Kurzfassung

In den letzten Jahren dominierte das Smartphone die Absatzmärkte in der Elektronikbranche. Da sich dieser Markt in den letzten Monaten langsam sättigt, wird nach neuen Technologien gesucht um einen neuen profitablen Markt zu erschließen. Immer populärer werden Wearable Devices wie Smartwatches, welche in letzter Zeit durch eine ausgereifte Technologie Marktreife erlangt haben. Viele Elektronikkonzerne entwickeln heutzutage Smartwatches um in diesem aufstrebenden Markt Fuß zu fassen. Dabei wird auch Fokus auf die Kombination von Smartwatches und dem Fitness- und Gesundheitssektor gelegt und es werden bereits viele verschiedene Applikationen dafür entwickelt. Da Wearables direkt am Körper getragen werden und diese meist mit mehreren Sensoren ausgestattet sind, bieten diese Geräte eine gute Basis zur Kontrolle verschiedener Körperfunktionen durch die Überwachung von Sensorenwerten.

In dieser Arbeit wird ein System vorgestellt, welches mit den Sensoren einer Smartwatch den Gesundheitszustand von Menschen überwacht und bei gewissen Situationen eine Nachricht an ein Android Smartphone schickt. Die vom Smartphone aufgezeichneten Daten, werden zu einem privaten Cloudspeicher gesendet und weiterverarbeitet. Es besteht die Möglichkeit über ein Webinterface die Daten auf einem Computer mit Internetverbindung und Webbrowser darzustellen. Es können Diagramme von den Daten angezeigt werden und über eine tabellarische Übersicht ist es möglich die Daten und Nutzer zu bearbeiten. Zur Verfügung steht dazu eine *MetaWatch* Smartwatch, welche über *BLE - Bluetooth Low Energy* mit einem *Android* Smartphone verbunden ist. Die Smartwatch bietet hierfür Accelerometer Daten, welche für eine Aktivitätserkennung und einen Schrittzähler verwendet werden. Durch die Einbindung von einem speziellen Service ist es möglich bei gewissen Situationen eine Alarm Nachricht an ein Smartphone zu verschicken. Abschließend werden noch existierende Applikationen mit den Resultaten der in dieser Arbeit entwickelten Applikation verglichen.

Abstract

In the last few years smartphones dominated the market in the electronic sector. With the saturation of the smartphone market, the electronic branch is looking forward to a new technology, which opens up a new profitable market. Wearable Devices like the smartwatch have evolved in the last month and matured to a new domain in the sales. Many manufacturers produce smartwatches nowadays, focusing on the fitness and health sector and provide new possibilities for diverse applications. Wearables provide a good basis for health control, because these devices are directly worn on the body and are usually equipped with one or more sensors.

This work presents a system, which is capable of monitoring the health of humans with the sensors of a wearable device. The accelerometer data is derived from a *MetaWatch* smartwatch, connected via *BLE - Bluetooth Low Energy* to an Android smartphone. The smartphone collects the data, pre-processes it and sends it to a private mobile cloud, where it is further processed and monitored. With a web interface it is possible to view the data on any workstation with internet access. The system is capable of showing charts of the data and the tabular view makes it possible to edit user data and data sets. The smartwatch provides accelerometer data to the system, which is used for activity monitoring and a step counter application. Through the integration of a special service it is also possible to send an alert to a smartphone. Beside the development of this application, the results are compared with other applications offered on the application stores.

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material, which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

....
date (signature)

Danksagung

Diese Diplomarbeit wurde im Studienjahr 2014/15 am Institut für Technische Informatik an der Technischen Universität Graz durchgeführt.

Ich bedanke mich bei allen am Institut für Technische Informatik, die mich bei meiner Diplomarbeit unterstützt haben und mir bei fachlichen Fragen stets zur Seite gestanden sind. Besonderer Dank gilt meinem Betreuer Herrn Ass. Prof. Dipl.-Ing. Dr. techn. Christian Steger für die fachliche und gute Betreuung.

Ganz großer Dank gilt auch meiner Freundin, die mich während der ganzen Zeit unterstützt und immer Verständnis für mich aufgebracht hat. Außerdem möchte ich mich bei meiner Familie, insbesondere bei meinen Eltern, herzlich bedanken. Sie haben mir mein Studium ermöglicht, mich während meiner gesamten Studienzeit begleitet und auch immer an mich geglaubt.

Graz, im April 2015 Stephan Oberauer

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The mobile sector in computer science gained more popularity with the introduction of smartphones in the year 2007. Since then, smartphone disposals have reached a saturation and the companies in the technology sector are researching into new technologies to gain market share. With the introduction of powerful smartwatches onward the year 2011, the wearable technology pushed into the electronic consumer market. Many manufacturers started developing smartwatches to establish themselves in the wearable sector. With the increase of smaller wearable devices, also the popularity of fitness and health monitoring systems has evolved. Because of the smartwatch sensors, like 3-axis accelerometers, GPS or heart rate sensor, many fitness applications can be found in the mobile application stores nowadays. The smartphone is usually carried in pockets or bags, whereas wearables like smartwatches are worn directly on the body, which enables every minute monitoring of human vital signs like heart rate or recognize activities like walking.

The history of computerized wearables, like the popular smartwatches nowadays, date back to the early 1980s, where Steve Mann designed one of the first wearable computers. Professor Mann, who is teaching at the University of Toronto, built a back-packed head mounted display device, which was capable of providing text, showing video and which had the possibility to be programmable during movement like walking. The 1980s were also the beginning of the today called smartwatch. With the Seiko Epson RC-20[1] one of the first computerized watches was placed on the market, which could be connected to a personal computer and which was capable of loading programs. In the late 1990s and the beginning of the year 2000, two Linux watches were developed. The first one from Steve Mann was able to stream live video [Man14], the second one from IBM had a touch screen with X11 display graphics, wireless communication and was constructed as a complete miniaturized computer [NK02]. The following years many attempts were made to bring the smartwatch to the mass market, but no brand could establish itself. In 2014 and upcoming 2015 the large mobile operation system companies launch their products on the market and try to lead the smartwatch to success.

The sector of wearable computing spans many different sectors like the health and

[1] http://pocketcalculatorshow.com/nerdwatch/seiko-computer-watch-fun/

sports sector, the communication and entertainment industry, the military branch or even the clothing sector. In each of these diverse fields, many examples of wearable computers can be found.

Health Sector

In the health sector, the ambitions to support the human body with electronic devices date back to the beginning of computer technology. The ability to help people gain back lost senses like sight and hearing, has been an impulse to build devices like hearing instruments and visual prosthesis. Hearing instruments have made a huge progress from the beginning of electronic devices support in the 1960s till now. Despite these devices get smaller every year and the industry aims in the direction to ubiquitous computing, the computational power and the application skills increase. Nowadays digital hearing instruments are able to adjust the microphones according to the dialogue partner or filter out background noise [Sch04]. Also the health monitoring of patients and elderly people has become a huge field in the biomedical sector. Many papers show the monitoring of different health related values like heart rate and blood oxygen or fall detection and localization of dementia patients [BFG13], [SSSS13], [MLC14], [MSJ11]. Another part in medicine, which deals with sensing of smartphones and other microcomputers is called mHealth - short for mobile health. This field combines mobile telecommunication with the collection, management and evaluation of medical data [LTKY05], [JMOdG05], [DJRW03].

Sport Sector

Monitoring of the physical condition during sports is necessary for endurance sport to keep the body healthy and not overexert it. There were existing heart rate watches before the smartphone or different solutions existed like a paper from 2009 shows [LC09], where Lee attached a wearable sensor node to the shirt, which is capable of transmitting electrocardiogram and acceleration data to a server. But with the additional sensors and also the social aspect pushing to the foreground, many fitness apps are emerging in the mobile phone application stores. These apps offer mainly GPS tracking for running tracks or step counter for calories burning charts. With the upcoming Android wearables, Google will provide a new open health platform called Google Fit[2], which is providing APIs to make it easier for developers to address sensor data.

Entertainment and Communication Sector

Regarding entertainment and communication, wearable computers like smartwatches move from special customers market to the mass market. With smartwatches it is possible to communicate via calls and different message types or show various kinds of information. Introducing *Google Glass*[3], the American internet service corporation Google designed

[2]https://developers.google.com/fit/
[3]https://www.google.com/glass/

glasses including a display, which is capable of displaying video or pictures in the sense of augmented reality. This device should overlay the natural view with additional useful information like navigation data or historical data during sightseeing. With Google Glass it is also possible to take photos or videos, which also led to privacy concerns and became an outrage in the news.

Military Sector

Another form of wearables can be found in the military sector. Small electronic devices are meant to support soldiers on the battlefield, whether in form of information and communication accessories or built-in applications in helmets, weapons or clothing. Augmented reality and heads-up displays play a big role regarding military wearable computers. With the ability to see in the night, show additional information or even the possibility to aim around corners, wearables offer many new possibilities for improvement on the battlefield [ZJ02].

1.1 Motivation

The introduction of smartphones showed, that the combination of various sensors, a powerful processor and mobility can provide a variety of possible applications. Many fitness applications were developed in the past months and gain popularity in the different application stores. As the first smartwatches, which are suitable for the mass entered the market, this trend continues and provides new opportunities. To help elderly people to prevent accidents and react faster on emerging problems motivates to develop applications for wearable devices with the ability of a better health monitoring. Because elderly people and invalids will be more important in the future, regarding people get older and the constant increase of the imbalance between old and young people, it will be necessary to establish good health monitoring systems. With the multiple sensors in smartphones and wearables it is possible to address a wide range of health monitoring actions, like life threat detection and fitness application for a longer and more comfortable life. There already exist a lot of fitness and health care scenarios for smartphone and wearable sensors in the application stores.

Accelerometer Sensor

Accelerometer sensors are small and cheap and therefore one of the most common sensors in a smartphone. They are usually used for aligning the orientation of smartphone screens, but have also many other possible means of applications. The accelerometer sensor can be utilized for the recognition of activities showed in the papers of Bao [BI04] and Tapia [TIL04] or counting of steps or fall detection like seen in the papers Bourke [BOL07] and Chen [CKC+05]. The activity detection is a big topic in research papers, as well as the detection of falls, which is popular in the medicine sector. The data collected from an

accelerometer is also often combined with gyroscope or GPS data, to get more accurate results or to cover another key sensor topic like localization.

Heart Rate Sensor

While the first mass market smartwatches were usually only equipped with an accelerometer and a light sensor, the most upcoming smartwatches have a heart rate sensor installed. Monitoring the heart rate can be used for fitness activities like running and cycling or to control the heart beat of people with bad health or people who suffer on heart diseases. The sensor checks, if the pulse is normal and can warn in case of danger or emergency, which can help save peoples lives.

GPS Receiver

The GPS receiver is a common component in a smartphone and can be seen as a sensor of the environment. It is used for localization and tracking of GPS signals on maps, for example during running to collect the track data. While for outdoor localization GPS is the most accurate mean, it does not work indoors and it often has problems in street canyons or in narrow valleys, when the receiver can not reach all satellites or because of occurring reflection problems [PK10]. Indoor localization can be accomplished for example with fingerprints of RSSI signals of the wireless network or regarding RFID tag fingerprinting [BP00].

Other Useful Sensors

Beside the usual ones, modern smart devices nowadays often include other useful sensors like a temperature sensor, a pressure sensor, a humidity sensor or a photometer sensor. Addressing these sensors can enable possibilities for monitoring health data as well. With a temperature sensor it is possible to monitor the environment temperature and warn in case of a low or high temperature [YJS12] or distinguish between indoor and outdoor activation [KZX+11]. A hygrometer or humidity sensor can be used indoors to provide humidity control, for example it tells if the air is too dry. Pressure Sensors can be used for a fitness application during hiking as a weather forecast. In smoggy cities like Delhi, air pollution is a big problem and can lead to serious health problems. Attaching a sensor board to the smartphone it is possible to check the air quality and warn in case of high pollution [NVZB12].

Combination of Sensors

With the combination of the acquired data from different sensors it is possible to obtain better accuracy on desired results or combine various results to get a new output. Integrating an accelerometer, a sound and a photometer sensor, it is possible to identify the current environment regarding the movement, light ambiance and sound [MPR08]. With the combination of these sensors it is possible to figure out the current location of

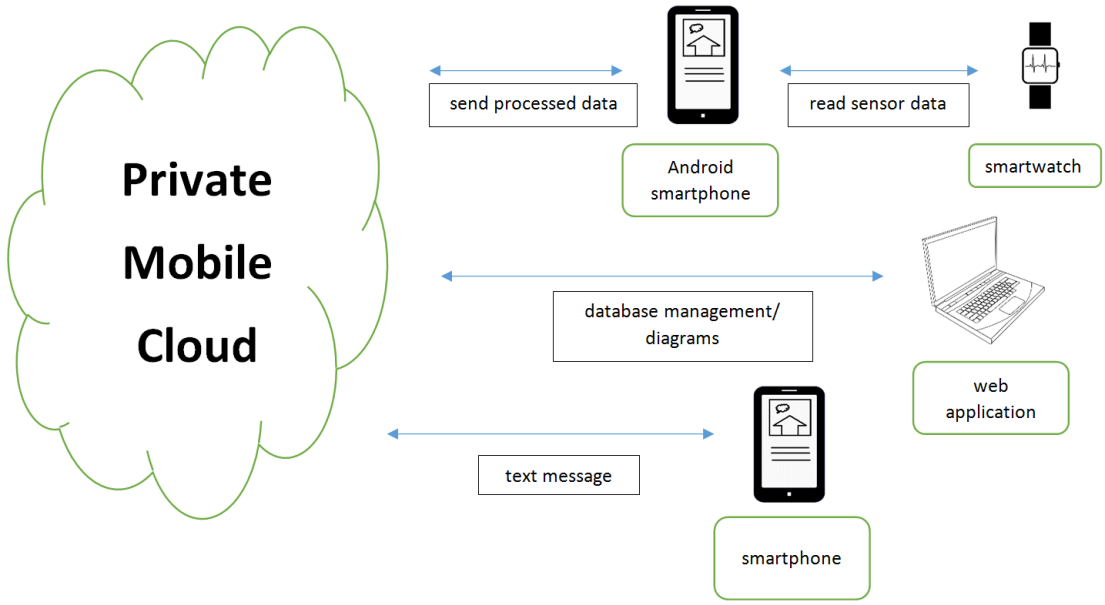

Figure 1.1: Overview of the project

a person wearing a smart device. A lot of different examples exist for the combination of accelerometers, gyroscopes and GPS receiver, like the illustration of a joggers running route and the running speed of the person or the identification of potholes on streets. Combining these sensors, it is possible to build up a health monitoring system, where the medical condition of a person can be observed. Combining a lot of these sensors and the possible sensor applications to a private health and fitness monitoring system is the goal of this research. The gathering and evaluation of the huge amount of sensor data for a health monitoring service requires a central system with a large storage and a universal reachable interface. To ensure this, a private mobile cloud is needed with an interface, which is capable of connecting any kind of a wearable device easily to the system.

1.2 Project Overview

The project combines a private mobile cloud with access from a personal computer over a web service and the connection of a wearable device linked to an Android smartphone. With the wearable device it is possible to send different kinds sensor data to the smartphone, evaluate it and then transfer it to the private mobile cloud. With the data on the private servers it is possible to further process it or display it on different device screens. This work should be the basis for a health and fitness monitoring system based on five major components: A wearable device, two smartphones, a private server and a personal computer.

In the figure 1.1 the overview of the project is shown, a more detailed view is presented in the Chapter 3, where the design is described. The wearable device, in the figure shown

as smartwatch, sends sensor data like raw accelerometer data to the smartphone. The smartphone, which is connected to the private mobile cloud, processes the data and sends it to the server, where it is stored on a database. With a web service it is possible to access the data provided by the server and display various diagrams representing it. There is the capability to monitor the sensor data and in case of an unexpected behavior or a reached goal alert a third person with a message to a smartphone. This happens via an XMPP service, which is linked to the phone and guarantees reliable connection. With an universal interface it is possible to connect different wearable devices to the smartphone and ensure a simple utilization.

1.3 Outline

This work is structured as follows, in Chapter 2 the related work to this project and their influence is shown. Used and similar algorithms are described and related systems and works are pointed out there, which use sensors from mobile devices. Also the used smartwatch in this work and related projects are described, as well as similar wearable computers. Chapter 3 explains the Design of the work in details including an overall design showing the coordination of all components, followed by Chapter 4, which describes the implementation of the work of each part in detail. The results of the project are presented in Chapter 5. In Chapter 6 the future possible work can be found and how the project could be improved, followed by the last Chapter 6, where the conclusion of this work is shown.

Chapter 2

Related Work

This Chapter shows related work like the processing of sensor data, fall detection recognition, possibilities of indoor localization or activity recognition. The used protocol XMPP, the different applications for the smartwatch *Metawatch*, which are already existing and general knowledge of cloud storage is also described here. The section of the wearables shows, which other wearable devices are available on the market.

2.1 Processing of Sensor Data

Receiving sensor data and the interpretation of it is a main aspect for further processing and useful utilization of the information given from the raw data. The multiple sensors provide a lot of data and information, but with the raw values it is difficult to gain knowledge out of it. To get the information out of the data, many different algorithms and approaches exist for data mining. First of all it needs to be determined how the raw signal is read from the sensor. The second part would be to obtain features from the raw data and analyze the similarities and the differences. The classification of the results from the processing of the data would be the next step. For this step, a lot of different machine learning algorithms exist, which can determine the desired behavior. A few of these algorithms and means are described in the following subsections.

2.1.1 *K*-Nearest Neighbor

The k-nearest neighbor is a classification system, which is used to assign specified objects to classes regarding its properties and similarities to the k-nearest neighbors [Pet09]. The training data is saved for each class, which is trained separately to differentiate the data. If an object is assigned to a class depends in the easiest solution to the majority vote of the k-nearest neighbors. To calculate the distance between the objects, different distance measurements can be chosen dependent on the features used to differentiate.

A popular metric for measuring the distance is the *Euclidean Distance*. The distance between two points in the euclidean space is a direct line between the two points. Regarding the Cartesian coordinate system, the equation would look like the square root of

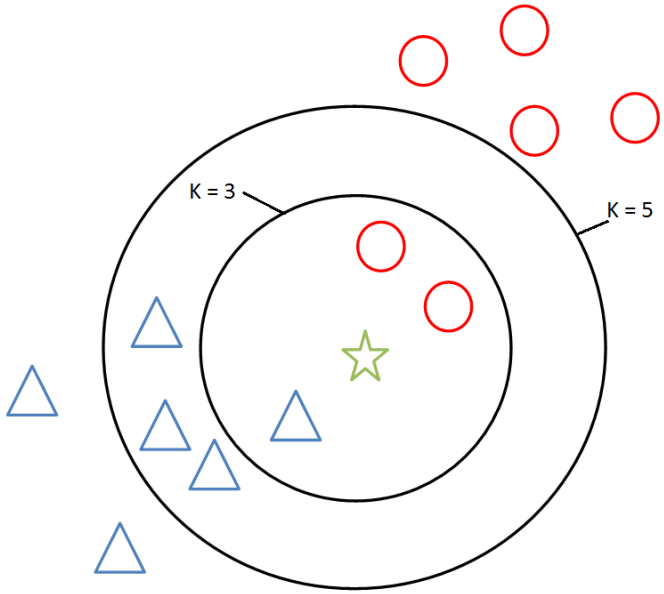

Figure 2.1: Shows the choice of a different k value

the sum of the difference between point a and point b squared. Other distance measures used for the k-nearest neighbor algorithm are the *Hamming Distance* and the *Mahalanobis Distance*. The *Mahalanobis Distance* is a unit less measure and can deal with different relations between the data sets and has certain advantages, when it compares different features. If the units would have equal variances, then it would be the same as the *Euclidean Distance*.

An important part of the success of the k-nearest neighbor algorithm is the choice of the k value, which is also shown in the figure 2.1, where the unknown sample can either be a triangle or a circle depending on the chosen k value. This choice is in the most case depending on the condition of the training data. If there is a lot of noise in the training data, which means that some points are taken out of their usual environment and a small k value is selected, the chance that the classification goes wrong is higher. The separation of the classes is not that obvious, which means that if the points of the different classes are very close, than it can happen that a wrong neighbor is closer than the correct neighbors. Choosing a rather high k value and there are only a few training values available, than it can be that some points, which are far away are taken into account and lead to a wrong classification in case of a majority vote. Another mean for getting better results can be achieved with adding a higher weight to the closer points and a lower weight to distant points. Whereas this method does not need time to learn and it has a good performance regarding a large training set, the testing phase takes a long time and needs a lot of memory with a huge data set. Another problem can be wrong classified training examples, which can weaken the algorithm.

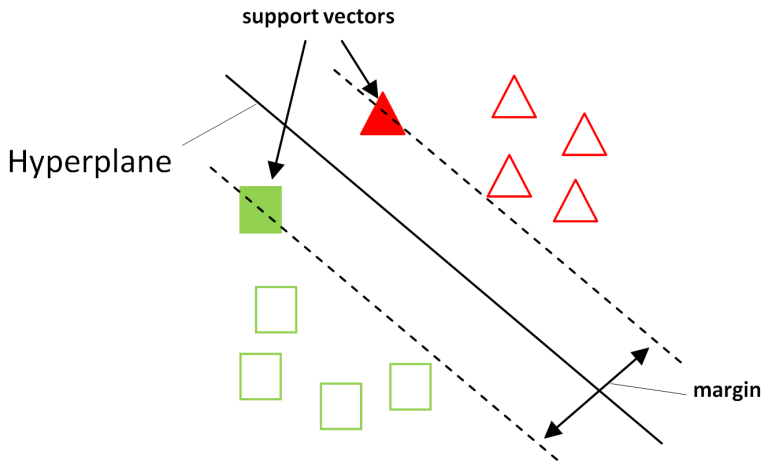

Figure 2.2: Shows the support vector machine with the hyper plane, margin and the support vectors

2.1.2 Support Vector Machines

Another popular mean of classification is the classifying method of *Support Vector Machines* [Cri00]. This classification model divides the objects into classes and tries to gain a largest possible margin between them to obtain clear differences. This is made possible by moving the objects into a vector space. Adding vectors to the objects the data points can be mapped in a vector space, where it is possible to separate the data with a so called hyper plane. The decision how the hyper plane looks like, depends on the closest vectors to this parting plane, these vectors are called support vectors. The hyper plane and the margin between the objects is shown in the figure 2.2, where the support vectors are colored. This is only possible to distinguish between linear separable data. To get a hyper plane for non linear separable data the vector space needs to be transferred into a higher dimensional space. The mechanism for this process is called *Kernel Trick*. In a high dimensional space it is possible to separate the classes, even if the data is not linear separable. The drawback of this transformation is the high computing effort, which is necessary to separate the classes and also to compute the back transformation. Another problem necessary to consider choosing this algorithm is the over fitting of the data set, which can lead to a wrong classification of the test data. Beside these disadvantages the algorithm delivers good performance, low processing time and memory space.

