcomm 2015*, pp. 269–282, 2015.

[6] P. Le Dortz, N. and Gain, F. and Zetterberg, “Wifi Fingerprint Indoor Positioning System Using Probability Distribution Comparison,” *Acoust. Speech Signal Process. (ICASSP), 2012 IEEE Int. Conf.*, pp. 2301–2304, 2012.

[7] K. Thapaliya and Goo-Rak Kwon, “Enhanced weighted K-nearest neighbor algorithm for indoor Wi-Fi positioning systems,” *Comput. Technol. Inf. Manag. (ICCM), 2012 8th Int. Conf.*, vol. 1, pp. 515–520, 2012.

[8] Shopify, “http://cdn.shopify.com/s/files/1/0219/9382/products/ibw_02_large.PNG?v=1401283951.” .

[9] A. D. Velasco and S. M. Delgado, “Indoor Positioning using the Android Platform,” 2014.

[10] NASA, “http://missionscience.nasa.gov/ems/03_behaviors.html.”

[11] Gaussian Wave, “http://www.gaussianwaves.com/gaussianwaves/wp-content/uploads/2013/07/Multipath_1_copyrighted.png.”

[12] A. W. S. Au *et al.*, “Indoor tracking and navigation using received signal strength and compressive sensing on a mobile device,” *IEEE Trans. Mob. Comput.*, vol. 12, no. 10, pp. 2050–2062, 2013.

[13] A. Disha, “A Comparative Analysis on indoor positioning Techniques and Systems,” *Int. J. Eng. Res. Appl.*, vol. 3, no. 2, pp. 1790–1796, 2013.

[14] F. Evennou and F. Marx, “Advanced integration of WiFi and inertial navigation systems for indoor mobile positioning,” *EURASIP J. Appl. Signal Processing*, vol. 2006, pp. 1–11, 2006.

[15] J. Stook, “Planning an indoor navigation service for a smartphone with Wi-Fi fingerprinting localization,” pp. 1–145, 2011.

[16] LevelUpAssist, “http://www.levelupassist.nl/.”

[17] Lone Rooftop, “https://lonerooftop.com/.”

[18] SecureLink Nederland, “https://www.securelink.nl/.”

[19] Aruba Networks, “http://www.arubanetworks.com/products/networking/ale/.”

[20] Raspberry Pi, “https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.”

[21] TP-Link, “http://nl.tp-link.com/products/details/cat-5689_TL-PB10400.html.”

[22] Kingston, “https://www.kingston.com/en/flash/microsd_cards/sdc4.”

[23] Dell, “http://www.dell.com/nl/bedrijven/p/precision-m2800-workstation/pd.”

[24] Eclipse, “https://eclipse.org/.”

[25] N. Sadawi, “https://github.com/nsadawi/KNN.”

[26] Google, “https://www.google.nl/maps/place/Zilverling,+Hallenweg+19,+7522+NH+Enschede/@52.2391942,6.8546106,17z/data=!3m1!4b1!4m5!3m4!1s0x47b813db72496ee9:0x349c7c8ce617ec95!8m2!3d52.2391942!4d6.8567993.”

[27] A. Te Grotenhuis, M., & Matthijssen, *Basiscursus SPSS*, Version 18. 2011.

[28] D. Rieffe, “The security of private information for Indes equipment, Course: Creative Exploration in AS & T,” 2017.

Appendix A

Does the problem of lost medical equipment sound familiar within your organization?

How is the work flow at this moment for the specialized medical equipment?

What is your professional opinion on how the work flow could be improved in the future?

What kind of wireless network technology is available within your buildings?

How accurately do you think the system should be able determine the location?

In your professional opinion, do you think that there is a demand for a system like this?

What is your professional opinion on how the feedback from the system should be returned?

Do you expect personnel is willing to work with a system like this?

Do you think management of hospitals and retirement homes are willing to invest in a system like this?

Do you expect a lower searching time when a system could help the personnel during their work?