2.1.3 Artificial Neural Network

Artificial neural networks can also be used for classification. The algorithm tries to simulate a biological neural network in the way of building up a complex network with neurons, which can have special properties like a weight and directed connections [Yeg09]. The number of neurons, the weight and the connections can be adjusted during the learning phase of the algorithm. The information is transferred during the forward propagation to deter-

mine the output of the neurons and the whole network, whereas the failure function can be calculated during back propagation. For the binary classification with an artificial neural network or short *ANN*, the output neurons are rounded off to interpret it as a class. For more than two classes, more output neurons than classes are chosen, because otherwise the round off would lead to a proximity between the classes. With binary vectors it is possible to encode the classes with the highest possible *Hamming Distance*. For classification this algorithm has usually a good performance, which however is at the most not as good as the support vector machines.

2.2 Activity Recognition

The communication between humans and computers should be easy to use and not be an obstacle for users. With the help of activity or motion recognition it can be easier for a computing device to react on the users behavior in certain scenarios. Regarding a special activity, the computer can exclude certain functions, which the user does not need in this situation and therefore can focus on key functions of the activity. This context aware computing concept makes it possible to adjust the behavior accordingly to the activity. To get a good context awareness it is necessary to provide accurate activity recognition. Most of the papers use the data of accelerometers to recognize activities, in the earlier years with sensor boards [BI04], [Bao03], [RDML05], [TIL04], nowadays the accelerometers are often used from a common smartphone [BVU09], [Yan09]. Three papers, which are strongly related to this project are further described in the following subsections. The first paper from Bao and Intille is from the year 2004 and uses sensor boards for the activity recognition, the second one from Ravi deals with activity recognition and the context aware behavior and the third paper from Kwaspisz is from the year 2009 and it uses the accelerometer from a smartphone to recognize activities.

2.2.1 Activity Recognition from User-Annotated Acceleration Data

In this paper of Bao and Intille [BI04] it is shown, how to recognize various activities with five accelerometer sensors worn all over different body parts. With different algorithms and machine learning algorithms the sensor data is processed and 20 physical activities are evaluated. Before this work most of the training data occurred under laboratory conditions, which brings accurate results in a clean environment, but lacks under naturalistic conditions. In this paper the training data is collected mostly under semi-natural conditions where different users try to train during daily life conditions, which is at a disadvantage costly and time consuming.

The paper shows a figure with the sensor boards, which are attached on a males body parts like arms, hips, thigh and ankle. The signals produced from the accelerometers are shown in the diagrams on the right for the activities running and brushing teeth. As it can be seen, these signals look different between the activities. For the computation of the features of the accelerometer signals, the energy of the data, the mean, the entropy of

the frequency domain and the correlation were used to get the results. The window size for each period was taken with 6.7 seconds and the frequency of each sample was taken on 76.25 Hz. With these features and values precise results were achieved like in similar works before. For the decision, which result fits to which training data and therefore the classification of the activity, different machine learning algorithms were used like decision trees or Bayes classifiers. The most accurate results where evaluated for the decision tree. For the classic activities like walking, running, standing and sitting an accuracy between 85% and 90% could be reached and it showed that these results can compete with data used in previous laboratory environmental works. Because of the various behavior of persons it is necessary to train the data individually, to get good results. This work also presents, that many different activities beside the usual once can be detected with the use of accelerometers.

2.2.2 Activity Recognition from Accelerometer Data

In the paper from Ravi called *Activity Recognition from Accelerometer Data* [RDML05] it is shown how to recognize activities with the transmission of data from an accelerometer applied to a hoarder board. It describes how the data was collected, the features used to determine the activity and the interpretation and classification of the gained information from the data. In the experiments, the data was sent via Bluetooth from the hoarder board with the applied tri-axis accelerometer to an HP personal computer. For the different activities the data from the signal of the accelerometer was collected for each axis and time. In the paper it is distinguished between eight different activities like walking, standing or climbing the stairs up and down. The collected samples are labeled to the various activities. For the extraction of the features, the following means were calculated for each axis, the mean, the standard deviation, the energy and the correlation of the signals. For the classification of the training samples, several different classification methods were used and compared like decision trees and tables, *K-nearest neighbor* and support vector machines. Not only the classification of the different methods was evaluated, also the performance has been rated. Four different settings were applied and evaluated by the different classification methods. The results of this work showed that the best performance was achieved by the plurality voting, which had the best performance rating in three of the four settings. Plurality voting is a classification scheme, which votes for the majority of plural classifiers and therefore results in a high accuracy measure. For the different recognition, the standing activity is the most recognized activity, followed by vacuuming and walking. The paper further concluded that the combination of classifiers delivers the best results and the energy feature is the attribute, which provides the worst information.

2.2.3 Activity Recognition Using Cell Phone Accelerometers

Kwapisz describes in his paper [KWM11] a system, which performs activity recognition of physical activities with an accelerometer from a mobile phone. The work in this paper is divided into three parts, the data collection, the feature generation and the transformation

of the data and the definition of the activities. For the collection of the data it was necessary to get a high number of people attending to collect data, to get a large basis of reference data to work with. 29 volunteers were selected, who carry a mobile phone in their pocket and collect accelerometer data for the different activities. The data for the accelerometer was collected each 50 ms to get 20 samples in a second. To compare the various accelerometer data samples, a 10 second period was concluded to get the features needed. For the features, which are used to recognize different activities, six different basic types were defined. The six features are:

- An average value of acceleration for all axes
- The standard deviation for the three coordinates
- The average value for the distance between the median and each value
- An average square root of the sum of the square values of each axis
- The time between two peaks
- The range from a maximum to a minimum value

With these features it is managed to differentiate between six predefined activities. In the paper the following activities are distinguished between walking, jogging, the ascending and descending of stairs as well as sitting and standing. These activities were selected, because they are widely used during the daily process and their accelerometer data include periodicity, which helps to distinguish between the activities. Each of the activities has a typical characteristic regarding the values of the axes, for example walking and jogging can be distinguished with the different high and low peaks and the time between the peaks, for jogging the peaks are usually higher and the interval of the peaks is shorter. Sitting and standing do not have periodical behavior, which separate themselves from the others. These two activities can be differentiated with the average value of the axes.

The results for the activity recognition showed, that for the most activities a high recognition can be achieved. For walking and jogging a percentage rate of over 90% could be reached, whereas the activities for climbing stairs are the more difficult types of activity to recognize. The difference between ascending and descending stairs seems not to be easily distinguished with only the results of an accelerometer sensor. For the classification of the results, three different techniques were used, decision trees, logistic regression and neural networks. The best percentage of recognized activities was covered by the neural networks.

2.3 Fall Detection

Fall detection is a popular topic in the field of the evaluation of accelerometer data sent from a small mobile device. Especially in the health care sector this topic is relevant for

elderly people and can provide fast aid. With the decrease of the size of mobile devices it is possible to sense the environment in a more convenient way and prevent people of harm and alert in case of emergencies. Already in the year 2003 a paper about a wrist worn fall detection system has been published by Degen [DJRW03]. In the paper it is described how a wrist worn system for detection of a heavy fall is implemented, which sends an automatic alarm to a call center. This system is similar to a watch and has the capability to make a data analysis of the accelerometer and send the information with a wireless communication. The system works with three different calculations, the integration of the norm of the three axis, the integration of the norm of all three axis after the subtraction of the static acceleration as well as the integration of each axes and the following normalization. With this three components it is possible to detect a fall. As the evaluation of the system shows in this paper, the best result was detected with a forward fall, the worst with a sideways fall.

A more generic way on how to detect falls has been published by Noury in the year 2007 [NF07]. In this paper a survey is shown, which deals with the different fall detection algorithms, the evaluation of sensors and systems. This more theoretical approach attends to compare different papers from the literature about the existing fall detection systems and their approaches and defines a theoretical standard for fall detection. In the year 2009 the first fall detection systems with the utilization of an *Android* smartphones were published. The paper *iFall: An Android Application for Fall Monitoring and Response* [ST09] from Sposaro uses an *Android* smartphone with an accelerometer to detect the fall, which then sends a notification to the user. If the user does not respond to this message an alert is sent to a predefined number per a short message service. For the fall detection, Sposaro differentiates the gravitational values of the accelerometer and detects a fall, if two defined thresholds are exceed during a certain amount of time. In case of a false positive, the user can cancel the alert during a specific duration. The thresholds are adjusted, if the user carries the phone in the pocket or carries it in a more movable part like the hand.

Another paper, which deals with fall detection with a smartphone is called *PerFallD: A Pervasive Fall Detection System Using Mobile Phones* from Dai [DBY+10]. In this paper from the year 2010 also an *Android* based system is used to detect falls. A prototype was implemented and the results were evaluated on detection rate and performance and compare it to other systems on the market. The following three papers from Chen [CKC+05], Bourke [BOL07] and Li [LSH+09] are described more detailed in the subsections.

2.3.1 Wearable Sensors for Reliable Fall Detection

The paper from Chen from the year 2005 [CKC+05] deals with a sensor mote system in a wireless network to gain a higher living standard for elderly people. This work of an indoor fall detector handles the detection with a small device, which is worn on the body and several connected sensor motes distributed all over the environment. The device with the accelerometer is worn on the waist to get a stable accelerometer signal, which

does not change the axis of the accelerometer. The sensor motes, which are spread in the environment are used to gain connection to the accelerometer mote all of the time. The data, which is sent from the accelerometer can be either processed on one of the environment sensor boards or to reduce overhead, send the data to the main station, which possess more resources than a single mote. A strong indicator for a fall detection using accelerometers is the magnitude of each of the axis. In this paper the author states that only the magnitude does not conclude a fall. Therefore it is suggested to calculate the norm of all three axis. This threshold based approach can provoke many false positives, if the user does a more accelerating movement like running or jumping. The solution to this problem is corrected in the observation of the orientation of the axis. If the position of a person changes very fast, also the values of the different axis increase or decrease rapidly. While compared to other systems, the approach lies mainly on the threshold, the shift of the orientation is the main aspect in this paper, which shows better results.

2.3.2 Evaluation of a Threshold-Based Tri-Axial Accelerometer Fall Detection Algorithm

This paper written in the year 2007 from [BOL07] is more concerned with the algorithm to better detect falls than to build a system out of it. It is examined, how the daily activities can be distinguished from a fall. Therefore an accelerometer was attached to the body and eight fall types were investigated. The algorithm was distributed into two parts, the upper and the lower fall threshold. For the activities to distinguish different sitting down and standing up situation were chosen as well as walking. Because the work is conceived for elderly people, situations like running or jumping are not considered. As the author of this paper could show that the usual activities of an elderly person do not feature many high or low peaks in the accelerometer axis. The graph of an slow movement usually stays very flat. In the paper Bourke showed in the experiments that all of the selected daily life activities could not be confused with a fall, so that there were no falls positives in this work. The work also concluded that the sensor worn on the trunk of a person seems to be the most effective part on a body to differentiate between a fall and other activities. The algorithm to detect a fall was kept simple in order to maintain low processing power to the small device, which is worn on the body to on one side keep the power consumption low and on the other side keep the device as small as possible that it is not conceived as annoying. Further it is proved that an accelerometer, which is worn on the body and connected to a wireless network or other electronic devices monitoring the data can be used as an fall detection system for elderly people.

2.3.3 Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information

Li wrote in his paper from 2009 [LSH+09] about a fall detection system, which uses gyroscopes and accelerometers for detection. Whereas other similar fall detection systems only rely on the use of an accelerometer, this work uses two accelerometers and a gyroscope

to differentiate between activities and a fall, where the gyroscope works as an additional resource to prevent false positives. A figure in the paper shows the sensor nodes and where they are placed on the body parts. The algorithm for the fall detection has three phases: the first phase monitors the person and differentiates between static and dynamic activities. The key part of the second phase is to determine, if the static posture is a lying posture. In phase three it is then decided about the transition before the static lying position. The question, which has to be solved there is about the process of the transition, if it was on purpose. Has the transition been made intentionally, the dynamic movement should not be detected as a fall, otherwise a fall has occurred.

As the accelerometer delivers the best results for distinguishing between a movement and a posture, the gyroscope is mainly needed for the recognition of the posture of the person, whereas the combination of these two sensors is needed for the differentiation between a transition on purpose or an unintentional transition. As the information gained from the sensors is rather high, the detection of the fall still needs to consider the different behavior of individual users. The work shows that it is possible to reduce the mismatched samples and improve the detection of a real fall, but it also reached its limits differentiating between special cases like falling against a wall and being placed in a seated position.

2.4 Localization

Another topic regarding health monitoring and sensor evaluation is the localization of people using sensors from smartphones. There are several different approaches to localize people, for the outdoor localization one of the most precise solutions regarding the usage of a small device like a smartphone is the GPS localization, which is also described in the paper of Bulusu from the year 2000 [BHE00]. Another option would be the wireless network module, which can be used in cities were many wireless hot spots exist. This approach is explained in the paper from Paek [PK10], which is further described in the following subsection. Indoors for example, with the wireless network module, the accelerometer and the compass from a smartphone it is possible to track the path and locate people indoors, like it is shown in the paper *Zee: Zero-Effort Crowdsourcing for Indoor Localization* from Rai [RC12]. In this paper it is described how it is possible to locate a person using a smartphone indoors. The precondition for this indoor localization is a map, which defines the rooms, corridors and walls in between. Using the accelerometer, compass, gyroscope and the Wi-Fi of the smartphone it is possible to locate the person, after walking through the building, without the knowledge of the initial position. The location of the person is separated in four different sections:

- A step counter and the estimation of the orientation
- Particle filters to estimate the users stride and walking attributes
- Back propagation for a more accurate performance

- The wireless initialization of the particles for a faster approximation of the location

With these four principles, the combination of sensor information and the signal strength of the wireless data Rai shows a high accurate way to locate people indoors.

Another indoor localization paper from Patel shows how to locate a person using the buildings power line [PTA06]. This work deals with the wire network in the walls and the detection through receiving a tone from the electrical wires. With the various signal strengths it is possible to build a map and localize the correct room with a k-nearest neighbor algorithm. Another useful technique for an approach as a prerequisite for a localization is to rapidly gain knowledge about the users walking direction. A technique for this technique is described in the paper of Roy [RWR14]. In the work a prototype is shown, which is able to detect the direction, where a person heads to, during a few specific steps. It is evaluated how the walking movement influences the smartphone and how the sensors behave and inform the user of the heading direction. Therefore the system is divided into three sections, the analysis of the walk, an local estimation of the direction and a global walking direction. While the local estimation of the walking direction works with evaluating the accelerometer and the gyroscope, the compass is used for estimating the global direction. Another three papers, which achieve good results regarding localization are described in the following subsections in detail.

2.4.1 RADAR: An In-Building RF-based User Location and Tracking System

The paper *RADAR: An In-Building RF-based User Location and Tracking System* [BP00] from Bahl from the year 2000, was one of the first papers dealing with the indoor localization regarding the radio frequency technique. The system, which has been implemented, deals with tracking the user inside a building regarding the signal strength of sensor boards. With the information of the signal strength of different wireless network stations it is possible to build a map of the signals. For the analysis of the test data six different analyses are performed. For the basic analysis of the data a random point is selected and a nearest neighbor analysis is done, which simulates a localization. With this step the error probability is too high for the author, therefore five different approaches to improve the basic analysis is done. The first basic step only took one neighbor, so the first improvement was the k-nearest neighbor approach with more neighbors taken into account. The second improvement uses only the signal strength with the highest signal data sets. Further it was analyzed how the number of data points influenced the results as well as limitation of the samples. The last improvement was regarding the orientation of the user, where different directions of the user were tested. To this empirical approach, a second method was implemented regarding the propagation of the radio signal. The used propagation model for this work was the *Floor Attenuation Factor* model, which is flexible regarding building structures. The model was modified to specify the signal strength from the sender to the receiver regarding walls and not the floor. This propagation takes the number of walls into account, how many walls stand between the receiver and the sender. The decision of

considering the factor of attenuation depends on the number of walls. As the results in this work showed, the empirical approach performed more accurate, but it needs a higher preparation effort than the propagation model.

2.4.2 Energy-Efficient Rate-Adaptive GPS-based Positioning for Smartphones

In this paper [PK10], Paek and Kim show a solution for localization in urban areas with several different approaches like radio signal strength, acceleration and Bluetooth to unload the high efficient GPS module in case of a weak signal strength. As on the one hand GPS has a very accurate positioning distance outdoors, on the other hand it is not very power efficient and in densely populated regions through reflection from buildings or shielding of obstacles it can result in inaccuracy. A figure about the accuracy of the different signals for GPS, WPS and GSM is shown in the paper, which illustrates the differences between these three modules. Paek discusses in the paper four technique parts to perform localization with low usage of the GPS module. Determine the movement of the person regarding the accelerometer, activation of the GPS module only in certain position and velocity, which are previously determined, information of the signal strength regarding the GSM module and the use of the Bluetooth communication to determine the information of the position are the main aspects of this work. One of the main points in this work is the so called '*Space-Time History*'. This approach deals with the movement of a person, the speed and the estimated ratio of the activity, to determine whether to use GPS or not. It therefore builds up a history of older movement patterns. With these approaches, the evaluation showed a significant improvement in the power consumption of the phone and a high reduction of the usage of the GPS module at a high accuracy of localization.

2.4.3 Precise Indoor Localization Using Smart Phones

In the year 2010 Martin wrote a paper about indoor localization with the usage of a smartphone [MVFB10]. For the localization several different components are used from the smartphone like the radio signal strength of the wireless and the cellular radio as well as the acceleration signal data from the accelerometer and data from the magnetometer. Regarding the *Wi-Fi* a radio map of the rooms and aisles is prepared to get a fingerprint for the location. Reading the *RSSI* values of the *Wi-Fi* during the test run it is possible to match the current signals with the signals from the map. To improve the accuracy of the fingerprinting technique, the accelerometer and magnetometer is used. Because the radio signal strength of the wireless nodes is depending on external influences like different obstacles or the body posture, it is necessary to provide the same orientation than used in the collected data from the radio map, which is handled with the accelerometer and magnetometer. For the cellular radio signal, this paper only makes an assumption how to enhance the system, which would result in a clear decision, if the *RSSI* signal is not accurate enough. In the conclusion of the paper the author claimed to get an accuracy of up to 1.5 meters for the experiments.

2.5 Cloud Computing

The data storage in the cloud has been a huge topic in the computer sector in the recent years. Although it has many advantages to store the data on servers connected to the internet, the protection of privacy and legal and judicial gray areas are still a drawback for users and companies to use these services. The general idea of cloud computing can be already dated back to the *90s*, where the telecommunication industry used a virtual private network - *VPN* - to increase the bandwidth and therefore be more efficient. The term of a cloud and how it is known in today's context, was introduced in the year 2006 by *Amazon*. The basic idea behind it is to outsource server infrastructure, storage and processing capacity to a private centralized or even public infrastructure. Most of the bigger companies use a private cloud to manage the huge amount of data and also the most of the world's biggest computer companies like *Apple*, *Google* or *Microsoft* tend to outsource the infrastructure and store the data in the cloud. The cloud offer can be divided into two parts, as defined by the federal technology agency NIST - *National Institute of Standards and Technology* [MG11], the cloud as a service model and as a deployment version. When the cloud is used as a service it is distinguished between, *Software as a Service*, *Platform as a Service* and *Infrastructure as a Service*. On the deployment models it is distinguished between the *Private Cloud*, the *Community Cloud*, the *Public Cloud* and the *Hybrid Cloud*. Whereas the private cloud is managed by the company itself or outsourced to a third party company and only used by this single company, the public cloud is provided for open use over a public network. Outsourcing data to a cloud service raises new considerations regarding security. At least the following points should be considered, when saving private data in the public cloud:

- Loss of the data
- Unauthorized access to the personal data
- Malicious activities happening on the outsourced infrastructure
- Legal and contract condition problems

In the year 2011, Bradshaw published a survey about the terms and conditions of the different providers offering a cloud service [BMW11]. The procedure of the analysis of the Cloud services was handled as follows. At first the cloud offers were classified, then the legal documents like *Terms of Services* or the *Privacy Policy* were analyzed as well as contract properties like applicable law, contract terms or liability. As seen in the survey it can happen that the jurisdiction and applicable law conflict with local laws, where the customer is located or in terms of data integrity or preservation, where the companies try to pass on the responsibility to the customer.

2.6 MetaWatch

The smartwatch *MetaWatch*, which is used in this work, was developed as a project on the *Kickstarter* platform and claims to be a developer friendly environment. Since the beginning in the year 2011 different projects were released and published on a developer site[1]. Two of these projects are described more detailed in the following subsections, the third subsection describes a paper regarding activity monitoring with a *MetaWatch* smartwatch.

2.6.1 Multimedia and Room Light Remote Control

These two projects, one about a multimedia remote control and the second about room light remote control are both similar designed. The multimedia remote control was developed to remotely control multimedia players on an *Apple Mac* device. The graphical user interface was built as a static image rendered with a graphical designer program. It shows the remote control necessary to control a multimedia player. For each button on the watch, a control unit is assigned. The remote control works with the wireless audio streaming solution *Airfoil for Mac OS* and the therefor supported *reemote* as a server. To trigger the actions it was necessary to implement *HTTP* requests using *JSON*, which was realized with the command line tool *curl*. Mapping the buttons to the shell commands it is possible to transfer data with *URL* commands to the *reemote* server and manage the controls of a multimedia player.

The second project regarding the room light remote control works similar using an open source project for ambient intelligence and home automation called *openAMI*. This project also uses shell commands triggered from the buttons, which were enabled on the smartwatch to transfer data to the *openAMI* server. The used graphical user interface is also a static picture as described in the previous project.