Appendix B

```
package nl.dennis.structs;

import java.io.Serializable;
import java.util.Locale;

public class AP implements Serializable{
        static final long serialVersionUID = 1L;
        private final String BSSID;
        private final int RSSI;
        private final String SSID;
        private  Double RSSIDouble;

        public AP(String BSSID, int RSSI, String SSID) {
                this.BSSID = BSSID;
                this.RSSI = RSSI;
                this.SSID = SSID;
        }
    public String toString(){
        return String.format(Locale.US, "%s: %d dBm %s", BSSID, RSSI, SSID);
    }

    public String getBSSID(){
        return BSSID;
    }

    public int getRSSI(){
        return RSSI;
    }

    public String getSSID(){
        return SSID;
    }

    public Double getRSSIDouble(){
        RSSIDouble = 1.0 * RSSI;
        return RSSIDouble;
    }
}
```

```
package workingPackage;
import java.util.ArrayList;
import java.util.*;

/*
 * Database class. This class is build up as a struct. With the decleration of a new datapoint all the
five values in the constructor are filled.
 * every Datapoints has 5 object
 * ID
 * ArrayList MAC
 * ArrayList RSSI
 * xpos
 * ypos
 */


public class Database {
        private final int ID;
        private List<String> BSSID = new ArrayList<String>();
        private List<Double> referencePoints = new ArrayList<Double>();
        private int xPos;
        private int yPos;

        public Database(int newLocation, List<String> newBSSID, List<Double> newRefrencePoint, int
newxPos, int newyPos) {
                ID = newLocation;
                BSSID = newBSSID;
                referencePoints = newRefrencePoint;
                xPos = newxPos;
                yPos = newyPos;

        }

        public int getLocation(){
                return ID;
        }

        public int getXvalue(){
                return xPos;
        }

        public int getYvalue(){
                return yPos;
        }

        public List<String> getBSSID(){
                return BSSID;
        }

        public List<Double> getReferencePoints(){
                return referencePoints;
        }
}
```

```
package workingPackage;

import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.*;

/*
 * Class resposible for adding new datapoints to the csv file. Has it's own public static void main
method
 *
 *
 */
public class DataFinder {

        public static void main(String[] args) {

                DataFinder dataFinder = new DataFinder();
                try {
                        dataFinder.findValue();
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

        public void findValue() throws FileNotFoundException, IOException {
                String IPAdress = "130.89.225.227";
                FileWriter pw;
                String fileName = "wifidata.csv";
                pw = new FileWriter(fileName, true);
                int amountOfReferencePoints = 6;
                String ID = "1";
                String xLoc = "50";
                String yLoc = "50";
                ArrayList<String> BSSID = new ArrayList<String>();
                ArrayList<String> RSSI = new ArrayList<String>();
                ValueRequester valueRequester = new ValueRequester();
                valueRequester.start(IPAdress);
                if (valueRequester.getFilData().size() < amountOfReferencePoints) {
                        System.out.println("Niet genoeg values gevonden op deze locatie...");
                } else {
                        for (int i = 0; i < amountOfReferencePoints; i++) {
                                BSSID.add(valueRequester.getFilData().get(i).getBSSID());

        RSSI.add(Integer.toString(valueRequester.getFilData().get(i).getRSSI()));
                                // BSSID.add(valueRequester.getAllData().get(i).getBSSID());
                                //