2.6.2 Game Development on the Smartwatch

Another prototype found on this developer page is the illustration of a variation of the computer game *Space Invaders* running on the *MetaWatch* smartwatch. For this project, at first the menu of the system was modified to add a new item to the system and reorganize the system structure to easily navigate. The second part of the project was the implementation of a variation of the game *Space Invaders*. The game has a simple principle, where the user has to move left and right to avoid getting hit by enemy bullets and otherwise shoot enemies to win the game. The navigation and shooting works with the buttons of the smartwatch.

[1] https://sites.google.com/site/metawatchdev/software/

2.6.3 Sensor Requirements for Activity Recognition on Smart Watches

In the year 2013 Bieber wrote a paper about Activity Recognition regarding the usage of a *MetaWatch* smartwatch [BFG13]. The paper deals with the requirements needed to recognize activities using the sensors from a smartwatch. It describes methods to detect activities like walking or running and more static activities like sleeping or resting and the challenges regarding the activity monitoring. For the definition of the requirements of a sensor in a smartwatch, the author mentions the desired applications:

- The monitoring of activities in the fitness sector
- The monitoring of activities in the health care sector
- Sleeping phase recognition or fall detection
- Recognition of gestures

To determine the activity with the accelerometer a new feature was introduced, the so called *Activity Unit*. The *Activity Unit* is defined by the equation 2.1, were x, y and z are the values of the sensors and N describes how often the values are read out. The mean values of each axis represents the average calculation. The *Activity Unit* is a normalized value and it describes the acceleration level during acceleration changes.

$$1AU = \frac{1}{N}\sum_{n=1}^{N}\sqrt{(x_n - x_{mean})^2 + (y_n - y_{mean})^2 + (z_n - z_{mean})^2} \tag{2.1}$$

In the paper several applications are described, which have been implemented to show the activity recognition with a smartwatch. The detection of wearing the smartwatch, the sleeping detection and a so called heart rate detection application were implemented. Regarding the heart rate detection, the acceleration of the accelerometer has been measured, when the smartwatch was hold to the chest of a person. Another industrial application is shown, which guides a person wearing the smartwatch through triggering vibration signals. The outlook given in this paper expects the smartwatch to provide a huge field of technical support for the people in the future.

2.7 Wearables

Wearables can be defined as small computational devices, mostly including sensors, which are worn directly on the body, over clothing or even attached to clothing. The wearable computers usually collect sensor data and processes it or sends it to other electronic devices via wireless communication. Turning on the device is not necessary for wearables, because body and computer constantly interact during the whole time. This definition can also be found in the book of Steve Mann [Man14]. As wearable devices can cover a huge field of computing devices worn on the body, the most important wearable today is the smartwatch.

2.7.1 Smartwatches

Smartwatches become more popular nowadays as the large manufacturers on the community market tend to develop smartwatches and operating systems or design simple extensions to the smartphone devices. The history of a smartwatch, as it is known in today's definition, can date back to the year 2000 were the *IBM Linux Watch* was developed. Nowadays the market is full with different smartwatches from various companies. As in the early years the smartwatch was designed as an independent product, the products on the current market are usually accessories to smartphones with limited capabilities. Whereas the most manufacturers concentrate on high resolution touch screens and a powerful processor, the battery of such a device can not last very long with regular use. To reduce the processing time of a smartwatch, the main use is to show notifications from the smartphone like calendar notes, short messages or weather conditions. Equipped with different sensors, the smartwatch can also be used for different applications like location tracking with *GPS*, fitness tracking with an accelerometer or even measuring the blood pressure.

IBM Linux Watch

The prototype of IBM's *Linux* based smartwatch from the beginning of the 2000s was one of the first smartwatches, which combines a feature-rich operating system, which had at this time a powerful hardware, wearable on a wrist. In the journal article *IBM's Linux Watch: The Challenge of Miniaturization* [NK02] from 2002, Narayanaswami reports about the design and implementation of the smartwatch.

The article reports about the challenges, which were faced during the design, implementation, prototype testing and development phase. Although the project was never launched on the market, the prototype attracted much attention. It was possible to view calendar entries, browse through the address book, get notifications and wireless connect to a personal computer or other devices, which had Bluetooth enabled. Two prototypes exist with different displays, which were necessary during implementation, because the favored *OLED* display, which meets their needs of usability and power consumption needed to be designed at first. In the meantime an *LCD* display was used for implementation and testing. For the processing unit an *ARM* processor was used, which did not need much power and was smaller than a standard processor. For the memory the largest possible solution at this time was achieved with 8 MB RAM. The processor was also equipped with an infrared module, which was not used because of usability reasons, therefor a Bluetooth module was added, where it is possible to connect to other devices without the direct interaction of the user. For interaction a touch screen with four different sectors and a crown, which is also used for analog watches was used. With the operating system *Linux*, the prototype was equipped with an open source *OS*. For the graphic display the graphics library *X11* was used, which was claimed as stable and generally complete. The prototype

is shown in the figure on the web page, which shows a demo of the output of the Linux shell command line.

On the Expo of *LinuxWorld* the prototype was presented for the first time. Shown was a shell command line and a watch display. It was possible to send commands via a serial interface with a personal computer and execute Linux commands on the shell. The watch received a lot of attention. Although the prototype was promising, it also faced power consumption and display problems with the *OLED* display. In the year 2002 the development of the wristwatch was discontinued, the reasons were not communicated to the public.

Pebble Smartwatch

As well as the MetaWatch, the *Pebble* smartwatch was also a project on the *Kickstarter* platform, which has been funded by the crowd. The crowd funding project started in the beginning of the year 2012 and was already funded in May. The operating system is a free real time operating system, short *RTOS*. It is possible to communicate via Bluetooth with an Android and a *iOS* operating system. It has 128 KB of RAM and uses a flash storage to store the watch applications. For the display it uses an 'e-paper' display and for the input controls it offers four buttons. Equipped with a magnetometer, an accelerometer and an ambient light sensor, the smartwatch offers a variety of sensors to address. The *Pebble* watch can be seen as a direct competitor to the *MetaWatch* as it is also a developer friendly environment, which offers a free software development kit for developers on their homepage[2]. On the homepage many different guides, a lot of documentation, tools, libraries and also a forum for developers can be found. The large community provides a huge repository of third party applications for the smartwatch and the community is still very active in contrast to the *MetaWatch* developers forum. A picture of the smartwatch is shown in the figure on the web page.

In the paper *Outsmarting Proctors with Smartwatches: A Case Study on Wearable Computing Security* Migicovsky uses a *Pebble* smartwatch to show an application, which is possible to cheat on multiple choice tests [MDRH14]. It is shown, how to cheat during a multiple choice test while cooperating with other students with a *Pebble* smartwatch. Therefore the cloud service is used, together with a smartphone connected to a smartwatch. A prototype was developed, which could receive messages on the smartwatch concealed as encoded pixels in a watch face. The cloud application is used to save exam questions and distribute it to registered users. The smartphone works as an interface between the cloud application and the smartwatch display. The paper raises security concerns regarding smartwatches, which could lead to cheating in exams or help to defraud in a casino.

[1]Source: http://bowyerinsider.blogspot.co.at/2011/04/citizen-watch-and-ibm-research.html
[2]https://developer.getpebble.com/
[2]Source: https://blog.getpebble.com/

Android Wear

With Android Wear, *Google* introduced a platform, which is able to extend the smartphone operating system to wearables and specifically to smartwatches[3]. Many smartphone manufacturers started to build a smartwatch regarding the Android Wear ecosystem for wearables like *LG*, *Samsung* or *Motorola*. From Android version 4.3 onward it is possible to communicate through Bluetooth with a device featuring Android Wear. The *API* offers the synchronization of notifications like text messages or emails, show different information like weather, stock prices or other useful data from the internet, or access the sensor data from the smartwatch for personal applications. On the developers site *Google* offers guidelines and training to start implementing watch applications, as well as tools and services. To operate the operating system Android Watch smartwatches usually implement a touch display and also the API supports a voice control. The design is kept simple and flat, the touch handling works with wipe gestures. The Android Wear API also supports round layouts, which results in a more watch like appearance as it can also be seen by the Motorola smartwatch *Moto 360*.

2.7.2 Google Glass

In the year 2012 *Google* developed a miniaturized computer, which is attached to glasses and provides a display showing information to the user, called *Google Glass* [4]. The used technique can be classified as augmented reality, like it has already shown Steve Mann in his early works [Man97] or later [Man14]. The operating system used for the small computer is based on *Android* and uses a powerful hardware with a dual core *SoC*, 2 GB of RAM and 16 GB flash storage, for the display it uses a glass prism to project the information. Including wireless communication and Bluetooth, the device is capable of accessing the internet and connecting to other *Android* devices. The input works with voice control and additional there are buttons, which can be used to switch through the different display screens. The head mounted display also includes a camera to record video seen by the wearer of the glasses, which resulted in controversies regarding privacy protection and prohibitions in different facilities[5].

2.7.3 TI SimpleLink SensorTag

With the Bluetooth *SimpleLink SensorTag*, Texas Instruments developed a wearable device capable of sensing the environment with its different applied sensors[6]. The sensor tag is designed to easily access the sensor data like a temperature, pressure and humidity sensor and an accelerometer, gyroscope and magnetometer, which is able to communicate with an *Android* or *iOS* smartphone or other Bluetooth devices. Texas Instruments therefor offers the source code for different platforms and various user guides to easily

[3] http://www.android.com/wear/
[4] http://www.google.com/glass/start/
[5] http://www.theguardian.com/technology/2013/mar/06/google-glass-threat-to-our-privacy
[6] www.ti.com/sensortag

implement applications connecting to the sensor tag. The tag is powered by a cell coin battery, it applies two buttons, which can be used to trigger an action like an alarm and with *Bluetooth Low Energy* it uses a power efficient protocol to communicate to other devices. Designed for health and fitness monitoring, the tag is also used for educational programs to raise awareness on sensor driven applications.

Chapter 3

Design

In this chapter, the design for the work is presented, which is divided into an overall design of the whole project and the relation of the different parts, the design of the application for the smartwatch, the app design for the smartphone, the design of the processing of the data as well as the private mobile cloud design. The overview of the private mobile cloud is again separated in three different parts, the server, the database and the *Android XMPP* service design.

3.1 Overall Design

The project realized in this work consists of a private personal cloud service, a smartphone connected to the cloud and a smartwatch paired with Bluetooth to the mobile phone. To be able to save the data permanently and not lose it depending on a broken phone or new setup, the design decision was chosen to transfer the data to a private cloud. Another aspect of the server storage is that the data can be accessed by a third person to manage and get an overview of the data. The third aspect of a cloud service is to have the ability to send an alert to a third person smartphone in case of a certain trigger. To ensure the integrity of the data, the decision was made to keep the server private and not store it in a public cloud. The private mobile cloud consists of a server, which manages the incoming and outgoing connections, a database for storing the relevant data, a service to notify a third person smartphone and a service to show the stored data via a web browser. To get the various components to a working system, it needs a well conceived design, about the data flow, how the different parts work together as well as the purpose of the whole system. The figure 3.1 shows a project overview of the overall design, where all relevant components are shown and how these components are connected to each other. It is shown, that the smartwatch is connected with the smartphone, which happens via Bluetooth. The smartphone sends data to the private mobile cloud, this is working with a web service capable of processing the data and storing it on the database. The data can also be evaluated on the server side and an alert can be sent with an XMPP service to a smartphone. With a web browser it is possible to access the data and display a graph.

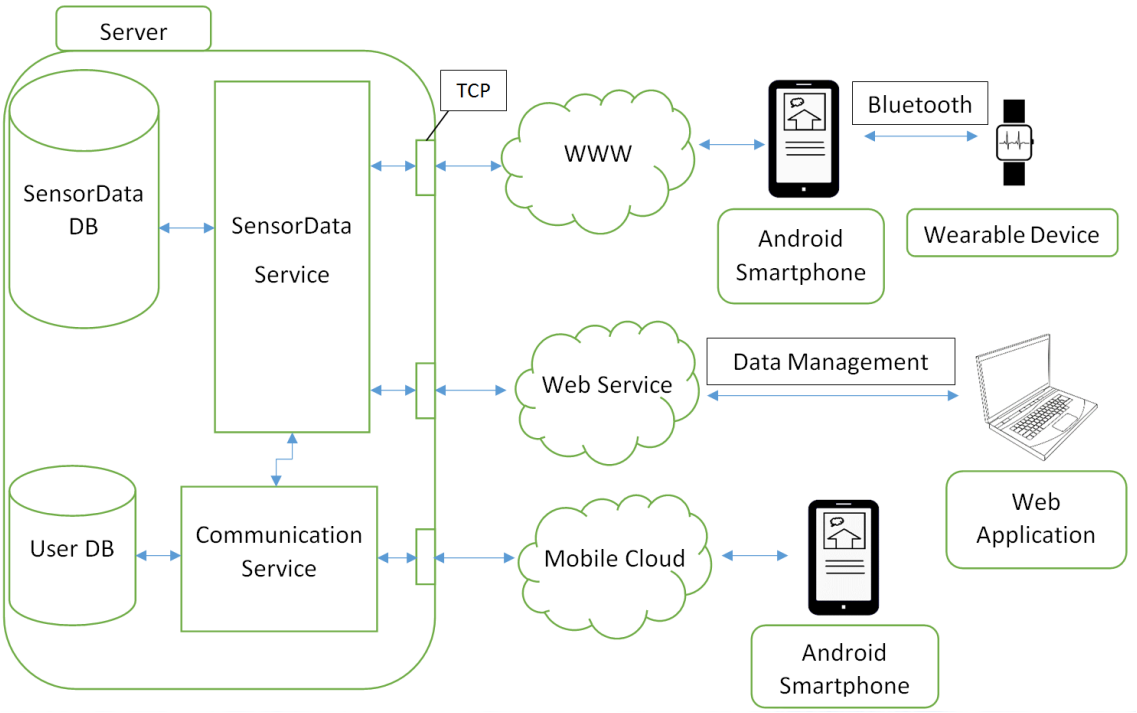

Figure 3.1: Shows overall design overview of the project including all components.

For the overall design of the project the sustainability of the system needs to be considered. If a part of the system will be replaced, it has to be possible to substitute it and ensure the integrity of the system. Therefore the overall system needs to have universal interfaces to be able to connect between the different components, for example another smartwatch or a different sensor device connected to the smartphone. For the current system a *MetaWatch* smartwatch and an Android *Nexus S* smartphone is used, these two components should be able to be replaced with other Android smartphones and smartwatches compatible to the operating system. Also the sensor data received from the smartwatch should be accessed in a uniform way to provide the support of different data, than to the actual implemented accelerometer data.

3.2 4 + 1 View of the Project Architecture

To show the design of the project, the 4 + 1 architectural view is used like it is described in the article *Architectural Blueprints - The '4+1' View Model of Software Architecture* [Kru95], which should help to understand the structure, the layout and the relations of the whole architecture. In the article it is shown, how to describe the architecture of a software project with different views and therefor provide a good overview of the system. The figure 3.2 shows how the different views are associated and which view belongs to to which interested party. The logical view presents the functionality information for the end user, the process view takes care of the dynamic processes of the system, the development

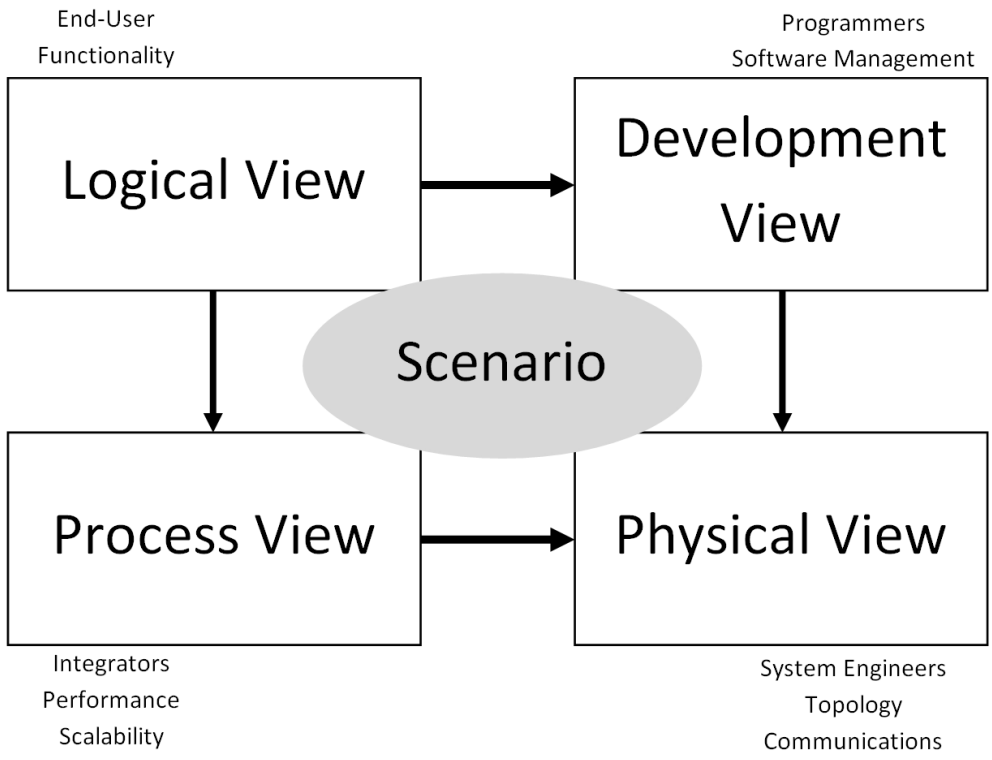

Figure 3.2: Shows the 4+1 architectural view model as described from the article [Kru95]

view provides an overview about the software management for the developer and the physical view maps the software components to the hardware. For the four different views a few use cases or scenarios should exist, describing the use of the different models in combination.

3.2.1 Logical View

In the logical view of the system, the functionality is shown, which is necessary for the end user. Therefor an interaction diagram is designed. The interaction diagram shows the communication of the user with the system and helps to understand the functionality of the end product for the user to operate with. For the design of this project, two kinds of users are introduced:

- Test users (one or more) wearing the smartwatch and operating the activity monitoring system
- Administrator or trainer, who maintains an overview of the users, supervises the reached goals and receives an alert in case of a specific event

In the figure 3.3 the sequence of the activity of a test user is presented. It shows how the end user, in this case the test user, connects his smartwatch with the smartphone

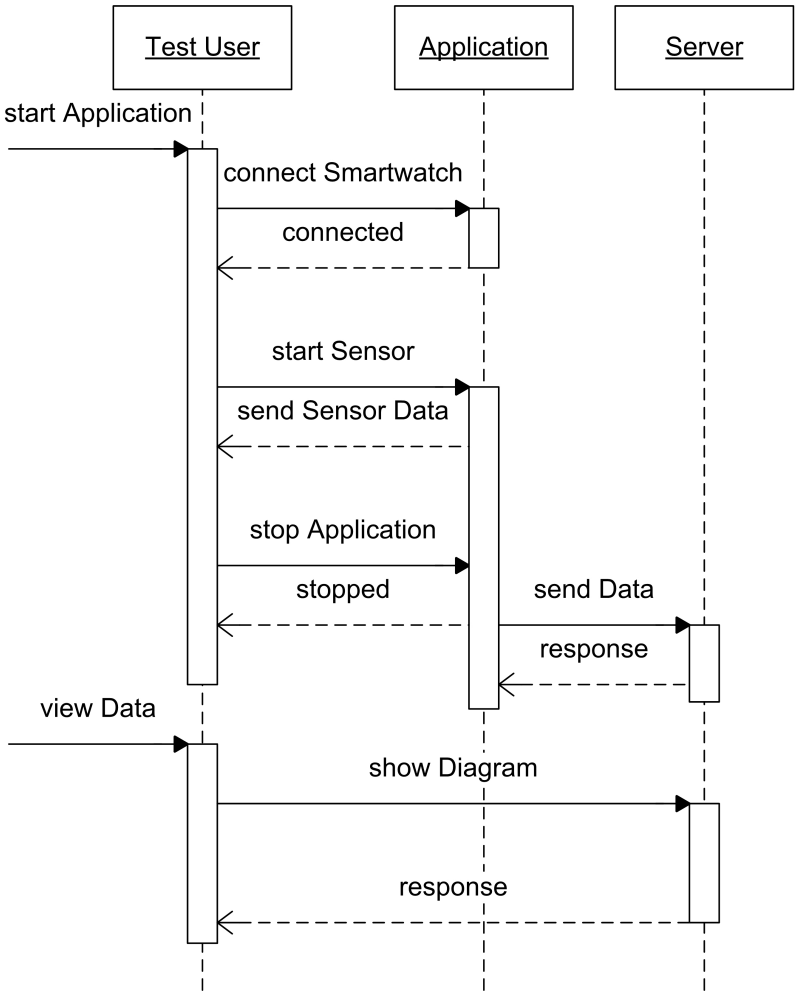

Figure 3.3: Shows the sequence diagram of the test user

application. After the successful connection, the state of the connection is shown for the user. Afterwards the user can start the sensor application and trigger the sensor to send data. The smartwatch then sends sensor data to the smartphone until the application is stopped by the user. Each of the sensor data samples are stored locally on the smartphone until the sensor is triggered to stop. Once the user commands the application to stop the sensor sending data, the data is processed and sent to the private server, where it is stored and further analyzed. The server responds with a message about the transmission, in case of an unsuccessful transmission the server responds with the corresponding error message. The second process the user can start is to request the data from the server and view a chart about the users data stored on the servers database.

The figure 3.4 shows the sequence of a task for the administrator as an end user. After the administrator starts the application for accessing the server, he can choose between three major operations. At first he can simply display the data of all the test users in the database on a web page where it is shown as a table. The second operation the

administrator can do, is to edit the data from the users. It is possible to edit, add or delete users or change specific database entries. A third task the administrator can handle is to show a chart or diagram of the data from the test users. With a chart the administrator can have an overview of the last added data the user uploaded with during a test users task.

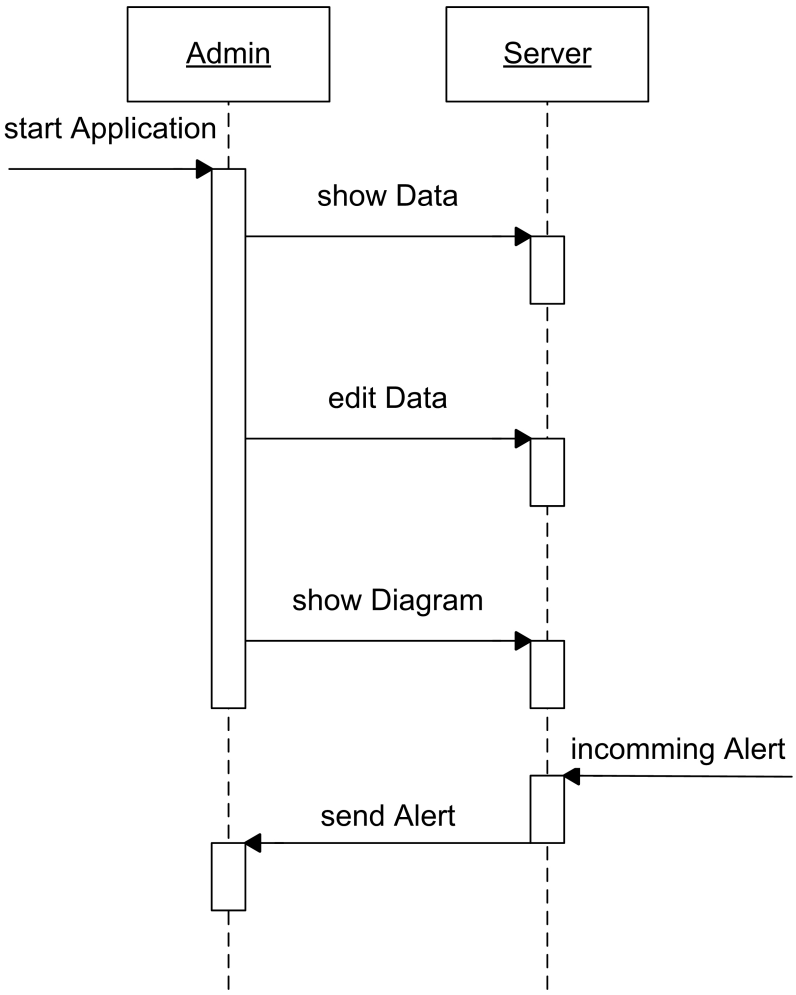

Figure 3.4: Shows the sequence diagram of the admin user

3.2.2 Development View

In the development view, the components and the interaction between the modules are shown, which can be seen in the figure 3.5. There it is presented how the different components from each application interact. The component diagram displays the structure of the systems regarding the components, interfaces and connections. As it can be seen in the figure 3.5, the smartphone application can be presented as the main central application, which connects all other parts of the system.

The smartphone application implements an interface to the smartwatch application, where it handles the sensor data, the smartwatch sends to the mobile phone. The second important component of the smartphone application consists of the activity monitoring, which manages and processes the incoming sensor data from the smartwatch. It is composed of the database component, where the training data and test data are stored in its raw format, as well as in the processed state. Another component of the activity monitoring is the mathematical tools component, which includes the algorithms and data structures necessary to process the sensor data. The detection component of the activity monitoring part of the smartphone application treats the assignment of the sensor data test samples to the training data samples and therefore detects the current activity. The connection between the smartphone application and the server provides the information exchange between the two parties, where the application for the mobile phone sends data to the server and the server responds with a transmission status report.

The server application consists of three major parts, the database management, the *XMPP* service and a web service component. In the database management component, the processed sensor data, which is transmitted from the smartphone, is stored and managed for the various test users. The data is evaluated and in case of a certain event it triggers an *XMPP* service message via the server application. The *XMPP* service component implements the server part of the communication service, which is capable of sending a message via the short message service to the third party smartphone when a certain event is triggered from the server application. The third part of the server is the chart web service component, which implements the representation of the data.

3.2.3 Process View

For the process view, the overall flow control of the main process of the system is shown. In the figure 3.6 it is presented how the control flow of the project works. At first it is necessary to connect the smartphone to the smartwatch, which works with Bluetooth pairing. After the connection is successfully established, the accelerometer on the smartwatch can be enabled and the preferred accelerometer mode can be started, which is performed by pressing buttons on the smartwatch. At this time the smartwatch sends accelerometer data to the smartphone continuously and the smartphone collects the data. Stopping the accelerometer after finishing the desired training triggers the processing and evaluation of the received data on the smartphone. After the data is processed, it is sent to the server, where it is stored on a database. The data is evaluated on the server and sends an alert to a third person smartphone in the case of fulfilling the desired conditions. Beside that, it is possible to access the data from a web browser, show it in a table were it is possible to manage the entries and present it in a chart. After the process it is possible to start again, with a new recording of sensor data.

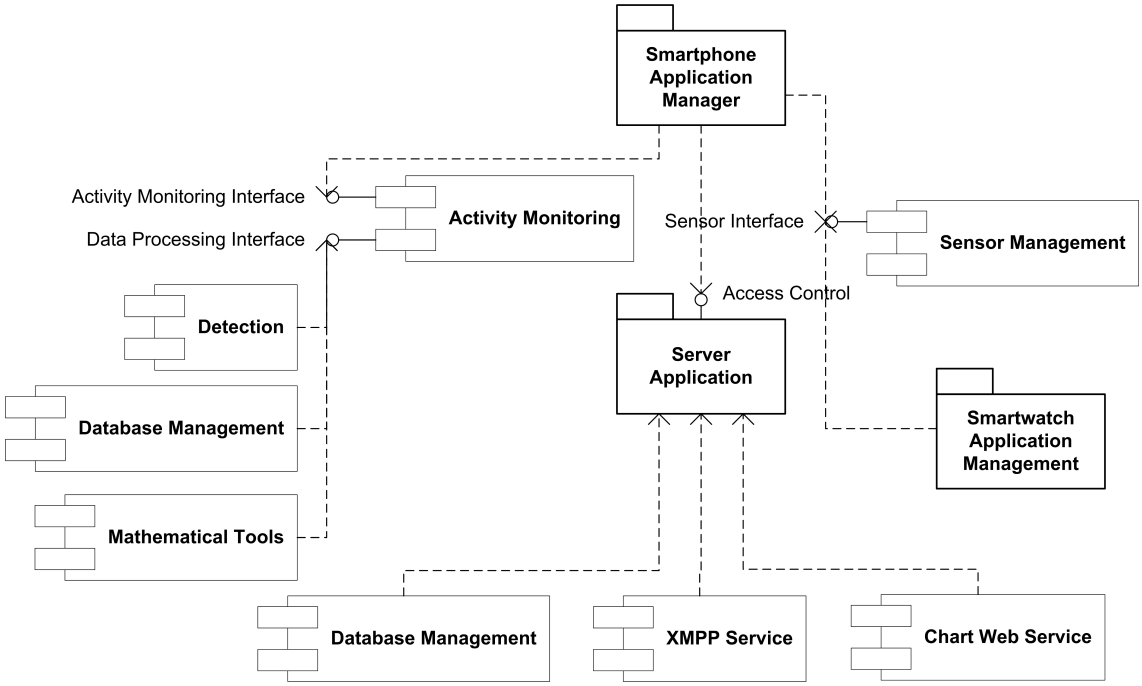

Figure 3.5: Shows the component diagram of the system

3.2.4 Physical View

With the physical view, the interaction between the software and the hardware is explained as well as the communication meanings of the different components. The mapping of the software components and the hardware parts shows the topology of the system for engineers. As it can be seen in the figure 3.7, there are four hardware components involved in the system. The two smartphones need to be Android smartphones, which are capable of operating the implemented application. The first smartphone runs the smartwatch manager application whereas the second smartphone implements the *XMPP* service to be able to receive messages from the server. The first smartphone is connected to the smartwatch with a Bluetooth communication, where the data is exchanged via the *MetaWatch* remote protocol. With a free RTOS operating system, the smartwatch applies an application to send sensor data to the smartphone. The server implements an SQLite database for storing the data sent from the smartphone, an Apache HTTP server for the communication and the XMPP service to communicate to the second Android smartphone. The communication with the first smartphone works with HTTP requests.

3.2.5 Scenarios

The most interesting scenario, which is implemented in this work, is the use case of a fitness application in terms of a step counter and the monitoring of the users in terms of a trainer or administrator. The details of the procedure look as follows, the test user is an owner of a smartwatch and a smartphone. With the application provided to the

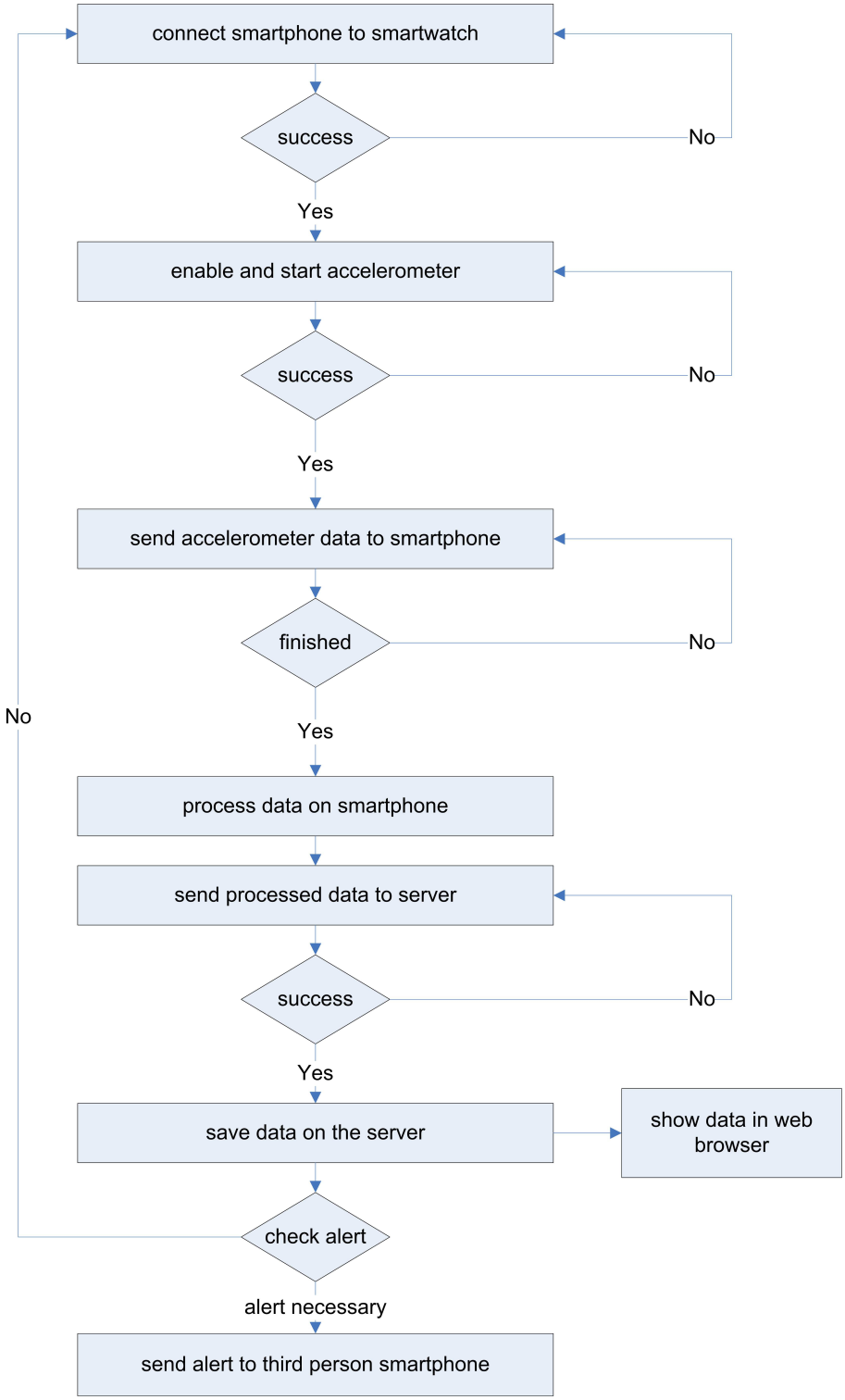

Figure 3.6: Shows the control flow of the project

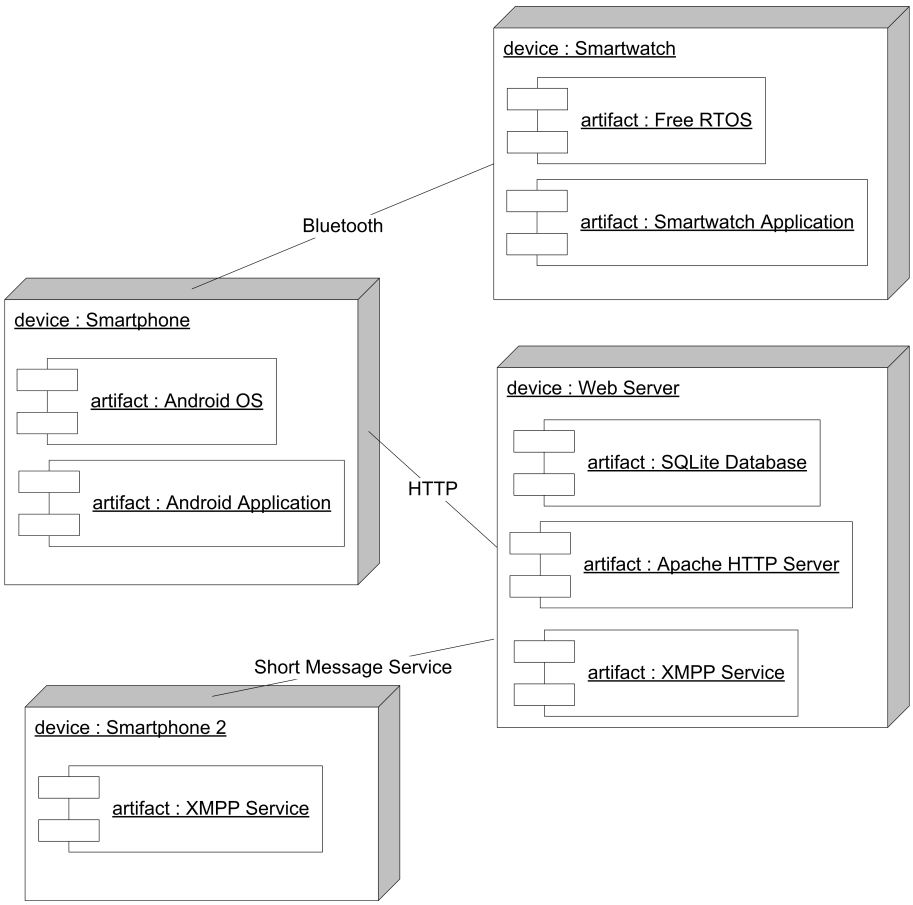

Figure 3.7: Shows the physical view of the system

user and the registration to the system from the administrator, the user is ready to start the application. The use case is designed for fitness situation, where the user starts the smartwatches accelerometer for a workout. If he finishes the workout, the sensor data is processed on the smartphone and then sent to the server, in this special case it is the transmission of the taken steps. The steps are saved on the database of the server and can be displayed on a web browser as a chart. The chart shows the progress and in the case of a reached goal a message is sent to the administrator. The administrator is also capable of monitoring the data from the users and edit the database entries. The figure 3.8 displays the interaction of the user and the administrator with the system. Another use case could be to train the activity recognition part to detect an occurred fall of the test person. In this scenario the administrator performs as a monitoring user, which receives an alarm, when a fall has been detected, but this feature has not been implemented in this work.

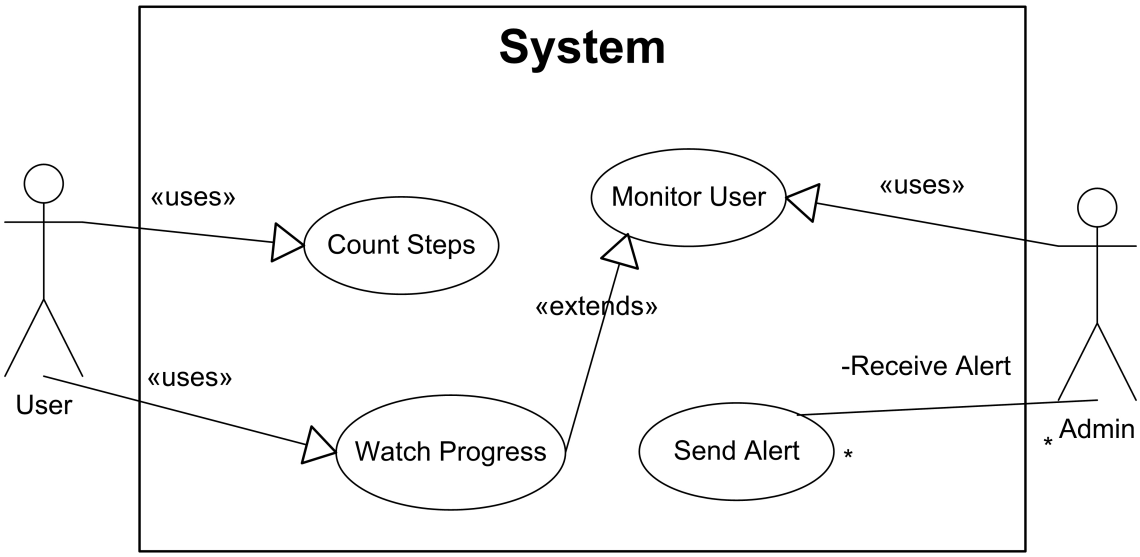

Figure 3.8: Shows the use case of a step counter including a user and a monitoring trainer

3.3 Smartwatch Application Design

The design for the smartwatch application is kept simple and because of the operation with buttons the requirement is that it should be easy to use and not be too complicated. As the smartwatch has all its buttons occupied with other functionality, the middle left button is modified to trigger the application. This works only if the smartphone is connected to the smartwatch, otherwise the button has the original configuration. Pushing the button the application will start, where the screen of the smartwatch will display a notification, which states that the user is now in the accelerometer application. The three right buttons are modified for the use of the application. The screen of the application has space to show text and for each of the right side buttons it displays a symbol, which represents the possible actions. As the configuration of the accelerometer happens remotely via the smartphone application, because of the easier usage with a large touch screen than having several functions for a button, the smartwatch application simply has to start and stop the accelerometer as well as to exit it and show the main screen of the smartwatch.

Another design decision is to let the application run in the background, which means that if the screen mode falls back to idle or another application like telephone call or text message appears, it is still possible to reopen the accelerometer application and stop it or exit again. This prevents the application to interrupt during recording of accelerometer data, when unforeseeable events occur. A drawback of this decision would be that the user may forget about the data streaming in the background and the resource intensive sending of accelerometer data causes battery drain. If the application has been moved in the background, the buttons get the initial configuration back and with the right middle button it is possible to reopen the accelerometer application. The design of the application can be seen in the figure 3.9, it shows the configuration of the buttons and a text in the

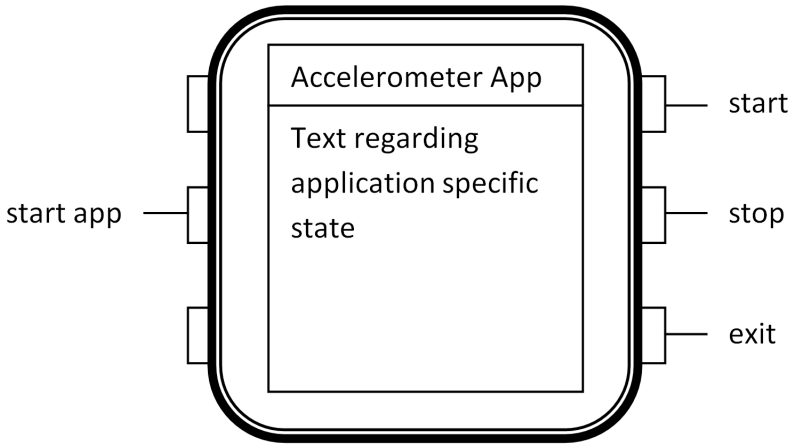

Figure 3.9: Shows the design of the smartwatch application

middle, which can be adjusted regarding the state of the accelerometer.

3.4 Smartphone Application Design

For the graphical design of the smartphone application the design was predefined, because the basis of the application implemented in this work results from the community edition of the *MetaWatch Manager*[1], the accelerometer application design is derived from the watch manager to ensure a uniform graphical user interface. The graphical user interface of the *MetaWatch Manager* is built up in tabs, where the main tab contains the connection button and status to the smartwatch. For the accelerometer application, a new tab is implemented with the following requirements:

- Show the status of the smartwatches accelerometer
- Remote setting possibilities about the accelerometer of the smartwatch
- Display the step counter of the last training
- Inform about the connection to the private server
- Give status information about the sending of data to the server and show messages if any error has occurred

The status of the accelerometer shows, if the accelerometer is started and sends data to the smartphone or stopped. With the remote protocol for the *MetaWatch* smartwatch it is possible to send commands and trigger actions on the smartwatch. For the accelerometer it is possible to set the mode of the accelerometer. With a switch button it is possible to vary between streaming mode and motion detection mode. A message box is used to

[1]https://github.com/benjymous/MWM-for-Android

enter the threshold for the accelerometer in motion detection mode, therefor a numerical keyboard pops up when the message box is selected. For the status messages about the connectivity of the server, the status of the accelerometer and the transmission of the data to the server a space is reserved below the settings.

The smartphone has two connections to other devices, one connection to the smartwatch and the other connection to the private cloud service. To ensure that the connection of both sides are maintained, the application provides for each a status notification, which helps the user to maintain an overview.

3.5 Data Processing Design

The data from sensors can provide a huge amount of information, but it is the responsibility of the developers to acquire the information and the knowledge out of the data. Depending on the use case and the behavior regarding the sensor, a design is necessary to handle the data and gain knowledge out of it. Receiving data continuously through streaming the amount of data is huge and it needs to be processed to gain the information needed for the desired purpose. The single raw signal data of the sensor does not provide much information, but the processing of multiple signal data enables the user to interpret it and gain the information needed for the required project. To be able to process the data, at first the raw signal has to be suitably read from the accelerometer for the required purpose. The second step would be to declare the features, which describe the properties of the data and save the information accordingly to a database for further usage. With these features it is further possible to detect an activity, therefore if the features are not very meaningful, the activity detection will not succeed. After filtering the features it is necessary to classify the information and make further decisions based on the gained knowledge. For the classification, it needs to be determined, which is the best classification technique for the present data and the desired application. Depending on the use cases of the application, the classification techniques show different results. The following subsections show the design for the data processing in the case of the implemented accelerometer sensor processing.