RSSI.add(Integer.toString(valueRequester.getAllData().get(i).getRSSI()));
                        }
                        System.out.println(BSSID);
                        System.out.println(RSSI);

                        StringBuilder sb = new StringBuilder();
                        sb.append(ID);
                        sb.append(",");
                        for (int i = 0; i < BSSID.size(); i++) {
                                sb.append(BSSID.get(i));
                                sb.append(",");
                        }
                        for (int i = 0; i < RSSI.size(); i++) {
                                sb.append(RSSI.get(i));
                                sb.append(",");
                        }
                        sb.append(xLoc);
                        sb.append(",");
                        sb.append(yLoc);
                        sb.append("\n");
                        pw.write(sb.toString());
                        pw.close();
                }
        }
}
```

```
package workingPackage;
/*
 * Struct class for all the devices
 *
 */
public class Devices {

	private String IPAdress;
	private String deviceName;

	public Devices(String IPAdress, String deviceName) {
		this.IPAdress = IPAdress;
		this.deviceName = deviceName;
	}

	public String getIPAdress() {
		return IPAdress;
	}

	public String getDeviceName(){
		return deviceName;
	}

}
```

```
package workingPackage;

import java.util.*;
import java.io.*;


/*
 * In this class the struct Database is called and filled from a csv file created in Excel.
 * All the data is filed in a List of Databases called allDatapoints.
 * This can be called through the entire program.
 *
 */
public class FillingDatabase {

	private List<Database> allDataPoints = new ArrayList<Database>();
	private ArrayList<ArrayList<String>> testArrayString = new ArrayList<ArrayList<String>>();
	private ArrayList<ArrayList<Double>> testArrayDouble = new ArrayList<ArrayList<Double>>();
	private String fileName = "wifidata.csv";
	private File file = new File(fileName);
	private static final int databaseSize = 65;

	public FillingDatabase() {

	}

	public void startFilling() {
		addPoints();
	}

	public void addPoints() {

		try {

			Scanner inputStream = new Scanner(file);
			inputStream.next();
			//filling database from csv file
			for (int i = 0; i < databaseSize; i++) {
				String data = inputStream.next();
				String[] values = data.split(",");
				ArrayList<String> nSM1 = new ArrayList<String>();
				testArrayString.add(nSM1);
				ArrayList<Double> nSD1 = new ArrayList<Double>();
				testArrayDouble.add(nSD1);
				nSM1.add(values[1]);
				nSM1.add(values[2]);
				nSM1.add(values[3]);
				nSM1.add(values[4]);
				nSM1.add(values[5]);
				nSM1.add(values[6]);
				Double dRSSI1 = Double.parseDouble(values[7]);
				Double dRSSI2 = Double.parseDouble(values[8]);
				Double dRSSI3 = Double.parseDouble(values[9]);
				Double dRSSI4 = Double.parseDouble(values[10]);
				Double dRSSI5 = Double.parseDouble(values[11]);
				Double dRSSI6 = Double.parseDouble(values[12]);
				nSD1.add(dRSSI1);
				nSD1.add(dRSSI2);
				nSD1.add(dRSSI3);
				nSD1.add(dRSSI4);
				nSD1.add(dRSSI5);
				nSD1.add(dRSSI6);
				int index = Integer.parseInt(values[0]);
				int xpos = Integer.parseInt(values[13]);
				int ypos = Integer.parseInt(values[14]);
				Database database = new Database(index, testArrayString.get(i),
testArrayDouble.get(i), xpos, ypos);
				allDataPoints.add(database);
			}
			inputStream.close();
		} catch (FileNotFoundException e) {
			e.printStackTrace();
		}
		//printData();
	}
	public List<Database> getAllData() {
		return allDataPoints;
	}

	public void printData() {
		for (Database data : allDataPoints) {
			System.out.println(data.getLocation());
			System.out.println(data.getXvalue());
			System.out.println(data.getYvalue());
			System.out.println(data.getBSSID());
			System.out.println(data.getReferencePoints());
		}
	}
}
```