3.5.1 Read Raw Signal

During the reading of a raw signal from a sensor, different properties of the signal need to be considered and taken into account. For reading a streaming accelerometer signal it is necessary to consider the sampling rate and the sampling window, which can than be further processed for the activity recognition. Regarding the sampling rate, if a too high sampling rate is chosen, it can happen that the system is inefficient. In this case it takes too many samples, which are not useful to reproduce the signal and therefore have a clear overhead of data. If the sampling rate is chosen too low, it can happen that the signal can not be reproduced and it can lead to a wrong interpretation. Therefore the *Nyquist-Shannon sampling theorem* should be considered, which states that the sampling

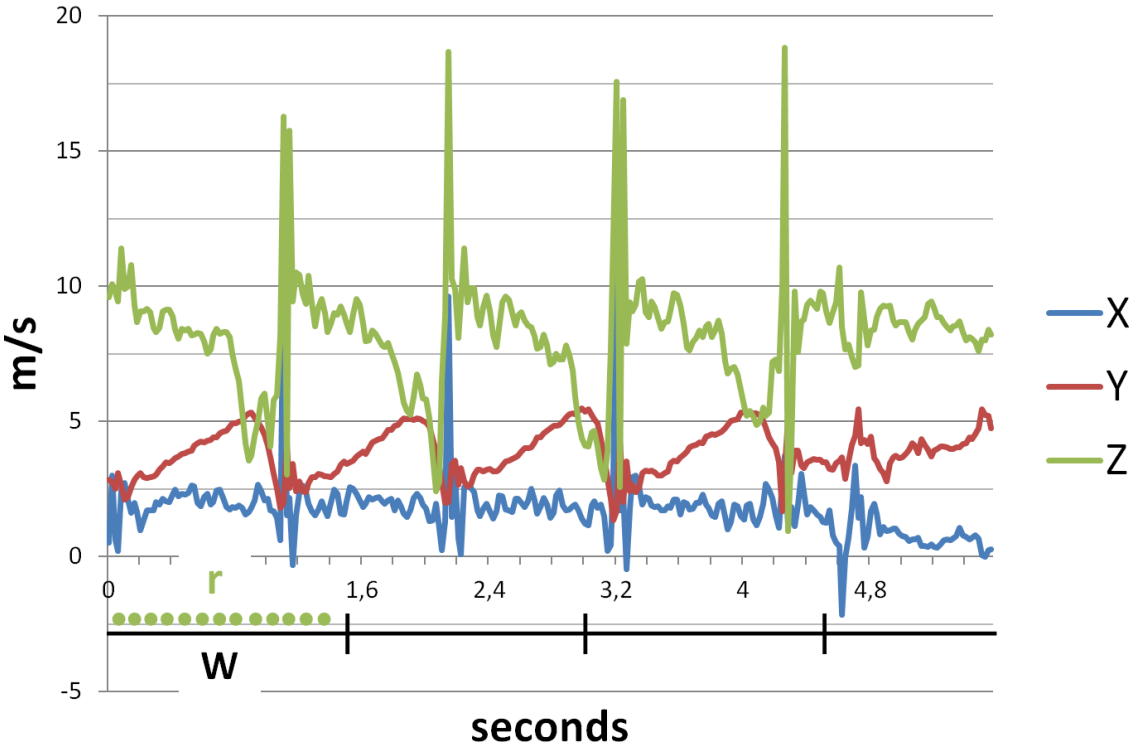

Figure 3.10: Shows the sample window and the sample rate of the acceleration signal

rate should be at least two times the highest frequency of the signal. First experiments have showed, that the sampling rate of 100 samples per second would fit the needs of this project. For the sampling window it is necessary to have a window, which cover a potential periodicity in the signal. If a signal has a certain pattern, like a recurring peak in a constant velocity, which can occur during walking, the sampling window should not be taken too short, otherwise for different samples the peak is not visible and for other samples maybe two peaks are present. This would falsify the results and a wrong interpretation of the sample would occur and therefore the activity recognition would suffer in its accuracy.

For choosing a too large sampling window, again the sampling would be too inefficient. A first suggestion of the sampling window regarding some examples resulted in 1.5 seconds, were also the progression of a slow step is contained. This is also shown in the figure 3.10, where the acceleration values for the different axes and examples of the sample window w and the sample rate r are visualized, where the green dots represent the sample rate and the black line with the sections represents the sample window. The sample rate and the sample window is only shown for explanation of the concept, it does not represent the real value. In this the figure 3.10 also the periodicity of a signal is shown, which shows that a too small sampling window could lead to false results, if the window would cover the same length as the periodicity. Then it could be that the samples could result in a wrong activity. Another point, which needs to be considered regarding reading a raw signal from a sensor is the power consumption. It needs to be declared if it is necessary for the system

to continuously read the signal or if triggering the signal in case of a special purpose fits the needs. Streaming the accelerometer data is extremely costly for the battery and can lead to a short operating time of the smartwatch and therefore the whole system. Triggering the sensor only in a certain case depending on the sensitivity of the accelerometer can help saving battery.

3.5.2 Obtain Features

After the successful transmission of the raw signal data it is necessary to obtain the required features of the signal. Many features can be calculated out of a signal and it is necessary to identify the most interesting features, which are useful for the desired operations. The most interesting features, that should be extracted out of the signal for each of the axis, regarding activity recognition and which are used in this work can be defined as follows:

- Mean and energy of the acceleration
- Variance and furthermore the standard deviation
- Maximum range of the signal
- Interval between the occurring peaks

Other interesting features can be the *FFT*, short for *Fast Fourier Transformation* or the correlation, which help for example to distinguish between the activity walking and to climb a stair, because of the dimensions of translation. These features are necessary to classify the samples and assign them accordingly to the correct activity. Each of the enumerated features cover a different related field of identifying the required activity, depending on the used classification method. The mean and the energy of the signal are very useful, when it is necessary to compare between an intense activity like running and a more slow activity like standing or sitting. Although with the energy of the signal it is hardly possible to distinguish between two similar intense or slow activities, the feature is able to determine the periodicity of an activity.

The mean value can also help to determine different positions, for example when a person is standing or sleeping on a bed. The variance and furthermore the standard deviation are appropriate means, which help to differentiate between running, jumping or walking. With the range of the acceleration it is possible to get a normalized signal, which can be compared and provide a good basis for classification. The range from the lowest peak of the signal sample to the highest peak of the signal sample can easily distinguish between activities like jumping, where high peaks are reached and walking, where usually the peaks are much lower. Another measure, which helps to determine between activities is the interval between peaks. This measure helps for example, when it has to be determined between two similar activities like walking and running, which can have a rather similar signal amplitude, but running has a much smaller interval than walking.

After the successful extraction of the signal features it is necessary to store the information into a database, where the data can be accessed later on for further purposes like classification. For the activity recognition it is necessary to know the label of the trained activity. Therefore beside the data of the extracted features, also the label has to be stored in the database. This step has to be done, that the classification can take place. Therefore the activities have to be tested separately, after the training the features and the label of the trained activity are stored on the database.

3.5.3 Classification

To classify an activity it is necessary to have the training data already prepared in a database, like it is mentioned in the previous section. If the signals of the training samples are plotted, there can be seen some similarities in the signal path. Comparing the features of the different samples can result in some overlapping and depending on the classification method it can lead to false positives, where a wrong activity is chosen. Therefore the classification method has to be chosen depending on the data and the required information, which is gained out of it. A simple classification method, which has also a good performance is the *K-nearest neighbor* classification. The *K-nearest neighbor* is a supervised learning method, which is concluded of training and test samples with many different input vectors, in our case the features of the signal, and only one output vector, which is the activity label mapped to the sample. Advantages of the *KNN* algorithm are that the number of the parameters used can vary and that the algorithm has a good effectiveness depending on the simplicity of the method. This process to determine the class of an object provides a simple assignment, where the distance of the object is measured and the object is classified regarding the k-nearest neighbors. Therefore it has to be decided, which distance metric should be used to determine the distance between the objects. The most common metric is the Euclidean, which is also used in this work. Other possible metrics would the Manhattan distance, the Hamming distance or the Mahalanobis distance. The advantage of the Mahalanobis distance would be, that it is a unit less distance measure and therefore it can compare various objects with different properties. This can also help regarding the classification of objects, when various features are compared with a different weighting. Some preliminary tests have shown that the majority voting and a k value between 3 and 5 show good results in classification of the test samples.

3.6 Cloud Design

A well designed model of the cloud service is necessary to provide a persistent infrastructure for a durable system. The private cloud used in this work consists of a server, which includes a general service, a database and a service to link an Android smartphone. The design of the server is shown in the figure 3.11.

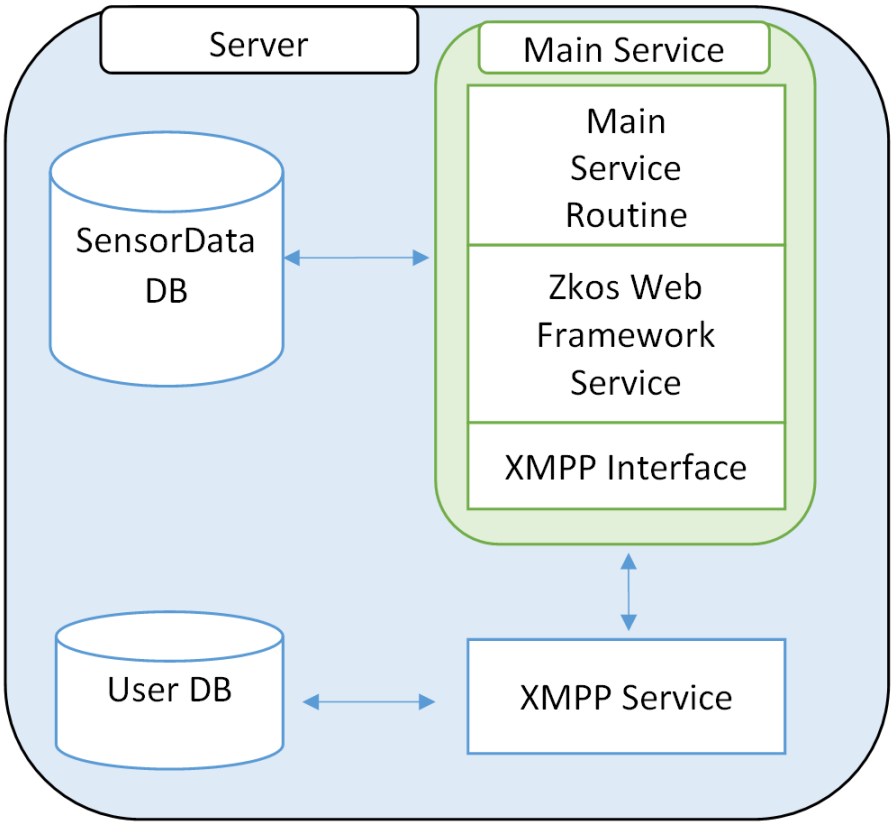

Figure 3.11: Shows the design of server in a more detailed overview

3.6.1 General Service

The general service of the server is the main entry point for the incoming data from the Android smartphone. It has to correctly receive the data and process it, that it can be saved on the database storage. Therefore an interface is needed, which provides a connection to the smartphone and receives the data over the *TCP port* of the server. There it needs to be distinguished, whether the received data is a new entry for the step counter and the data is correct. The data is received as an *JSON* object, which is parsed by the service routine. It also works as an interface to the database and the front end application, which supports a framework to present the data as a table or a chart with a web browser. The requirements for this service can be defined as follows, it has to be stable to withstand occurring problems and should provide a certain degree of flexibility to ensure a durable system, which means that it should not collapse when an error occurs and if a new service is needed the additional work should be simple. Another requirement for the service is that it has a good usability and that it is easy to use.

3.6.2 Database

The database works as a storage for the server data and the front end application, which provides the management and visualization of the data as well as the user data also for the *XMPP Service*. The sensor data is also stored there for further processing to be able to react on the properties of the incoming data with a possible alert on the Android smartphone. It is necessary that the database is capable of handling a huge amount of data, because depending on the number of users sending data to the service the receiving data volume can increase accordingly. The consistency of a database is one of the most important issues regarding the storage of data. It is required to keep the data secure also after a crash of the system. The database has to be reliable and the durability of the storage is also a significant criterion to conform the users demands on a secure cloud database storage.

3.6.3 Front End Applications

The front end applications for the private cloud service include the *XMPP* service, which is capable of binding a connection to an Android smartphone and sending a text message in a specific case and the web service, which supports the user with a management possibility and an overview of the data stored in the database. The alert for the smartphone is depending on the data, which is stored in the database. This data has to be processed and in a certain predefined case a message has to be sent. If the threshold is exceeded than an alert is sent to the smartphone. Therefore the connection to the smartphone has to be reliable and secure to not provide a platform for an attack. The design of the *XMPP* service implemented on the Android device and the connection to the server is shown in the figure 3.12. The service *xmppd* runs in the background of the android application and maintains a constant connection to the server. The database is used for storing intermediate data and works as a backup, if the connection to the server is interrupted or data packets are lost, the data can be sent later on, when the connection is re-established. The status screen shows the information of the connection and informs in case of a problem.

The requirements for the visualization of the data through a web service in the browser should be the usability of the tool, which has to be user friendly, to easily support the user with information to the complex data. Therefore a table view is necessary to provide management options to edit, add and delete certain data and a chart view to show the information as a good overview and summary. To provide a good user friendly design for the web browser front end, the *ZK Framework* (see subsection 4.2.6) is used, which provides table views and different charts that are sufficient for this works requirements. With this framework, a table view and a chart view is designed, were it is possible to select the desired data from the table and then show it in the required chart. For the chart view the line chart is required, which presents the steps in a good graphical overview.

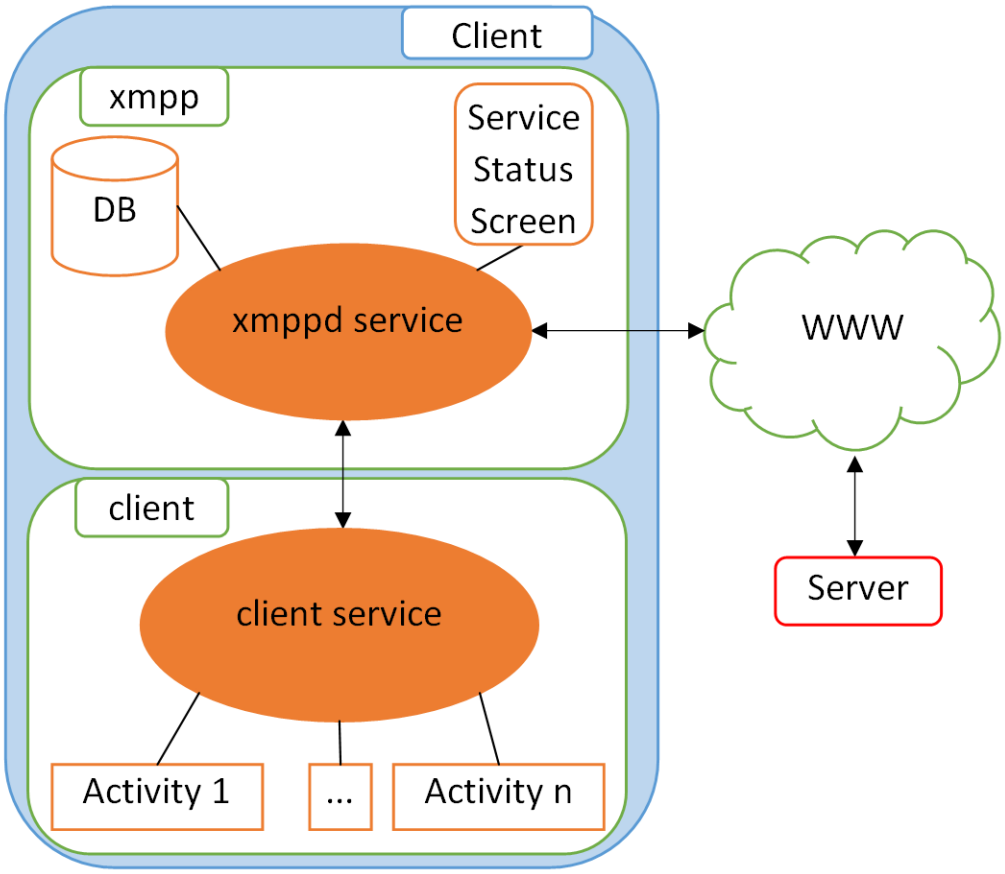

Figure 3.12: Shows the design of the xmpp service implemented on the Android device, which is connected to the server

Chapter 4

Implementation

In this Chapter the implementation of the different parts of the project is explained in details. Starting with the changes made to the watch's real time operating system, following by the Android application implementation with a detailed description of the used algorithms to the cloud service implementation including the server, database, web interface and push notification service.

4.1 Watch Modification

The *MetaWatch* smartwatch[1] started as a *Kickstarter* project with a free available open source firmware, but with the version 1.5.1 it changed to a closed source project. For this work an older modified version 1.4.0 is taken, because it provides an open source Bluetooth protocol stack, were the possibility to modify the Bluetooth stack and guarantee the communication to the smartphone is given. This version is also compatible with the necessary Android smartwatch manager application, which controls the smartwatch and is the basis for the communication to the smartwatch. Another advantage of the open stack version is that the Bluetooth communication between Android and the *MetaWatch* can be debugged. The open stack version is also used, because of the ability to change the source and the way the messages are sent to get a probably less power consuming communication. Regarding the bad documentation of the *MetaWatch* API, the source code gives a good insight on the applications, interfaces and functions of the real time operating system. The behavior of the OS and the communication to the Android platform can be debugged with the developer clip and the *Code Composer Studio*, which helps to understand the implemented remote message protocol and helps for further modification.

The version 1.4.0 contained a few bugs and a problem with the Bluetooth communication between the watch and the used Android version for the smartphone existed. With the current update to the new Android 4.4 image on the smartphone, the Bluetooth protocol could not be detected on the watch side and therefore no messages were sent over the Bluetooth stack. For this problem a fix on the send function of the accelerometer is

[1] http://metawatch.org/

applied in this work, so that the raw accelerometer data can be sent to the smartphone again. Beside this problem, the operating system works and the communication to the Android smart device operates as expected and therefore no further modifications on the watch firmware are necessary for this work.

4.2 Development Tools and Components

For this work different tools are used for the development of the software to run the different components necessary for the system. Beside the tools, also the specific components for the implementation are introduced in this section and how these diverse parts work together is shown in the following subsections. For each of the components different software tools and frameworks are used. The Android software for the smartphone is developed with *Eclipse* integrating the Android SDK, the real time operating system is developed and modified with the *Code Composer Studio* and for the server side *SQLite*, *Apache Tomcat*, *RESTful Web Service* and also the *Eclipse* framework was used. To present the data via a web service, the *ZK Framework* is used, which is capable of presenting data in different considerable views. To create an alert, which is sent to an Android smartphone, the *XMPP Library* is added and the alert management is developed. The process of the toolchain is also shown in the figure 4.1, where the different components and tools are matched.

The figure shows which development tool is used for which component and also displays all the specified components used for this work.

4.2.1 Eclipse IDE

The *Eclipse Integrated Development Environment*[2] offers a platform to develop *JAVA* projects and an easy to use plug-in environment to extend it for example with the *Android Software Development Kit* or the *ZK Framework*, which is used for this work. Within the *Eclipse Marketplace* it is possible to add individual packages necessary for specific development to the environment. The framework also supports other programming languages like *C*, *C++*, *Python* or *Perl*. Within the *JAVA* development package, *Eclipse* offers the development tools including a *JAVA* compiler and a model for highlighting of the source code and documentation of the methods and functions. The software is open source and can be downloaded for free on the *Eclipse Foundation* homepage. Offering different platforms, the software development environment also supports the server, web tools, rich client and modeling platforms. The view of the program can be modified user specific and many different views and perspectives are predefined. To run programmed implementations and inspect the desired behavior of the application, the software kit offers a mode for running and debugging. For this work the version 4.3 is used with the code name *Kepler*. The *Eclipse* Development Kit is also written in *JAVA* and requires at least the version *JAVA 5* to run. The Android application for the smartphone is developed with *Eclipse* as well as the server and the web interface implementation.