W

```
package workingPackage;

import java.util.*;

/*
 * This class determines the difference in RSSI between the measured value and the database points.
 */

public class KNN {

        private List<DataPoint> dataPointList;
        private List<Result> resultList;
        private ValueRequester scanValue = new ValueRequester();
        private FillingDatabase values = new FillingDatabase();
        private LoadingImage image = new LoadingImage();
        private List<Double> query;
        private List<String> queryString;
        private int totalRounds = 6;
        private String choiceForDevice;

        public void startAlgorithm(String choice) {
                choiceForDevice = choice;
                values.startFilling();
                scanValue.start(choiceForDevice);
                // list to save data
                dataPointList = new ArrayList<DataPoint>();
                // list to save distance result
                resultList = new ArrayList<Result>();

                for (int i = 0; i < values.getAllData().size(); i++) {
                        dataPointList.add(new DataPoint(
                                        values.getAllData().get(i).getReferencePoints(),
                                        values.getAllData().get(i).getBSSID(),
                                        values.getAllData().get(i).getLocation()));
                }

                query = new ArrayList<Double>();
                queryString = new ArrayList<String>();
                for (int i = 0; i < scanValue.getFilData().size(); i++) {
                //for (int i = 0; i < scanValue.getDebugData().size(); i++) {
                        Double tempDouble;
                        String tempString;
                        tempDouble = scanValue.getFilData().get(i).getRSSIDouble();
                        tempString = scanValue.getFilData().get(i).getBSSID();
                        //tempDouble = scanValue.getDebugData().get(i).getRSSIDouble();
                        //tempString = scanValue.getDebugData().get(i).getBSSID();

                        query.add(tempDouble);
                        queryString.add(tempString);

                }

                startComparing();
        }

        public void startComparing() {
                for (int i = 0; i < dataPointList.size(); i++) {
                        double dist = Double.POSITIVE_INFINITY;
                        int counter = 0;
                        for (int j = 0; j < dataPointList.get(i).getdataBSSID().size(); j++) {
                                for (int h = 0; h < queryString.size(); h++) {
                                        if
(dataPointList.get(i).getdataBSSID().get(j).equals(queryString.get(h))) {
                                                dist = dist == Double.POSITIVE_INFINITY ? 0.0 : dist;
                                                Double refRSSI =
dataPointList.get(i).getdataAttributes().get(j);
                                                Double measRSSI = query.get(h);
                                                // System.out.println("Database: " + refRSSI);
                                                // System.out.println("Measure: " + measRSSI);
                                                dist += Math.pow(refRSSI - measRSSI, 2);
                                                counter++;
                                        }
                                }
                        }
                        // System.out.println(i + ": the counter: " + counter);
                        // System.out.println(i + ": the total rounds: " + totalRounds);
                        if (counter == totalRounds) {
                                double distance = Math.sqrt(dist);
                                resultList.add(new Result(distance,
dataPointList.get(i).getdataName()));
                                System.out.println("Matching Mac adress: " + totalRounds);
                        }
                }

                if (resultList.isEmpty()) {
                        totalRounds--;
                        startComparing();
                } else {
                        sortList();
                }

        }

        public void sortList() {

                // System.out.println(resultList);
                Collections.sort(resultList, new DistanceComparator());
                List<Integer> ss = new ArrayList<Integer>();
                for (int x = 0; x < resultList.size(); x++) {
                        // System.out.println(resultList.get(x).dataNumber + " .... " +
                        // resultList.get(x).distance);
                        // get classes of k nearest instances (data names) from the list
                        // into an array
                        ss.add(resultList.get(x).dataNumber);
                }

                for (Result results : resultList) {
                        System.out.println("Closest Access point: " + results.dataNumber);
                        System.out.println("Total Distance: " + results.distance + "\n");
                }

                int winner = resultList.get(0).dataNumber;
                image.startLoading(winner - 1);
        }

        public void printValues() {
                for (int i = 0; i < dataPointList.size(); i++) {
                        System.out.println(dataPointList.get(i).dataName);
                        System.out.println(dataPointList.get(i).dataAttributes);
                }
        }

        public List<Result> getResults() {
                return resultList;
        }

        static class DataPoint {
                List<Double> dataAttributes = new ArrayList<Double>();
                List<String> dataBSSID = new ArrayList<String>();
                int dataName;

                public DataPoint(List<Double> dataAttributes, List<String> dataBSSID, int dataName) {
                        this.dataName = dataName;
                        this.dataAttributes = dataAttributes;
                        this.dataBSSID = dataBSSID;
                }

                public List<Double> getdataAttributes() {
                        return dataAttributes;
                }

                public List<String> getdataBSSID() {
                        return dataBSSID;
                }
```

```
		public int getdataName() {
			return dataName;
		}

	}

	// simple class to model results (distance + class)
	static class Result {
		double distance;
		int dataNumber;

		public Result(double distance, int dataNumber) {
			this.dataNumber = dataNumber;
			this.distance = distance;
		}

	}

	// simple comparator class used to compare results via distances
	static class DistanceComparator implements Comparator<Result> {
		@Override
		public int compare(Result a, Result b) {
			return a.distance < b.distance ? -1 : a.distance == b.distance ? 0 : 1;
		}
	}

}
```

```
package workingPackage;
import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.WindowConstants;
import java.awt.Graphics;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

/*
 * This class is responsible for loading the backgroundImage and drawing the red dot on a specific
location.
 *
 */

public class LoadingImage extends JPanel {
        private static final long serialVersionUID = 1L;
        private BufferedImage backgroundImage;
        private BufferedImage redDot;
        private FillingDatabase newData = new FillingDatabase();
        private int xLocation;
        private int yLocation;
        private int winner;
        private static final String IMAGENAME = "finalMapUt.jpg";

        public void startLoading(int newWinner) {
                winner = newWinner;
                newData.addPoints();
                JFrame frame = buildFrame();
                try {
                        //Loading the images
                        backgroundImage = ImageIO.read(new File(IMAGENAME));
                        redDot = ImageIO.read(new File("reddot.png"));
                } catch (IOException e) {
                        e.printStackTrace();
                }
                //Determine the right location for the red dot.
                xLocation = newData.getAllData().get(winner).getXvalue();
                yLocation = newData.getAllData().get(winner).getYvalue();

                JPanel pane = new JPanel() {
                        private static final long serialVersionUID = 1L;

                        @Override
                        protected void paintComponent(Graphics g) {
                                //Drawing the images
                                super.paintComponent(g);
                                g.drawImage(backgroundImage, 0, 0, null);
                                g.drawImage(redDot, xLocation, yLocation, null);
//                              for (int i = 0; i < 4; i++) {
//                                      g.drawImage(redDot, newData.getAllData().get(i).getXvalue(),
newData.getAllData().get(i).getYvalue(), null);
//                              }

                        }
                };
                frame.add(pane);
        }

        private static JFrame buildFrame() {
                JFrame frame = new JFrame();
                frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
                frame.setSize(1900, 698);
                frame.setVisible(true);
                return frame;
        }

}
```

```
package workingPackage;


public class Main {

/*
 * Dennis Rieffe
 * An Indoor positioning system for Indes B.V.
 * 14-12-2016
 */

        public static void main(String[] args) {

//              LoadingImage loadingImage = new LoadingImage();
//              loadingImage.startLoading(2);

//              KNN searchingNear = new KNN();
//              searchingNear.startAlgorithm();

//              PractiseWithKNN searchingNearPractise = new PractiseWithKNN();
//              searchingNearPractise.startAlgorithm();

                Search search = new Search();
                search.addDevices();


        }
}
```

```
package workingPackage;

import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class Search {

        private ArrayList<Devices> allDevices = new ArrayList<Devices>();
        private KNN knn = new KNN();
        private String fileName = "devicedata.csv";
        private File file = new File(fileName);
        private static final int databaseSize = 3;

        public void addDevices() {

                try {
                        Scanner inputStream = new Scanner(file);
                        inputStream.next();
                        // filling database from csv file
                        for (int i = 0; i < databaseSize; i++) {
                                String data = inputStream.next();
                                String[] values = data.split(",");
                                // System.out.println("ip: " + values[0] + " name: " +
                                // values[1]);