[2] http://www.eclipse.org/org/

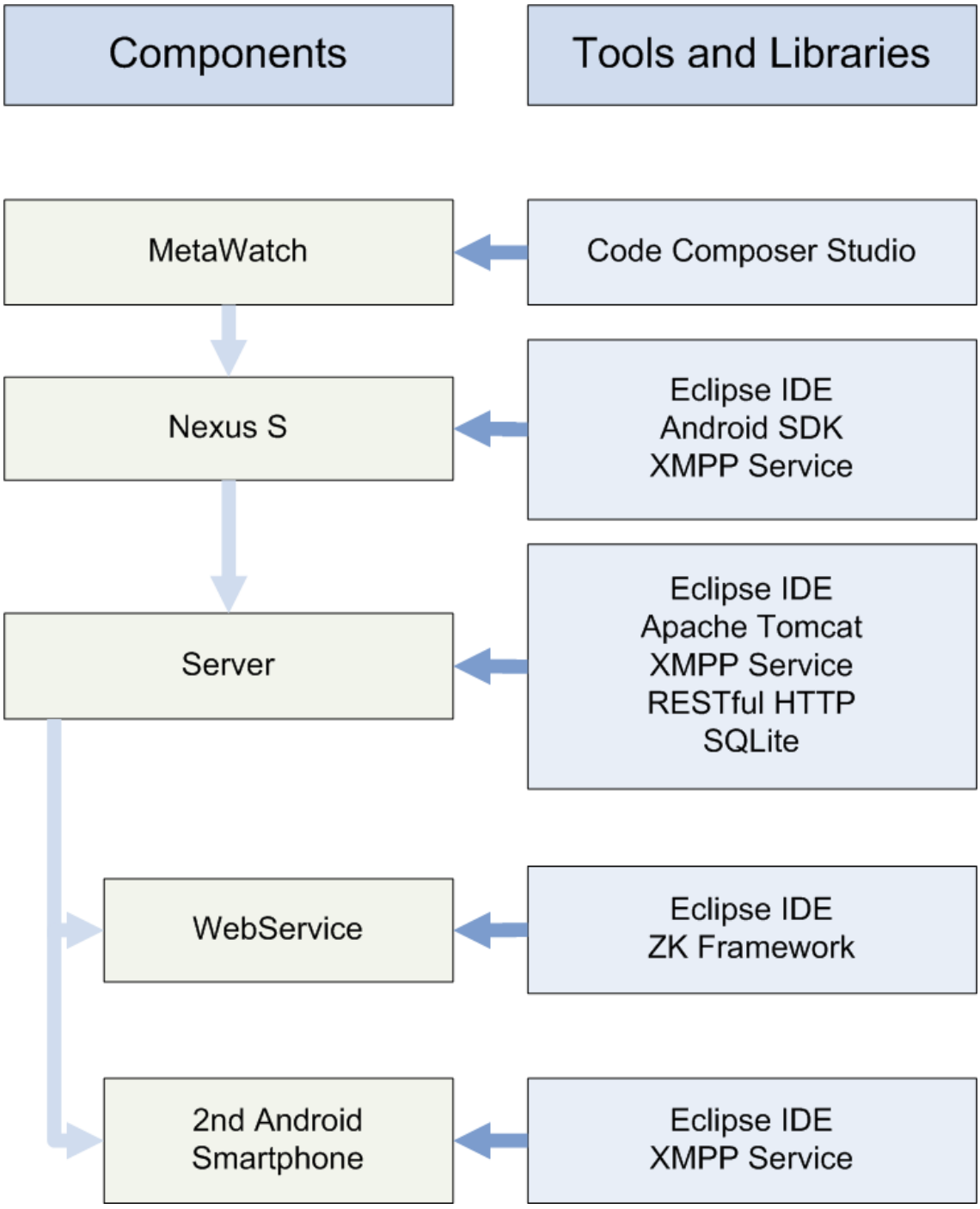

Figure 4.1: Shows the tool chain of the work and the used components

4.2.2 Android SDK

The *Android Software Development Kit*[3] offers all components necessary to implement an *Android* application for a smartphone or other device. It includes tools for development like a virtual device management for debugging, driver support, different libraries, *APIs*, compiler and documentation for the available platforms. To maintain the packages, the development kit is shipped with the *Android SDK Manager*, where it is possible to install desired packages easily. With the *Android* development kit it is possible to debug the application on an emulator or install and debug it on a real device. With the included tool *LogCat* it is also possible to monitor the system output or user specific debug logs. Within the debug view of the *Eclipse* environment it is possible to observer the running processes and thread hierarchy. For the *Eclipse* environment an *Android SDK* plugin exists, which can be downloaded from the *Android* developer homepage. For this work the *Android 4.4.2* platform is used, as well as the *Android Support Library* and the *Android SDK Platform-tools* packages. With this platform the application for the *Nexus S* smartphone is developed.

4.2.3 Code Composer Studio

The *Code Composer Studio*[4] - short CCS - offers a platform to develop programs for processors and micro controller from the manufacturer *Texas Instruments*. With the integrated *C* compiler and source code editor it is possible to implement and compile new applications or modify existing projects. The environment combines the familiar *Eclipse* environment with the possibility of debugging embedded applications for *Texas Instruments* micro controller. During this work, the developer environment is used to enhance the real time operation system for the smartwatch *MetaWatch*, which includes a *Texas Instruments MSP430* micro controller. The *Code Composer Studio* requires an account and a license from *Texas Instruments*, which is offered from the university.

4.2.4 MetaWatch

In this work the smartwatch *MetaWatch*[5] is used to showcase the capabilities of the implemented step counter for smartwatches. The watch was first released in the year 2011 and began as a crowd funding project on the platform *Kickstarter*. The *MetaWatch* was chosen for this work, because at the beginning of the work it was one of the few smartwatches, which supported *Bluetooth Low Energy* and was at that time a developer friendly project. The figure 4.2 shows the structure of the *MetaWatch* in a block diagram. As it is shown in the figure 4.2, the smartwatch consists of a 16-bit *Texas Instruments MSP430* micro controller for low cost and few power consumption and a *Texas Instruments Bluetooth CC2560* module, which supports the modes *Bluetooth Basic Rate*, *Enhanced*

[3] http://developer.android.com/sdk/index.html
[4] http://www.ti.com/tool/ccstudio
[5] http://meta.watch/

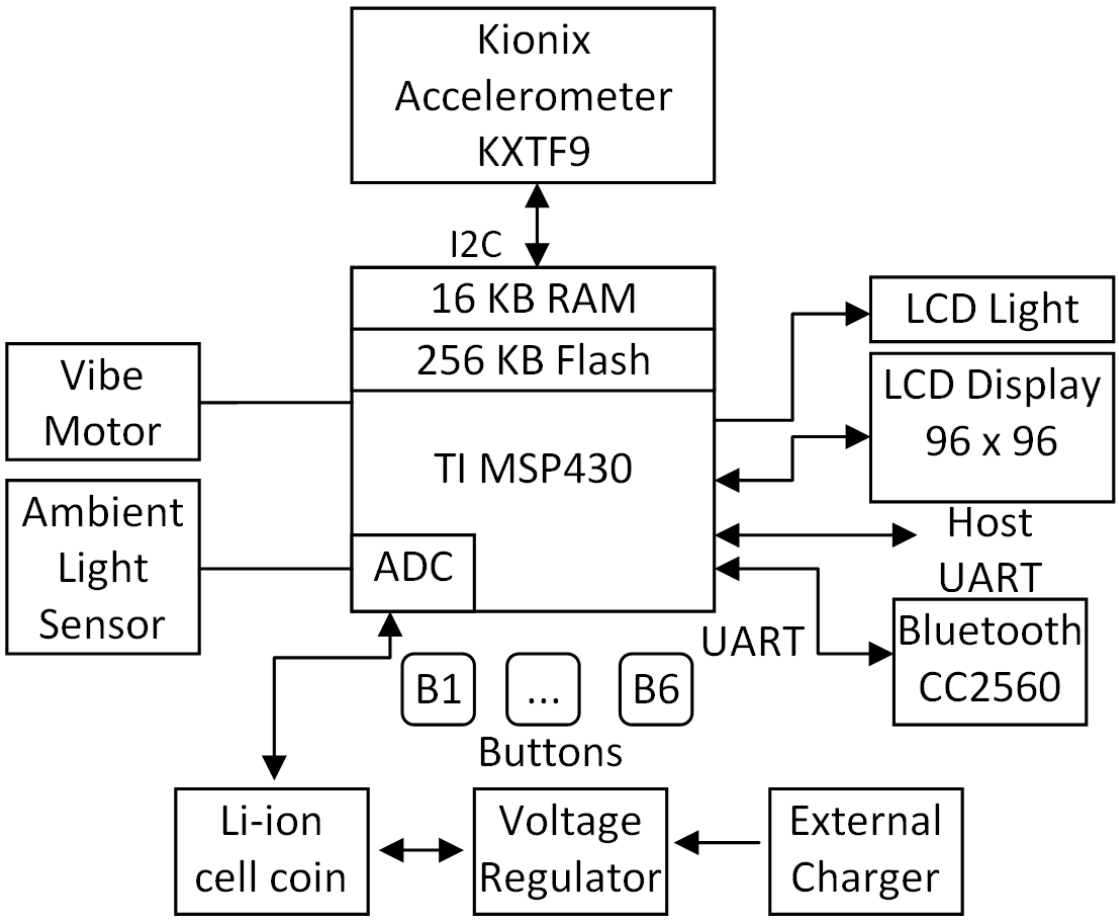

Figure 4.2: Shows the digital watch block diagram of the *MetaWatch* smartwatch [Met11]

Data Rate as well as the new Bluetooth standard *Bluetooth Low Energy*. The accelerometer is produced from the company Kionix and is a tri-axis silicon micro machined accelerometer [Kio11]. The Kionix accelerometer can be operated between a range of +\-2g to +\-8g, which is free select able for the user. The smartwatch operates on a free *RTOS*, which is short for real time operation system. With 16 KB RAM and 256 KB flash storage and a 96x96 pixel screen, the *MetaWatch* is composed of average values for a smartwatch at the release time. It includes 6 buttons to navigate and therefore does not have a touchscreen. The software for the smartwatch was implemented with the *Code Composer Studio* and the project is available under the *MetaWatch License* on the developer homepage.

4.2.5 Samsung Nexus S

To demonstrate the functionality of the implementation created for this work, an *Android Nexus S* smartphone was used. The smartphone is rooted and runs with the latest available stable software version from the project *CyanogenMod* based on the Android version 4.4.4. Running in developer mode, the smartphone has the ability to install and debug

applications within the *Eclipse* environment. The smartphone was released in the year 2010 and the specifications are a Samsung Exynos processor with 1 GHz, 200 MHz GPU, 512 MB RAM and 16 GB of flash storage. An LCD display with 800x480 pixel and an NFC as well as a Bluetooth module complete the picture of this device. The application for managing the smartwatch is installed on this device and via Bluetooth the smartphone is connected to the *MetaWatch*. For sending the data to the server and receiving alerts, the *XMPP service* running in the background is responsible.

4.2.6 ZK Framework

To present the data via access from the internet, the *ZK Framework*[6] is used. This framework allows the user to visualize graphics and charts from data stored in a database via a web browser. It is a framework for web applications and helps the user to build a graphical user interface supported by the *Ajax* based open source. With the sub feature *ZK Charts* it is possible to show the data in different chart variants. *ZK Framework* allows to implement in object oriented *JAVA* and uses for designing interfaces the markup languages *XUL* and *XHTML*. For updating information and the view of a web page, *ZK* uses *JSON* object description, which is more efficient than *HTTP* requests. The framework supports a lot of features like server plus client fusion, UI declaration in ZUML, UI implementation in JAVA, transparent AJAX and JSON, a model, view and controller model or data binding. For this work, the *ZK Framework* is used to present the data as a chart in the web browser and therefore visualize the data from the smartwatch sensors like the accelerometer.

4.2.7 SQLite

For the server database management, where the data is saved, which is sent from the smartwatch via the smartphone, an *SQLite*[7] solution has been chosen. This relational database system library is an free to use *SQL* database engine. *SQLite* operates to a disk file and therefore does not need a separate server process. An advantage of this library is the compactness of the overall size although most of the *SQL* standard is provided. Most of the popular web browser and the programming language *JAVA*, which is used in this work for the server implementation is also supported.

4.2.8 Apache Tomcat

Providing the data in a private cloud, *Apache Tomcat*[8] has been used as a web server. This open source project is widely used for *JAVA* implemented web applications and *JAVA Servlets*. It is available under the Apache license and was originally implemented as a reference implementation for *SUN*. The current stable version 8 requires at least *JAVA* 1.7 to work. *Tomcat* consists of a servlet container called *Catalina*, a *JAVA Server Pages*

[6] http://www.zkoss.org/
[7] http://www.sqlite.org/
[8] http://tomcat.apache.org/

engine with the name *Jasper* and an HTTP connector called *Coyote*. It has been chosen for this work, because it can be used under an open source license.

4.2.9 Representational State Transfer

Transferring the data from the smartphone to the web server, for this work *RESTful HTTP* is used. For implementation of a *REST* web service, the protocol rules of HTTP are used. The architecture of the *Representational State Transfer* is defined above resources. A *URI - Unique Resource Identifier* is needed for addressing the users identity, each resource works with methods like *POST*, *GET* or *DELETE* to access the desired data. These methods are used to read, create new resources or delete them. To ensure safe access of the data, the authentication can be realized via HTTP and encrypted with *SSL*. The *RESTful Web Service* is implemented in this work, because it is an easy and safe way to send data to a web server.

4.2.10 Secure Android Communication Service via XMPP

To ensure a secure and easy communication for sensitive data from an Android smartphone to a server, the *XMPP* communication protocol is used within an Android service. The purpose of the implementation of this service is to send an alert to a smartphone depending on certain data evaluated on the server. The reliable protocol between two communication partners *TCP*, the support of the *SSL* technology and the possibilities to send structured data via *XML* make the *XMPP* protocol a good choice to send critical data. Designed for instant messaging services, this open protocol offers a distributed network structure, which is decentralized and therefore an own private XMPP server can be established.

4.3 Android Application Implementation

For the interface between the private mobile cloud and the wearable device an Android application for the smartphone is operating in between. It ensures the communication from the smartwatch to the smartphone and from there to the server and performs different operations on the sensor data, like the raw accelerometer data processing, analysis of the received data and further the activity recognition. Furthermore it displays the current step counter of the last effected walk and offers various settings for the watches accelerometer sensor, like the changing of the accelerometer mode from a streaming mode to a threshold based detection motion mode or the setting of the desired threshold for the motion detection mode. The structure of the application, which is developed for an Android platform smartphone can be divided into five major parts:

- The base application used for the communication from the smartphone to the smartwatch
- The graphical user interface for the accelerometer and the step counter on the smartphone screen

- The interface for accessing and processing the sensor data sent from the wearable device
- The activity recognition part of the application for identifying the counted steps and other activities
- The communication to the private mobile cloud service as well as the secure service to send an alert

This application for the Android OS, which contains the main part of this work, is based on the developers edition of the MetaWatch Manager - called *MetaWatch Community Edition* [Ric], which implements communication establishment from the smartphone to the smartwatch and management options. This community edition was formed from the official sources of *MetaWatch Manager* reference implementation. The MetaWatch Manager ensures the connection from the Android device to the MetaWatch smartwatch. It contains various setting options, test cases to prove the correct functionality of the features and widgets like the clock display, receiving of messages and the telephone function, which can be placed on the smartwatch screen. The application is built up with several tabs for each section. In the status tab, it shows if the smartwatch is connected and gives information about upcoming message and notification queues, which are currently processed by the smartwatch. In the preferences section, settings for notifications, calls widgets and other applications can be made. The widget tab enables the user to select the main widgets, which should be displayed on the smartwatch screen and in the test tab it is possible to test some main functions like sending a message or showing calendar notifications. This base application also includes protocols and a service routine to communicate to the *MetaWatch*, that allows to send and receive data to and from the smartwatch. Integrated to this application an activity recognition for counting the steps and detecting other activities is applied. An interface for the processing of the data sent from the wearable device is developed as well as the communication to the private mobile cloud, which is included in this project.

4.3.1 Graphical User Interface for the Accelerometer

To be able to change the behavior of the smartwatches accelerometer a graphical user interface is developed. Because the smartwatch does not provide a touchscreen and it is only possible to interact with the watches buttons, it is simpler to operate some features from the smartphone device with an easy to use interface. As the *MetaWatch* provides a remote message protocol, it is possible to trigger some features remotely from the smartphone. A screenshot of the accelerometer GUI can be seen in the figure 4.3. It shows tab of the accelerometer manager and step counter, which is embedded in the original MetaWatch manager application. On the top of the screenshot it shows a switch button, which can be changed from a threshold based motion detection mode to a streaming mode. The threshold mode of the accelerometer implements the functionality, that the smartwatch

only sends data to the smartphone at a specific threshold. This threshold is per default selected with the value 16, which is converted to a 0.5g with a +/- 8g range value and can be changed in the text box below the switch button. In the text box the values between 0 and 255 can be entered, which would result of a threshold at a maximum of nearly 8g for the accelerometer.

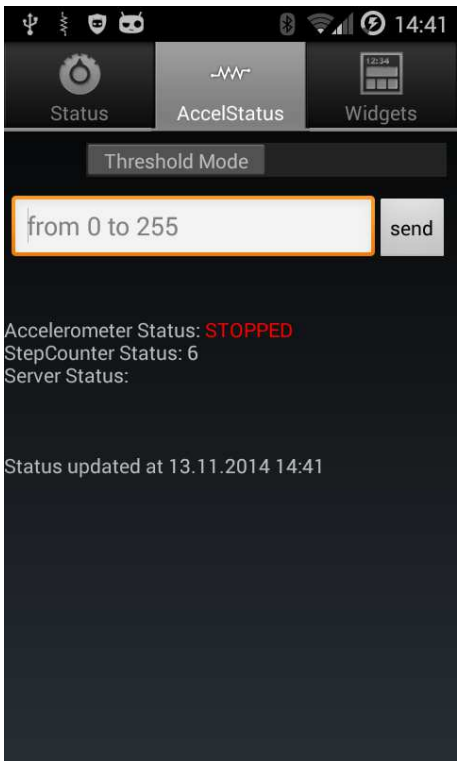

Figure 4.3: Shows the graphical user interface of the smartphone for handling the accelerometer of the smartwatch

The application running on the smartwatch for the accelerometer handling can be started with the middle button on the left side of the watch. Running the application, acceleration data is transmitted to the smartphone. The accelerometer can be started and stopped with the right-sided buttons on the smartwatch and the program can be closed with the lower button also on the right side. The GUI of the Android application on the smartphone indicates, if the accelerometer is started and data is transmitted or if it is stopped. The following notification underneath shows the steps, which were counted the last time the accelerometer of the smartwatch was activated, as it can also be seen in the figure 4.3. Beneath the notification of the steps, the status of the server is displayed. It indicates, if the last message, which was sent to the server, has been successfully transmitted or in case of a failure it shows the occurred error message. The bottom of the status screen for the watches accelerometer sensor shows the last modification date of the status, which is updated if the tab is changed or if the accelerometer starts or stops.

4.3.2 Smartphone to Smartwatch Communication

The *MetaWatch* wearable device was one of the first smartwatches, which uses an enabled Bluetooth 4.0 stack for communication. With the Bluetooth 4.0 specifications - also called *BLE*, which stands for *Bluetooth Low Energy* - new protocols for a reductive power consumption were defined [SIG10]. With a faster connection establishment and other optimization for a low energy transmission, the specification focuses on low power devices and the current version is therefore a good choice for the connection of wearable devices. Because *BLE* is not completely downward compatible, the MetaWatch smartwatch provides a dual Bluetooth implementation, which supports the classic Bluetooth connection besides the low energy specification. This dual mode allows to connect older smartphones over *Bluetooth SPP*, which do not support the 4.0 specification and therefore reach a high coverage of compatible connectable smartphones.

To communicate with the MetaWatch, a remote message protocol was designed, which allows to send bi-directional messages from the smartphone. With this protocol it is possible to have remotely control and change functionality of the smartwatch from a connected device. The format of the packets with a maximum length of 32 bytes, which are sent and received, are defined as showed in the following table 4.1:

Byte	*B0*	*B1*	*B2*	*B3*	*B4...n-3*	*Bn-2,Bn-1*
Field	Start	Length = n	Message Type	Options	Data	CRC
Value	0x01	0x06-0x20	0x00-0xFF	0x00-0xFF	-	-

Table 4.1: Shows the structure of the remote protocol of the MetaWatch

As it shows in the table 4.1 the protocol consists of 3 different parts, the header, the data field and a two byte checksum. The header is composed of four bytes, the start field, the length byte, the message type and the options field. The start field indicates the beginning of a remote message, it has the value 0x01. The length field contains the length of the total command and has to be at least the value of 0x06, which includes the size of the header and the CRC value. The size of the message can be at a maximum 32 bytes long. The message type byte defines the number of the type of message, which is sent. The available message types are defined in the MetaWatch OS. In the options field, different message specific options can be called. 0 - 26 bytes can be sent within the data byte array and the CRC field consists of a 16 byte checksum, calculated over the header and data bytes.

Most of the protocols are defined in the original source of the MetaWatch Manager and can easily be called within the service routine of the MetaWatch, however the implementation for the accelerometer is not implemented. For this work, two methods are defined for the accelerometer to send commands and receive data via the remote protocol. The first method is used for the setup of the accelerometer and has six different options, which can be selected from the sender. It has the ability to enable or disable the accelerometer as well as the modification of the accelerometer settings. The accelerometer sensor behavior

can be modified through sending one of the options for the two acceleration modes, where it can be switched between the streaming and motion detection. Another option is to change the acceleration range, where it is possible to switch between +/- 2g, +/- 4g and +/- 8g. Through the threshold option, the threshold for the motion detection mode can be adjusted. The structure of the message is shown in the table 4.2.

Byte	*B0*	*B1*	*B2*	*B3*	*B4...n-3*	*Bn-2,Bn-1*
Field	Start	n	Msg	Options	Data	CRC
Value	0x01	0x06-0x07	0xE1	0x00-0x05	-	-

Table 4.2: Shows the message for setting up the accelerometer

The second method deals with the transmission of the accelerometer message. When a message with the type 0xE0 is received on the smartphone, the MetaWatch smartwatch has sent accelerometer data. Therefore a method is implemented within the smartphone application, which handles the received data. The message consists of a header including a start byte, the length of the message and the message index and the body, which consists of 3 bytes, where each byte represents one of the axes X, Y and Z of the accelerometer. As the sent values are represented as hex values, the received bytes need to be processed to get the actual float values of the acceleration of the different axes. The receiving message protocol structure is shown in the table 4.3.

Byte	*B0*	*B1*	*B2*	*B3*	*B4*	*B5*	*B6*	*Bn-2,Bn-1*
Field	Start	n	Msg	Options	X	Y	Z	CRC
Value	0x01	0x09	0xE0	0x00	-	-	-	-

Table 4.3: Shows the message for receiving acceleromter data

With these two messages it is possible to setup the accelerometer remotely from the smartphone and transmit acceleration data, which fits the needs for this project requirements. To include these messages into the service process, a special watch mode for the accelerometer is implemented. In the service routine of the Android application, the device checks, if the smartwatch sends a message. This service runs in a service thread in the background and is started, when the smartwatch is connected to the smartphone. It then reads the bytes, which were sent and depending on the message type it decides how the incoming message is handled. The accelerometer application on the smartwatch starts, when the middle left button on the smartwatch is pressed. The wearable device then sends a message to the smartphone and it is checked, during the loop in the service routine, whereupon it triggers a notification to pop up on the smartwatch, which enables the accelerometer mode. In the figure 4.4, the accelerometer application on the watch-side is shown. It is a simple interface, which indicates that the user has started the accelerometer application on the smartwatch. On the right side of the screen the symbols, representing the mechanical buttons, can be seen. The symbols represent play, stop and discard of the application. Pressing these buttons the MetaWatch interacts with the smartphone

application via the message protocol with the messages shown before. Therefore the right buttons on the MetaWatch are redefined to trigger the functionality start and stop of the sensor and the functionality to exit the accelerometer mode. Leaving this mode via the exit button, the original functionality for the buttons is restored and the watch switches back to the idle mode, waiting for instructions from the user.