                                Devices device = new Devices(values[0], values[1]);
                                allDevices.add(device);
                        }
                        inputStream.close();
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                }
                choice();
        }

        public void choice() {
                Scanner scanner = new Scanner(System.in);
                System.out.println("What specific device you want to find? \nAvailible are:");
                for (int i = 0; i < allDevices.size(); i++) {
                        System.out.println((i + 1) + ": Name: " + allDevices.get(i).getDeviceName() +
" IP address: "
                                        + allDevices.get(i).getIPAdress());
                }
                System.out.println("Make your choice?");
                int choice = 0;
                try {
                        choice = scanner.nextInt();
                        if (choice > databaseSize) {
                                System.out.println("That specific device is not availible");
                                choice();
                        }
                } catch (InputMismatchException e) {
                        System.out.println("Wrong input, try something else...");
                        choice();
                }

                knn.startAlgorithm(allDevices.get(choice - 1).getIPAdress());
                scanner.close();
        }

}
```

```
package workingPackage;

import java.net.Socket;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.net.InetAddress;
import java.util.List;

import nl.dennis.structs.AP;


/*
 * This class is the requester for the values. On a raspberry pi who has a serverSocket a request can
be done for a connection.
 * The connection must be done to a specific ip adress.
 * When a connection is established the pi collects it's values and sends them back over the socket.
Next the socket will be closed.
 */
public class ValueRequester {

        public ValueRequester() {

        }
        //Name of the specific network you want to use
        public static final String SSID = "eduroam";
        //IPadress for you raspberry pi/tag.

        private List<AP> WifiValues = new ArrayList<AP>();
        private List<AP> filWifiValues = new ArrayList<AP>();
        private List<AP> debugList = new ArrayList<AP>();
        private Socket socket;
        private Search search;

        @SuppressWarnings("unchecked")

        public void start(String IPAdress) {

                String IPADRESS = IPAdress;
                search = new Search();
                        //Reading the values
                try {
                        InetAddress adress = InetAddress.getByName(IPADRESS);
                        socket = new Socket(adress, 4444);
                        ObjectInputStream objectInput = new
ObjectInputStream(socket.getInputStream());
                        try {
                                Object object = objectInput.readObject();
                                //Cast them to a list;
                                WifiValues = (List<AP>) object;
                        } catch (ClassNotFoundException e) {
                                e.printStackTrace();
                        }
                } catch (UnknownHostException e) {
                        System.out.println("Cannot find the specific device.");
                        search.choice();

                } catch (IOException e) {
                        System.out.println("Cannot find the specific device.");
                        search.choice();
                }

                 for (AP value : WifiValues) {
                 System.out.println(value);
                 }

                try {
                        socket.close();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                filterList();
                fillDebugList();
        }

        public void filterList() {
                for (AP value : WifiValues) {
                        if (value.getSSID().equals(SSID)) {
                                filWifiValues.add(value);
                        }
                }
//              for (AP value : filWifiValues) {
//                      System.out.println(value);
//              }

        }
        //fake values for debugging.
        public void fillDebugList() {
                debugList.add(new AP("00:26:cb:42:22:50", -77, SSID));
                debugList.add(new AP("30:37:a6:c3:b9:b1", -44, SSID));
                debugList.add(new AP("00:26:cb:42:0d:91", -40, SSID));
                debugList.add(new AP("30:37:a6:c3:a8:a1", -73, SSID));
                debugList.add(new AP("00:26:cb:42:12:81", -91, SSID));
                debugList.add(new AP("30:37:a6:c3:b9:81", -55, SSID));

                for (AP acces : debugList) {
                        System.out.println(acces);
                }

        }

        public List<AP> getAllData() {
                return WifiValues;
        }

        public List<AP> getFilData() {
                return filWifiValues;
        }

        public List<AP> getDebugData() {
                return debugList;
        }
}
```

```
package nl.dennis.structs;

import android.view.View;

import java.io.Serializable;
import java.util.Locale;

/**
 * Created by super_000 on 07-Dec-16.
 */

public class AP implements Serializable{
    static final long serialVersionUID = 1L;
    private final String BSSID;
    private final int RSSI;