Figure 4.4: Shows the graphical user interface of the smartwatch for handling the accelerometer

4.3.3 Accelerometer Raw Data Processing

To get useful data from the raw accelerometer signals, which can be further used for analyzing and evaluating, pre-processing of the given values is required. The raw values sent from the smartwatch accelerometer sensor, which is a Kionix KXTF9 sensor [Kio11], are 8 bit digital count values for each axis X, Y and Z. This byte needs to be processed to a float in order to get the real acceleration value, which represents the gravitational value of the axis. The data sheet for the sensor shows how to calculate the G acceleration value of the axes. For the 8-bit data register and a plus/minus 8g range, the calculation is shown in the figure 4.5, where the digital count is translated to acceleration G.

There are two ways to deal with the accelerometer data in case of step counting in this work. The first, easier, but a bit less accurate way, is to trigger the accelerometer only at a certain threshold of the gravitation values. This implementation has the following two advantages:

- Less power consumption
- Less processing effort of the accelerometer data

Regarding the power consumption of the Bluetooth module, it is related to the transmission of the accelerometer data. While in streaming mode, the smartwatch sends data

8-bit Register Data (2's complement)	Equivalent Counts in decimal	Range = +/-2g	Range = +/-4g	Range = +/-8g
0111 1111	127	+1.984g	+3.968g	+7.936g
0111 1110	126	+1.968g	+3.936g	+7.872g
...	...	...	...	...
0000 0001	1	+0.016g	+0.032g	+0.064g
0000 0000	0	0.000g	0.000g	0.000g
1111 1111	-1	-0.016g	-0.032g	-0.064g
...	...	...	...	...
1000 0001	-127	-1.984g	-3.968g	-7.936g
1000 0000	-128	-2.000g	-4.000g	-8.000g

Figure 9. Acceleration (g) Calculation

Figure 4.5: Shows how the 8 bit values are processed into float G acceleration values

constantly to the smartphone, in the threshold based motion detection mode the watch only sends data, when a threshold is exceeded. Sending constantly data to the smartphone is power consumptive and reduces the duration of the battery of the smartwatch and the smartphone as seen in the Chapter 5, where the different modes are compared in runtime. To count the steps regarding motion detection, the threshold for triggering a step needs to be well chosen and adjusted to the various people, who are wearing the watch. The problem with a too high threshold would be, that too less steps or in the worst case no steps at all are counted, because the threshold is rarely reached. With a low value chosen for the threshold, a problem can occur with too many false detected steps, which would lead to an unnatural high number and falsify the results. The accelerometer transmission is then triggered too often. Using this threshold based approach, the overhead of processing the accelerometer data is much less than for the streaming mode. On the one hand, because of the much less data, which is sent and needs to be processed and on the other hand, because the step is already detected with the exceeding of the threshold and it is therefore not necessary to evaluate a whole window of a transmitted period, like in streaming mode. This is also shown in the following Chapter. This rather simple attempt to recognize a step would help to give a general estimation for step counts. If the approach with the motion detection fits the needs for a competitive step counter is also shown in the Chapter 5.

To improve the step counter in the motion detection mode, it is necessary to process the acceleration values after receiving a threshold exceeded value from the smartwatch. The value needs to be analyzed on its probability of being a step. Therefore the range for possible step values need to be defined. Because a too low value is not sent to the smartphone, the high limit of a step needs to be chosen for each of the axis. If this range is exceeded, the probability that the triggered accelerometer value is not a step increases and therefore it needs to be considered, if it can be counted as a step. This range needs to be adapted for each user. Because of the different behavior of people making steps, this

range can vary from person to person. Also the height of a person and the arm length can influence the behavior of making a step. Another measure to detect false positives is to measure the time between two triggered values and conclude the possibility of two or more steps made after a short period. If the period is too small then at least one of the occurring threshold exceedances is a false positive and therefore needs to be discarded. With a usual step speed of 1 m per second, it is not very likely that two steps are made in between a 400 millisecond period. A disadvantage for the threshold based detection mode is that the activity monitoring is not possible and therefore no other activities can be recognized. With the threshold detection, only a single peak at one point is measured and therefore no relevant sequence before and afterwards can be evaluated, which would give information for further activity analyzing.

4.3.4 Training Data

For the activity monitoring the received data, streamed from the smartwatch, needs to be analyzed and compared to other data to conclude, if the data can be seen as a step or an other activity. To be able to compare the data under test with data already identified as steps or a walking move, previous stored training data is needed. For the recording of the training data a specific application is implemented, which can label training data for the state idle, walking and running and store it in a database. The user interface of the application is kept simple, because the only purpose is to collect training data of the specific activities for the main step counter application. The data is stored in an *SQLite* database on the external memory of the smartphone. The table of the database consists of the standard deviation and the range of each of the axis as well as the activity type and the ID of the record. This can be seen in the table 4.4.

field	***value***
id	integer primary key autoincrement
stdX	decimal
stdY	decimal
stdZ	decimal
rangeX	decimal
rangeY	decimal
rangeZ	decimal
type	varchar(45)

Table 4.4: Shows the table structure of the training data

The application for storing the training data is shown in the figure 4.6. There the graphical user interface is displayed and the different states with each button to train the activity is shown. To train a state, the specific button needs to be pressed to start the session, which is then relabeled as a stop button. After starting the training, the activity should be performed as it is made in the real life. During this period, the raw accelerometer data is stored in a list of sets and after a defined sample time the raw values

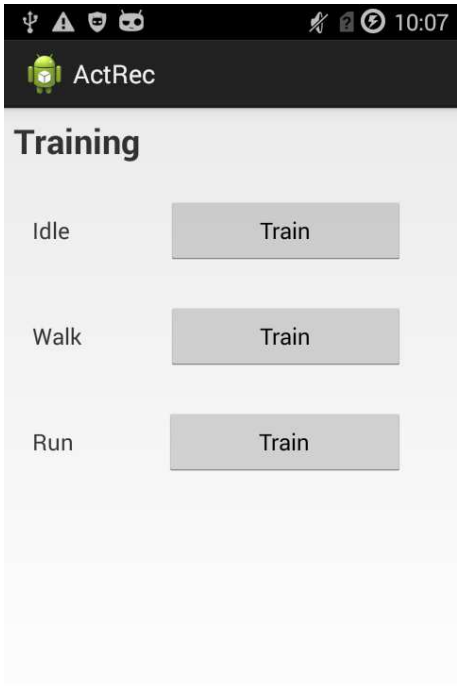

Figure 4.6: Shows the graphical user interface of the smartphone for storing training data

are processed and stored in the database. When the training is finished, the stop button needs to be pressed. The pre-processing of the data and the storage of it in the *SQLite* database is nearly the same procedure then for the test data, except that in the training data process, the assignment to the label is already known and therefore not necessary. The variance, the standard deviation and the ranges of the acceleration axes are stored as well as the activity type, which has been trained. The algorithm and the detailed processing is described in the following subsection 4.3.5.

4.3.5 Algorithm Implementation

The processing of the accelerometer data in the streaming mode works as follows. At first the raw accelerometer data are saved in a list of *RawValues*, after a sampling window of 1500 ms is reached, with the list of *RawValues* a new *ActivitySample* is added to a set of *ActivitySamples*. During the construction of the *ActivitySample*, the values like standard deviation, variance and range, which are necessary for the database table and further processing are calculated. The calculation of the standard deviation works as follows. At first the mean value of each axis needs to be calculated. This happens by summing up each axis values and then divide them with the number of the given values. In the second step, the variances of the mean values are calculated. The variance is the sum of the current axis value minus the mean axis value to the square divided by the number of values. The

equation has been implemented regarding the following mathematical formula 4.1.

$$Var(X) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{4.1}$$

For the standard deviation, the square root for the variance needs to be processed, this happens with a math library included in the project. The range for each axis is calculated through the maximum value subtracted by the minimum value. The equation for the standard deviation is shown here 4.2.

$$s_N(X) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{4.2}$$

After the accelerometer has finished sending data, the training samples are stored in the database with the calculated features. For the samples under test at first all of the training data samples are queried and stored in a Collection of *ActivitySamples*. For these *ActivitySamples* a new *kd-tree* is created. Each of the training samples is added to the k dimensional tree. The adding of the samples as a node in the tree works with an algorithm from the *Handbook of Algorithms and Data Structures* [GBY91]. For the value k, the number of standard deviation and ranges was selected, so that the dimension of the tree equals the value 6.

To calculate the nearest neighbor for each test sample, the algorithm proposed by Andrew Moore [Moo91] is used. The algorithm is shown in the figure 4.7, which explains how to find the nearest neighbor - short NNR - for the wanted sample. For the indicator k of the nearest neighbor, the value 5 was selected as this value showed good results on the testing phase. Using a too small k, the possibility that a wrong neighbor is found is high, which supports noise from the training data and would lead to volatile results. If the value for k was chosen to high, the possibility that samples with a large distance are taken into account, which are not reliable for the test sample. The number of the nearest neighbors selected can show different results for different training data and test samples, however tests showed that a good k value should not exceed the range between the value 3 and 6. The samples are all stored to an array list of test results and sorted regarding their time stamps. After finishing all samples, these samples get assigned to the activities regarding the nearest neighbor algorithm and in the case of the step counter, the steps are incremented for each sample with the NNR found for the walking activity. After completing this procedure, the information is sent via HTTP requests to the server, where it is stored for further evaluation.

4.3.6 Server Communication

After finishing the calculations of the steps and processing of the values, the data is sent to the private server. The data exchange happens via *HTTP* requests like *HTTPGet* and *HTTPPost*. After the communication to the server has successfully established and

Algorithm:	Nearest Neighbour in a *k*d-tree
Input:	**kd**, of type **kdtree** **target**, of type **domain vector** **hr**, of type **hyperrectangle** **max-dist-sqd**, of type **float**
Output:	**nearest**, of type **exemplar** **dist-sqd**, of type **float**
Pre:	*Is-legal-kdtree*(**kd**)
Post:	Informally, the postcondition is that **nearest** is a nearest exemplar to **target** which also lies both within the hyperrectangle **hr** and within distance $\sqrt{\textbf{max-dist-sqd}}$ of **target**. $\sqrt{\textbf{dist-sqd}}$ is the distance of this nearest point. If there is no such point then **dist-sqd** contains infinity.
Code:	
1.	if **kd** is empty then set **dist-sqd** to infinity and exit.
2.	**s** := **split** field of **kd**
3.	**pivot** := **dom-elt** field of **kd**
4.	Cut **hr** into two sub-hyperrectangles **left-hr** and **right-hr**. The cut plane is through **pivot** and perpendicular to the **s** dimension.
5.	**target-in-left** := $\textbf{target}_{\textbf{s}} \leq \textbf{pivot}_{\textbf{s}}$
6.	if **target-in-left** then
6.1	**nearer-kd** := **left** field of **kd** and **nearer-hr** := **left-hr**
6.2	**further-kd** .= **right** field of **kd** and **further-hr** := **right-hr**
7.	if **not target-in-left** then
7.1	**nearer-kd** := **right** field of **kd** and **nearer-hr** := **right-hr**
7.2	**further-kd** := **left** field of **kd** and **further-hr** := **left-hr**
8.	Recursively call **Nearest Neighbour** with parameters (**nearer-kd**,**target**, **nearer-hr**,**max-dist-sqd**), storing the results in **nearest** and **dist-sqd**
9.	**max-dist-sqd** := minimum of **max-dist-sqd** and **dist-sqd**
10.	A nearer point could only lie in **further-kd** if there were some part of **further-hr** within distance $\sqrt{\textbf{max-dist-sqd}}$ of **target**. if this is the case then
10.1	if $(\textbf{pivot} - \textbf{target})^2 < \textbf{dist-sqd}$ then
10.1.1	**nearest** = (**pivot**, **range-elt** field of **kd**)
10.1.2	**dist-sqd** := $(\textbf{pivot} - \textbf{target})^2$
10.1.3	**max-dist-sqd** := **dist-sqd**
10 2	Recursively call **Nearest Neighbour** with parameters (**further-kd**,**target**, **further-hr**,**max-dist-sqd**), storing the results in **temp-nearest** and **temp-dist-sqd**
10.3	If **temp-dist-sqd** < **dist-sqd** then
10.3.1	**nearest** .= **temp-nearest** and **dist-sqd** := **temp-dist-sqd**
Proof:	Outlined in text

Figure 4.7: Shows the algorithm for the nearest neighbor search [Moo91]

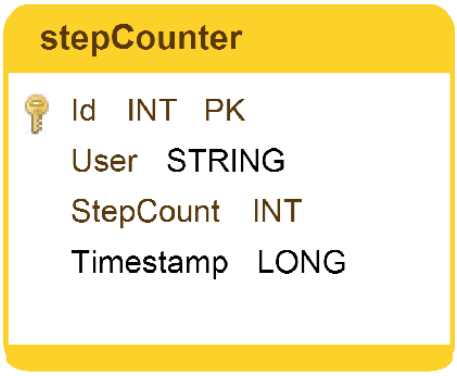

Figure 4.8: Shows the database table for the step counter

the accelerometer has finished recording data, this data is sent via an *HTTPGet* and execute command to the server. These commands include all data necessary for further processing on the server side. After a successful transmission of the data, the server sends a notification. In case of a unsuccessful attempt of sending the desired data to the server, the response contains the related error message. If the transmission of the data fails consecutively, the error message is shown in the Android application and the transmission attempt stops. If the data was sent successfully, a new entry in the database is created, where the specific data is stored for each registered user.

4.4 Server Implementation

The private mobile cloud consists of a server implementation, a database for storing the data on the server side and a service, which is capable of sending an alert to an Android smartphone. The data transmitted from the Android smartphone to the server needs to be managed on the server and therefore a service is implemented which handles the data and stores it to the database as well as it evaluates the data and forwards a message to send an alert.

4.4.1 Database

The database used on the server is a *SQLite* database. Within this database a table for the step counter is created. This simple table has the following entries: The id as a primary and unique key, a user field for storing the users identity, a field for the steps made during a session called stepCount and the specific time-stamp for the current session. The table is also shown in the figure 4.8. The data is used than for displaying it for user and trainer information as a table and a chart. For the alert, the data is queried regarding the user and the step count is summed up for the specific time frame.

The data of the table can be modified and displayed with the implemented web interface. The storing from the data and the modification happens through SQL statements

like select, insert, update and delete. This is realized with the *Hibernate ORM* framework, which supports the mapping for the object relations with databases in Java.

4.4.2 RESTful Web Service

When the smartphone sends data via HTTP requests to the server, the server side needs to process the incoming information. On the server side a method processes all the incoming messages and triggers the specific method, when the HTTP request includes the correct keyword. In case of an accelerometer message from the smartphone, the server processes the data as follows. At first the given object is split up in its different components. Afterwards the components are checked for plausibility, e.g. if the field for the user contains a String or if the field for the step count contains an integer value as expected. When these checks are successful a new SQL access is issued for inserting the data to the step count table. With a connection to the database and a creation of a new statement, the data is inserted as a new record into the database. After a successful update of the table and the closing of the database connection, the succeeding message is returned to the smartphone as a response. After each access to the database, the ID of the user is checked and all of the database entries are queried. For each of the users exist a threshold, which is dependent on the date, for example the current week or the current month and the goal for the steps taken during this period. If the threshold is reached, an alert is sent via the *XMPP* service to the Android smartphone of the trainer.

4.5 Web Interface Implementation

The web interface for displaying the step counter for the individual users is developed with the *zkos framework*. With this java web framework it is possible to show charts or spreadsheets of server data on web browsers. The implementation for the web interface consists mainly of two components, the table view of the database, where the data is shown and where it can also be modified and the chart view, where the data from the current month queried from the database can be illustrated in form of a line chart. With these two views it is intended to get a better overview of the data.

4.5.1 Table View

The table view page acts as the main page for the step counter visualization and the management of the database entries. It is the root page for the data lookup and handling of the database. Within this page the functions for deleting, inserting, updating and the ability to select data sets and proceed them to the charts section is possible. With the table view it is also possible to change the requested selection of users to show in the desired chart. The basic function of the table view is to show all the data in a table, with a query of the database step counter table, a list of all records available is displayed in this section. This functionality is shown in the figure 4.9.

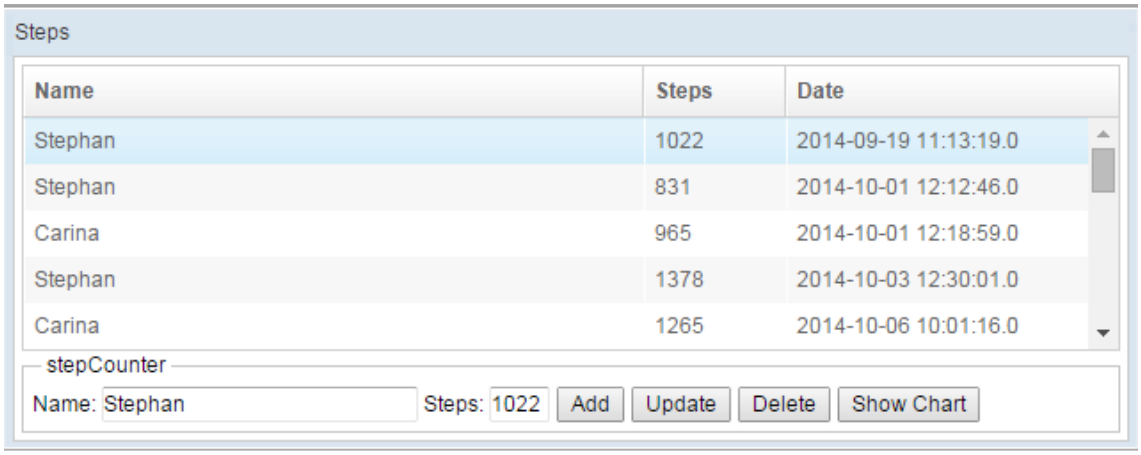

Figure 4.9: Shows the table view of the web interface for the step counter data

The figure 4.9 also shows the management options possible on this page. Selecting an entry in the database, it is possible to delete it by pressing the appropriate button. To be able to perform this action, a transaction defined by the *Hibernate ORM* framework is accomplished. Therefore it is necessary to open a new hibernate session. Within this session, the desired object, which has been selected, is deleted. Committing the transaction and closing the session, finishes the operation. The insertion and the updating functions act similar to the deletion case, where also a session is opened and the process is committed through a transaction.

4.5.2 Chart View

In the chart view of the web interface a graphic processing of the desired data is shown. The figure 4.10 displays the steps made on each day, during the month of November for the specific users. To get the specific user data for the current month, it is necessary to query the database table. Like already described in the previous subsection table view 4.5.1, this happens with the *Hibernate ORM* framework. On the table view page, the desired users are selected which should be shown in the chart, this selection is then forwarded to the chart view, where the displaying of the data is processed. With a new transaction the session needs to be opened. The users, which were selected in the table view are committed as objects to the charts page. Here all data sets, which have the current month are queried from this data sets. In the chart section, at first the title and the axes need to be filled, with the current data. The subtitle includes the name of the current month, the X axis consists of the days from this month. On the Y axis, the number of steps are plotted with an interval of 100 steps, where the minimum is zero and the maximum is adjusted depending on the given data. In the figure 4.10, which is an extract of the chart view page, also the legend of the chart is shown, which names the occurring lines with the user names from the database. After the placement of the chart environment, the data points from the database are loaded and for each entry the point is plotted on the chart.

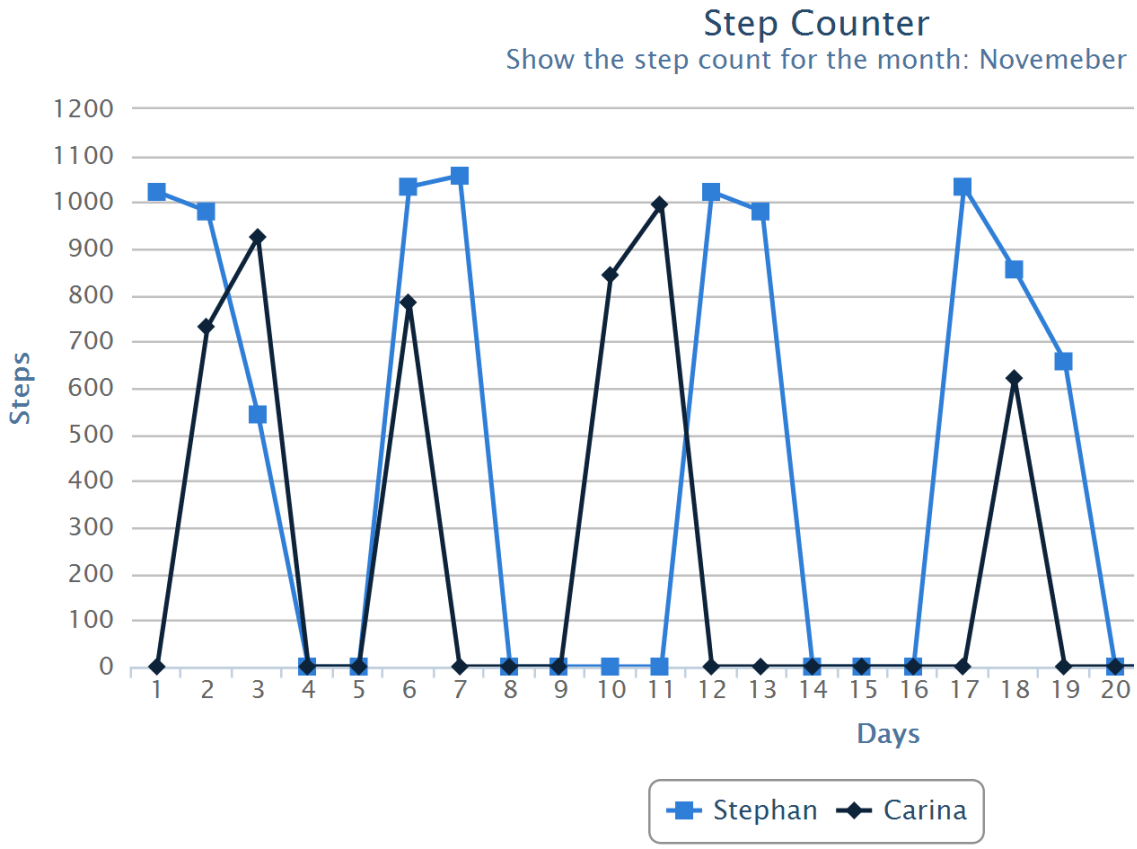

Figure 4.10: Shows an extract of the chart view of the web interface for the step counter data during November for the Users Stephan and Carina

If there exist more than one entry for a day, the steps are summed up. After including all entries, the transaction is committed, the session can be closed and the fully built chart is shown.