    private final String SSID;

    public AP(String BSSID, int RSSI, String SSID) {
        this.BSSID = BSSID;
        this.RSSI = RSSI;
        this.SSID = SSID;
    }
    public String toString(){
        return String.format(Locale.US, "%s: %d dBm %s", BSSID, RSSI, SSID);
    }

    public int getRSSI() {
        return RSSI;
    }
}
```

```
package com.example.super_000.wifilistactivity;

import android.Manifest;
import android.app.ListActivity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.net.wifi.ScanResult;
import android.net.wifi.WifiManager;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.text.format.Formatter;
import android.util.Log;
import android.widget.ArrayAdapter;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;
import java.util.Timer;
import java.util.TimerTask;

import nl.dennis.structs.AP;


/**
 * Created by Dennis Rieffe on 05-12-16.
 */

public class WifiListActivity extends ListActivity {

    private static final String IPADDRESS = "130.89.237.52";
    private ArrayAdapter<AP> adapter;
    private Thread thread;
    private List<AP> currentMeasurements;
    private WifiManager wifiManager;
    private WifiReceiver receiver;
    Timer timer = new Timer();
    TimerTask doAsynchronousTask = new TimerTask() {

        @Override
        public void run() {
            try {
                wifiScan wscan = new wifiScan();
                wscan.execute();
            }catch (Exception e) {}
        }
    };


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M &&
checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
{
            requestPermissions(new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
PERMISSIONS_REQUEST_CODE_ACCESS_COARSE_LOCATION);
        }


        wifiManager = (WifiManager) getSystemService(Context.WIFI_SERVICE);
        receiver = new WifiReceiver();
        currentMeasurements = Collections.synchronizedList(new ArrayList<AP>());

        thread = new Thread(new Runnable() {
            @Override
            public void run() {
                startServer();
            }
        });

        thread.start();

        // Now create a new list adapter bound to the cursor.
        // SimpleListAdapter is designed for binding to a Cursor.
        adapter = new ArrayAdapter<AP>(this, android.R.layout.two_line_list_item, android.R.id.text1);
        setListAdapter(adapter);
    }

    public void startServer(){
        try {

            WifiManager wm = (WifiManager) getSystemService(WIFI_SERVICE);
            String ip = Formatter.formatIpAddress(wm.getConnectionInfo().getIpAddress());
            InetAddress address = InetAddress.getByName(ip);

            ServerSocket serverSocket = new ServerSocket(4444, 50, address);
            while (true) {
                Socket skt = serverSocket.accept();
                registerReceiver(receiver, new IntentFilter(
                        WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
                wifiManager.startScan();
                synchronized (thread) {
                    if (currentMeasurements.size() == 0) {
                        thread.wait();
                    }
                }
                ObjectOutputStream objectOutputStream = new ObjectOutputStream(skt.getOutputStream());
                objectOutputStream.writeObject(currentMeasurements);
                objectOutputStream.flush();
                skt.close();
            }
            }catch(IOException e){
                e.printStackTrace();
            }catch(InterruptedException e){
                e.printStackTrace();
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions,
                                           int[] grantResults) {
        if (requestCode == PERMISSIONS_REQUEST_CODE_ACCESS_COARSE_LOCATION
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
        }
    }

    private final int PERMISSIONS_REQUEST_CODE_ACCESS_COARSE_LOCATION = 1001;


    public class WifiReceiver extends BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent) {
            WifiManager manager = (WifiManager) context.getSystemService(Context.WIFI_SERVICE);
            Log.v("Tag", "Scan results received");
            List<ScanResult> results = manager.getScanResults();
            List<AP> temp = new ArrayList<>();
            for (ScanResult res : results) {
                AP ap = new AP(res.BSSID, res.level, res.SSID);
                temp.add(ap);
            }
            Collections.sort(temp, new Comparator<AP>() {
                @Override
                public int compare(AP ap, AP t1) {
                    return t1.getRSSI - ap.getRSSI;
                }
            });
            Log.d("Lijst" ,Integer.valueOf(temp.size()).toString());
            currentMeasurements.clear();
            currentMeasurements.addAll(temp);
            synchronized (thread) {
                thread.notify();
            }
            adapter.clear();
            adapter.addAll(temp);
            adapter.notifyDataSetChanged();
        }
    }

    public class wifiScan extends AsyncTask<Void, String, Void> {


        protected void onPreExecute() {

        }

        protected void onPostExecute(Void results) {

        }

        @Override
        protected Void doInBackground(Void... params) {
            registerReceiver(receiver, new IntentFilter(
                    WifiManager.SCAN_RESULTS_AVAILABLE_ACTION));
            wifiManager.startScan();
            return null;
        }
    }

}
```