4.6 XMPP Android Service Implementation

For sending an alert to a smartphone, the *XMPP* library was included on the server and the smartphone and the implementation for staying connected to the server and receiving an alert was implemented. With this library it is possible to have a constant connection to the server with an Android service running in the background and send an alert regarding a certain event from the server to the smartphone. To get the connection to the server, the library is also included to the Android application on the smartphone. To connect the android phone it is necessary to register the telephone number to the service on the server. To connect to the service, the Android application integrates a tabulator, which also shows the current status of the connection. After a successful connection, the service icon in the status bar lights green.

The connection to the smartphone works with an Android service binder and needs a user, a password and the server name for authentication. This user needs to be already registered to the server, which is managed from the administrator. After the successful connection, the service is started with a new Android Intent. To send a message to the smartphone, the address of the receiver is necessary, this *ID* consists of the user and the server name. If the message could not be sent, because of server problems, the message is stored in the database and will be sent as soon as the connection is re-established. To receive the message on the Android device, an Intent Receiver is necessary. The message will be read with the *IPC* call for the last open message. For this work, the *XMPP* service is used as a reward system. When the user has reached a certain amount of steps during a determined period, a message is sent to the trainer and informs about the reached goal.

Chapter 5

Results

In this Chapter, the results of the work are shown and discussed. The result of the step counter in the two different modes is evaluated and the accuracy is compared to other systems available on the *Android Play Store*. The range of the Bluetooth is analyzed as well as the duration of the system at an excessive use. Tables show a good overview of the results.

5.1 Comparison of Modes

The application, which is developed for this work, is able to detect steps made by the user wearing a smartwatch. The detection works with the accelerometer data sent from the smartwatch to an Android smartphone. The detection of the steps works with a nearest neighbor algorithm, where it compares the accelerometer data during a certain period between idle, a step or a running activity. As the *MetaWatch* smartwatch implements an accelerometer motion detection mode, besides the activity recognition mode a second option to recognize a step is implemented. This motion detection mode triggers the sending of an accelerometer message each time a certain gravitation value threshold is reached. Receiving this accelerometer message on the smartphone, the data is processed and the probability of this data to being a step is calculated. In this section, the two modes are compared for their accuracy.

	Streaming Mode	***Motion Detection Mode***
Average	86%	82%
Best	96%	90%
Worst	77%	74%

Table 5.1: Shows the comparison of step counter for the two different accelerometer modes.

For the motion detection mode, the accuracy average can be seen around 82%. As the table 5.1 shows, the results for the motion detection mode fluctuate between 77% and 90%. This is quite a large span, which can be explained, because of the different movement

of the arm around the wrist. The accuracy depends on whether there is a more heavy movement of the arm or a nearly still movement.

5.2 Comparison of Applications

Nowadays a lot applications exist on the *Android Play Store*, which offer the counting of steps or similar health related topics. To compare the results of the implemented step counter with the two most popular step counter applications on the application store, these three applications are tested under real conditions. The following two applications are chosen for comparison, which are the two most popular applications for the search text 'step counter' in the *Android Play Store*:

- Runtastic Pedometer
- Pedometer 2,0

The procedure for comparing the step counter results works as follows. Each of the systems is tested separately. The step counter is started at the beginning of a predetermined walk with exact 100 steps. After the walk, the three applications are compared on how many steps were recognized and if there were too less or too many steps counted. This procedure has been repeated 10 times and an average result of the recognized steps has been calculated. In the table 5.2 it is shown how accurate these applications have been with the tested setup.

	Implemented Work	***Runtastic Pedometer***	***Pedometer 2,0***
Average	84%	94%	72%
Best	96%	98%	86%
Worst	75%	91%	45%

Table 5.2: Shows the comparison of the accuracy between this work and two different step counter applications.

The best result between the three tested applications accomplished the app *Runtastic Pedometer*. When the smartphone is carried in the pocket of the trousers the application reached the best results. With a recognition rate of 98% for the best result the application was the most precise. The average step recognition rate has been calculated with 94% for this application. The other tested application *Pedometer 2,0* from the *Google Play Store* did not take as good results as the *Runtastic* application. This application needs to be calibrated before using it for the first time. For this application the best recognition rate was achieved by carrying the smartphone in one hand, holding it in a vertical position. With this position the application reached a precision of about 86%. The average step recognition rate for this application has been measured with about 72%. For the application, which has been implemented in this work the results vary with the movement of the watch, respectively the movement of the accelerometer. It has a strong dependency

on the training data. How the training data has been collected, in which direction the accelerometer was adjusted, how did the different training set for idle state, walking and running vary and how much movement of the accelerometer has been made during the training and test phase influence the results of this system. The best recognition rate has been achieved with a precision of 96%, the worst test was executed with only 75%. After all tests, the average recognition percentage was about 84%. With the streaming approach, the algorithm compares the different standard deviation of the axes and the ranges of the axes. When the watch is on the wrist, the user usually moves it a lot, which results in many different orientation towards the accelerometer. This leads to a wide variation in resulting standard deviations, so that the most significant differentiation point is the range of the axes. The main point is the gravitation value, which is also regarded in the motion detection mode of the accelerometer.

5.3 Bluetooth Range

The range of the Bluetooth connection for the smartwatch and the smartphone is evaluated. As it is necessary to know the maximum distance, where the watch still has a connection to the phone, this information is shown in this section. Because of the circumstances that people do not wear the smartphone during the whole day, it is necessary to have a stable connection running the accelerometer. The connection was first tested outdoors and the best result was achieved without obstacles in between. With these conditions the maximum distance, where the smartphone and the watch still have a connection is measured with 12 meters. When wearing the smartwatch inside the house many obstacles can be in the way like furniture or walls. In the indoor test location, the walls are pretty massive and the connection can not be upheld between two walls. Depending on the obstacles between the two receivers also a maximum of 12 meters could be achieved indoors.

5.4 Duration of the System

Another important factor for a system is the duration of the battery and how long the system works during operation. The run time of the system without using the accelerometer is compared with the run time with accelerometer streaming mode and accelerometer motion detection mode using a certain threshold. Also the run time between using the old Bluetooth 2.1 protocol and running the system with Bluetooth 4.0 is compared. For this test, the smartwatch is fully charged before each trial. The table 5.3 shows the duration of the smartwatch, when the smartwatch is connected to the smartphone for the two different modes and without using the accelerometer at all. As the table 5.3 shows, the run time of the smartwatch without using the accelerometer is measured with about 160 hours, which comes very close to the manufacturers information of about 168 hours. The results are about 10 times better than for the motion detection mode and a permanent wearing of the

watch. With the streaming mode enabled, the smartwatch sends accelerometer messages permanently and therefore the battery of the smartwatch is reduced from 14 hours to about 8 hours, what is about 75% worse than in motion detection mode, which can be seen in the table 5.3.

	Time in hours
Usual Runtime	160 h
Runtime Streaming Mode	8 h
Runtime Motion Detection Mode	14 h

Table 5.3: Shows the duration of the system in the mode without accelerometer, with accelerometer in streaming mode and accelerometer in motion detection mode.

Connecting the smartwatch with the *LG G2* Android phone, the smartwatch communication is also tested with *Bluetooth 4.0*. Because the *Nexus S* does not have a *Bluetooth 4.0* module it is necessary to take another smartphone into account. In the table 5.4 it is shown how the duration between the two Bluetooth modes is affected. As it can be seen in the table 5.4, the result for the *Bluetooth 4.0* connection is almost double the time than with the connection of the *Bluetooth 2.1* module. This can be attributed to the performance and power improvements of the *Bluetooth Low Energy* protocol.

	Time in minutes
Bluetooth 2.1	8 h
Bluetooth 4.0	15 h

Table 5.4: Shows the duration of the system in the Bluetooth mode 2.1 and Bluetooth mode 4.0.

Chapter 6

Future Work

In this Chapter the possible future work is discussed and how this could improve the project with additional work and show new perspectives for other projects. It is divided into three different sections, at first the improvements are explained, which can be done to help gain better results. The second section takes care of including new wearables into the system like smartwatches regarding the Android Wear operating system. The third section deals with adding additional features to the work like addressing new sensors of the wearables.

6.1 Improvement

Although this work can be seen as completed, there are still possible concepts to improve it. The first approach to improve the work would be to extend the activity monitoring and obtain the feature to detect a potential fall of a person. With this feature it would be possible to promote the work as a health monitoring project, which can monitor elderly people and alert in case of an emergency. Another option to enhance the project could be the improvement of the battery performance of the wearable. As the *MetaWatch* smartwatch has shown some performance issues sending accelerometer data continuously an easy option for improvement would be a hardware upgrade for the sensor, which would result in adding a new wearable and replacing the smartwatch.

6.2 New Wearables

To support multiple wearables in the systems it is necessary to have a generic interface. This would allow to include new wearables to the system. As the current version of the work only supports the *MetaWatch* wearable, because the Android application is basically built on the community edition of the *MetaWatch Manager*, it would be necessary to implement a generic manager, which is capable of connecting and handling different wearables. One option would be to modify the Android application and support the connection to the *Android Wear* ecosystem. *Android Wear* offers an ecosystem including

several different smartwatches from various manufacturers, which basically implement the same operating system with various modifications. The different smartwatches offer various sensors, like for example *GPS* or a heart rate sensor.

6.3 Possible Additional Work

Additional to the accelerometer sensor, which is addressed in this work, other sensors can be addressed to enhance the project and provide a wide sensor monitoring network. As the system provides an interface to obtain sensor data, it should be possible to address different sensors and extend the work with for example a monitoring of the heart rate or a location service reading the *GPS* signal of the smartwatch. Beside the implementation part, which is necessary for this additional work, also the hardware needs to match the integration of the different sensors. If adding the various sensor checks to the system, the performance of the battery needs to be considered. As the smartwatches only contain a small battery compared to the battery of a smartphone, the continuous request for sensor data will drain the battery faster. A concept has to be developed to keep the battery drain at a minimum.

Chapter 7

Conclusion

This work presents the integration of a wearables in a private mobile cloud service. With the integration of the wearable device it is possible to transmit sensor data via a connected smartphone to the private cloud service. As the wearables gain popularity in the electronic market, it is necessary to research and develop projects regarding this devices like it is done in this work. The project developed in this work shows a system, which is capable of receiving accelerometer data from a smartwatch and process it in a way that it is capable of recognizing activities regarding the movement of the person wearing the smartwatch. Through the analysis of the accelerometer signal from the smartwatch it is possible to map the samples to a certain activity, which has been trained before. The application of a step counter is presented, where the smartphone works as a central point between the transmitting smartwatch and the private mobile cloud, which processes the accelerometer data and sends the results of the step counter to the server. The information is further processed on the server side and stored in a database.

With a front end application for the web browser it is possible to display the data in a table or show a chart of the last steps made by the different users in the current month. The usage of an Android service allows the system to alert a third person smartphone, which is triggered through the analysis of the data stored on the server, when the steps reach a predefined goal. The results of the step counter show that it can compete against other step counter applications from the application store and other systems shown in related works. Although the sensor axes change frequently through the adjustment on the wrist of a person, the detection of a step has shown some good results. Nonetheless the results could improve, for example if other sensors would be applied on different body parts where the axes of the accelerometer would be more stable and deliver more reliable results. The system also supports the addition of new wearables and the possibility to extend the variety of sensors. Within this work it is shown a system, which allows the integration of a wearable device in a private mobile cloud and process the sensor data provided for a health monitoring system.

Appendix A

Definitions

A.1 Abbreviations

API Application Programming Interface

GPS Global Positioning System

RSSI Received Signal Strength Indication

RFID Radio Frequency Identification

XMPP Extensible Messaging and Presence Protocol

KNN K-Nearest Neighbor

SVM Support Vector Machine

ANN Artificial Neural Network

BLE Bluetooth Low Energy

Wi-Fi Wireless Fidelity - any wireless local area network

WPS Wi-Fi Simple Config

GSM Global System for Mobile Communications

VPN Virtual Private Network

NIST National Institute of Standards and Technology

JSON JavaScript Object Notation

HTTP Hypertext Transfer Protocol

openAMI open source AMbient Intelligence

OLED Organic Light Emitting Diode

LCD Liquid Crystal Display

RAM Random Access Memory

OS Operating System

RTOS Real Time Operating System

SoC System on Chip

TI Texas Instruments

MWM MetaWatch Manager

FFT Fast Fourier Transformation

TCP Transmission Control Protocol

JDK Java Development Kit

SDK Software Development Kit

REST Representational State Transfer

IDE Integrated Development Environment

CCS Code Composer Studio

GPU Graphics Processing Unit

NFC Near Field Communication

URI Unique Resource Identifier

SSL Secure Sockets Layer

SPP Serial Port Profile

GUI Graphical User Interface

NNR Nearest Neighbor

CRC Cyclic Redundancy Check

ORM Object-Relational Mapping

IPC Inter-Process Communication

Bibliography

[Bao03] L. Bao. *Physical Activity Recognition from Acceleration Data under Semi-Naturalistic Conditions*. PhD thesis, 2003.

[BFG13] Gerald Bieber, Nicole Fernholz, and Mirko Gaerber. Smart Watches for Home Interaction Services. *HCI International 2013 - Posters Extended Abstracts*, pages 293–297, 2013.

[BHE00] N Bulusu, J Heidemann, and D Estrin. GPS-less Low-Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, 7(5):28–34, 2000.

[BI04] Ling Bao and SS Intille. Activity Recognition from User-Annotated Acceleration Data. *Pervasive computing*, pages 1–17, 2004.

[BMW11] Simon Bradshaw, Christopher Millard, and Ian Walden. Contracts for Clouds: Comparison and Analysis of the Terms and Conditions of Cloud Computing Services. *International Journal of Law and Information Technology*, 19(3):187–223, 2011.

[BOL07] A. K. Bourke, J. V. O'Brien, and G. M. Lyons. Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm. *Gait & posture*, 26(2):194–9, July 2007.

[BP00] P. Bahl and V.N. Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, 2:775–784, 2000.

[BVU09] Gerald Bieber, Jörg Voskamp, and Bodo Urban. Activity Recognition for Everyday Life on Mobile Phones. In *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments*, pages 289–296. Springer Berlin Heidelberg, 2009.

[CKC+05] Jay Chen, Karric Kwong, Dennis Chang, Jerry Luk, and Ruzena Bajcsy. Wearable Sensors for Reliable Fall Detection. *27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, 4:3551–4, January 2005.

[Cri00] J. Cristianini, N. ; Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. 2000.

[DBY+10] Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen, and Dong Xuan. PerFallD: A Pervasive Fall Detection System Using Mobile Phones. *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 292–297, March 2010.

[DJRW03] T. Degen, H. Jaeckel, M. Rufer, and S. Wyss. SPEEDY: A Fall Detector in a Wrist Watch. *Seventh IEEE International Symposium on Wearable Computers, 2003. Proceedings.*, pages 184–187, 2003.

[GBY91] G.H. Gonnet and R. Baeza-Yates. *Handbook of Algorithms and Data Structures*. Addison-Wesley, 1991.

[JMOdG05] Emil Jovanov, Aleksandar Milenkovic, Chris Otto, and Piet C de Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of neuroengineering and rehabilitation*, 2(1):6, March 2005.

[Kio11] Kionix. Kionix Tri-axis Digital Accelerometer Specifications KXTF9-2050. Technical report, 2011.

[Kru95] Philippe Kruchten. Architectural Blueprints The 4+1 View Model of Software Architecture. *Tutorial Proceedings of Tri-Ada*, 12:42–50, 1995.

[KWM11] JR Kwapisz, GM Weiss, and SA Moore. Activity Recognition using Cell Phone Accelerometers. *ACM SIGKDD Explorations Newsletter*, 12(2):74–82, 2011.

[KZX+11] Matthew Keally, G Zhou, Guoliang Xing, J Wu, and Andrew Pyles. PBN: Towards Practical Activity Recognition Using Smartphone-Based Body Sensor Networks. In *SenSys '11 Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, pages 246–259, 2011.

[LC09] Young-Dong Lee and Wan-Young Chung. Wireless sensor network based wearable smart shirt for ubiquitous health and activity monitoring. *Sensors and Actuators B: Chemical*, 140(2):390–395, July 2009.

[LSH+09] Qiang Li, John a. Stankovic, Mark a. Hanson, Adam T. Barth, John Lach, and Gang Zhou. Accurate, Fast Fall Detection Using Gyroscopes and Accelerometer-Derived Posture Information. *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, pages 138–143, June 2009.

[LTKY05] B Lo, Surapa Thiemjarus, Rachel King, and GZ Yang. Body Sensor Network - A Wireless Sensor Platform for Pervasive Healthcare Monitoring. *IEEE International Conference on Pervasive Computing and Communications*, pages 77–80, 2005.

[Man97] Steve Mann. Wearable Computing: A First Step Toward Personal Imaging. *Computer*, 30(February):25–32, 1997.

[Man14] Steve Mann. *Wearable Computing*. The Interaction Design Foundation, Aarhus, Denmark, 2014.

[MDRH14] Alex Migicovsky, Zakir Durumeric, Jeff Ringenberg, and J Alex Halderman. Outsmarting Proctors with Smartwatches : A Case Study on Wearable Computing Security. *Proc. 18th Intl. Conference on Financial Cryptography and Data Security*, (March), 2014.

[Met11] MetaWatch. MetaWatch System Overview Pre-Release Document Version A. Technical report, 2011.

[MG11] P Mell and T Grance. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. Technical report, 2011.

[MLC14] María José Morón, Rafael Luque, and Eduardo Casilari. On the Capability of Smartphones to Perform as Communication Gateways in Medical Wireless Personal Area Networks. *Sensors (Basel, Switzerland)*, 14(1):575–94, January 2014.

[Moo91] Andrew Moore. A tutorial on kd-trees. Extract from PhD Thesis, 1991. Available from http://www.cs.cmu.edu/*sim*awm/papers.html.

[MPR08] Prashanth Mohan, VN Padmanabhan, and R Ramjee. Nericell: Rich Monitoring of Road and Traffic Conditions using Mobile Smartphones. *SenSys '08 Proceedings of the 6th ACM conference on Embedded network sensor systems*, 2008.

[MSJ11] Mladen Milošević, MT Shrove, and Emil Jovanov. Applications of Smartphones for Ubiquitous Health Monitoring and Wellbeing Management. *Journal of Information Technology and Applications*, 1:7–15, 2011.

[MVFB10] Eladio Martin, O Vinyals, Gerald Friedland, and R Bajcsy. Precise Indoor Localization Using Smart Phones. In *Proceedings of the international conference on Multimedia*, pages 787–790, 2010.

[NF07] N Noury and A Fleury. Fall detection - Principles and Methods. *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 1663–1666, 2007.

[NK02] Chandra Narayanaswami and Noboru Kamijoh. IBM's Linux Watch: The Challenge Of Miniaturization. *Computer*, Volume 35;(January):33–41, 2002.

[NVZB12] Nima Nikzad, Nakul Verma, Celal Ziftci, and Elizabeth Bales. CitiSense: Improving Geospatial Environmental Assessment of Air Quality Using a Wireless Personal Exposure Monitoring System. In *WH '12 Proceedings of the conference on Wireless Health*, 2012.

[Pet09] L. E. Peterson. K-nearest neighbor. 4(2):1883, 2009. revision 136646.

[PK10] Jeongyeup Paek and Joongheon Kim. Energy-Efficient Rate-Adaptive GPS-based Positioning for Smartphones Categories and Subject Descriptors. *MobySys'10 the 8th Annual International Conference on Mobile Systems*, pages 299–314, 2010.

[PTA06] SN Patel, KN Truong, and GD Abowd. Powerline Positioning: A Practical Sub-Room-Level Indoor Location System for Domestic Use. *UbiComp 2006: Ubiquitous Computing*, pages 441–458, 2006.

[RC12] Anshul Rai and KK Chintalapudi. Zee: Zero-Effort Crowdsourcing for Indoor Localization. *MobiCom'12 - The 18th Annual International Conference on Mobile Computing and Networking*, 2012.

[RDML05] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and ML Littman. Activity Recognition from Accelerometer Data. *American Association for Artificial Intelligence*, pages 1541–1546, 2005.

[Ric] Richard Munn. MetaWatch Manager. `https://github.com/benjymous/MWM-for-Android/tree/master`.

[RWR14] Nirupam Roy, He Wang, and Romit Roy Choudhury. I am a Smartphone and I can Tell my User's Walking Direction. *Proceedings of the 12th annual international conference on Mobile systems, applications, and services - MobiSys '14*, pages 329–342, 2014.

[Sch04] Bernd Schöne. Electronic Ears. `http://www.siemens.com/innovation/pool/en/publikationen/publications_pof/PoF_Fall_2004/hearing_aids/PoF104art01_1230369.pdf`, 2004.

[SIG10] Bluetooth SIG. Specification of the Bluetooth System Version 4.0. Technical Report June, 2010.

[SSSS13] Dong-min Shin, Dongil Shin, Dongkyoo Shin, and Smart Care Service. Smart Watch and Monitoring System for Dementia Patients. *8th International Conference, GPC 2013 and Colocated Workshops, Seoul, Korea, May 9-11, 2013. Proceedings*, pages 577–584, 2013.

[ST09] Frank Sposaro and Gary Tyson. iFall: An Android Application for Fall Monitoring and Response. *Conference proceedings : Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, pages 6119–22, January 2009.

[TIL04] EM Tapia, SS Intille, and K Larson. *Activity Recognition in the Home Using Simple and Ubiquitous Sensors*. 2004.

[Yan09] Jun Yang. Toward Physical Activity Diary : Motion Recognition Using Simple Acceleration Features with Mobile Phones. In *IMCE '09 Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*, pages 1–9, 2009.

[Yeg09] B. Yegnanarayana. *Artificial Neural Networks*. PHI Learning Pvt. Ltd., 2009.

[YJS12] WJ Yi, Weidi Jia, and Jafar Saniie. Mobile Sensor Data Collector using Android Smartphone. In *IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 956–959, 2012.

[ZJ02] MJ Zieniewicz and DC Johnson. The Evolution of Army Wearable Computers. *IEEE Pervasive Computing*, 1(4):30–40, 2002.