Entwicklung eines Aktivitätserkennungssystems für Trimm-dich-Pfade für Android-Smartphones

Development of an Activity Recognition System for Fitness Trails using an Android Phone

Bachelor-Thesis von Sebastian Niederhöfer
Prüfer: Prof. Alejandro Buchmann
Betreuer: Christian Seeger
04.02.2013

TECHNISCHE UNIVERSITÄT DARMSTADT

Department of Computer Science
Databases and Distributed Systems Group DVS

Entwicklung eines Aktivitätserkennungssystems
für Trimm-dich-Pfade für Android-Smartphones
Development of an Activity Recognition System for Fitness Trails using an Android Phone

Vorgelegte Bachelor-Thesis von Sebastian Niederhöfer
Prüfer: Prof. Alejandro Buchmann
Betreuer: Christian Seeger

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 04. Februar 2013

(Sebastian Niederhöfer)

Zusammenfassung

In den 1970er Jahren entstanden im Rahmen der “Trimm-dich-Bewegung” viele städtische Trimm-dich-Pfade, die zur Erhaltung und Verbesserung des Gesundheitszustandes der Bevölkerung beitragen sollten. Auch in heutiger Zeit werden Trimm-dich-Pfade weiter aktiv genutzt. Aufgrund der heute allgegenwärtigen Nutzung von Smartphones, sind auf Basis der integrierten Sensorik viele Applikationen entstanden, die sportliche Übungen erfassen und erkennen können. Somit kann die Leistung von sportlichen Betätigungen eigenständig und objektiv durch den Sportler analysiert werden.

Im Rahmen dieser Bachelor-Arbeit wurden mehrere drahtlose Beschleunigungssensoren an eine Smartphone-Applikation angebunden, um Trimm-dich-Pfad Übungen und die Anzahl der Wiederholungen erfassen zu können. Hierzu trägt der Sportler Beschleunigungssensoren an Handgelenk, Knöchel und Brust. Die Applikation analysiert die drahtlos übertragenen Daten und erkennt automatisch die Anzahl der durchgeführten Wiederholungen. Ein Motivationsmodul gibt die Anzahl der noch zu leistenden Wiederholungen über eine Text-to-Speech Ausgabe aus und hilft dem Sportler seine Leistung zu steigern. Dabei greift das Modul auf vorher erbrachte Übungsleistungen, welche in einer Datenbank abgespeichert werden, zurück. Neu erbrachte Leistungen werden zudem in der Datenbank gespeichert.

Inhaltsverzeichnis

1 Einleitung

Im Zuge der “Trimm-dich-Bewegung” in den 1970er Jahren wurden in vielen Städten und Gemeinden in Deutschland Trimm-dich-Pfade zur körperlichen Ertüchtigung eingerichtet. Diese Pfade bestehen aus Laufstrecken, an deren Rand in gewissen Abständen Stationen für verschiedene Fitnessübungen wie Liegestütz, Klimmzüge, Bockspringen oder ähnliches eingerichtet sind. Mit dem wirtschaftlichen Aufschwung, den die Bundesrepublik Deutschland in den 1960er Jahren erfuhr, verzeichneten die lokalen Gesundheitsorgane eine merkliche Zunahme von Erkrankungen, die auf eine, mit dem steigenden gesellschaftlichen Wohlstand verbundene, wenig gesundheitsorientierte Lebensweise der Bevölkerung zurückzuführen war. Diese waren und sind vor allem Herz-Kreislauf Erkrankungen, Rückenprobleme als auch körperliches Übergewicht. Als Reaktion versuchte man mit der “Trimm-dich-Bewegung” einen Gegenpol zu etablieren. Dem Bürger sollten Anreize und Möglichkeiten geschaffen werden einen sportlichen Ausgleich finden zu können. Obwohl dieser Gedanke in der Zwischenzeit zunehmend in Vergessenheit geraten ist, bieten Trimm-dich-Pfade nach wie vor hervorragende Möglichkeiten ein sportlich vielfältiges Trainingsprogramm zu absolvieren. Gerade in Zeiten, in denen Bürotätigkeiten immer mehr in den Vordergrund rücken, ist es notwendig einen aktiven Ausgleich zu den psychischen Herausforderungen des Alltags zu finden. Ebenso erlauben Trimm-dich-Pfade die Nähe zur Natur zu wahren und haben damit einen entscheidenden Vorteil gegenüber räumlich geschlossenen Fitnesseinrichtungen.

Aufgrund der Mischung von Ausdauerbetätigungen mit Kräftigungsübungen sind Trimm-dich-Pfade vor allem für ein ausgewogenes ganzheitliches Training geeignet. Um den Trainingseffekt nachvollziehen zu können, gab es seit Einführung der Trimm-dich-Pfade lediglich die Möglichkeit die Trainingserfolge schriftlich festzuhalten. Interessant ist dabei vor allem wie viele Wiederholungen pro Übung durchgeführt wurden. Eine Aufzeichnung per Hand erfordert allerdings, dass das Training unterbrochen wird. Es entstehen negative Auswirkungen auf den Fluss und des Trainings und Beeinträchtigungen des Vergnügungsfaktors. Außerdem kann der Trainingseffekt entsprechend geringer ausfallen, wenn häufige Aufzeichnungspausen das Training unterbrechen.

In der heutigen Zeit besitzen viele Menschen ein Smartphone und tragen es zu jeder Zeit mit sich. Aus diesem Grund ist ein Einsatz dieser Technologie, im Rahmen der Aufzeichnung von Übungen von Trimm-dich-Pfaden, sinnvoll. Die verschiedenen Ein- und Ausgabemodalitäten eines Smartphones bieten hierbei die Möglichkeit mit Hilfe von multimodaler Sensordatenfusion die Anzahl der durchgeführten Übungen zu zählen und umfangreiche Auswertungen zu erstellen. Diese Auswertungen können den Sportler somit unterstützen seine Leistung kontinuierlich zu verbessern, da sie eine höchst objektive Messung der Aktivitäten ermöglichen. Um die Übungen sinnvoll erfassen zu können, können die im Smartphone integrierten Sensoren, wie Touch-Screen-Oberfläche oder Beschleunigungssensor genutzt werden. Da diese Daten allerdings nur an einem Punkt des Körpers erfasst werden können, ist es sinnvoll weitere drahtlose Sensoren an ein Smartphone anzubinden. Dies kann zum Beispiel über die häufig im Smartphone integrierten Bluetooth-Kommunikationsmodule geschehen. Durch den Einsatz mehrere Sensoren kann der Detailgrad und die Klassifikationsgenauigkeit somit verbessert werden.

Neben der Auswertung von Wiederholungen und der aktuellen Übung ist es ebenfalls möglich gezielt Einfluss auf die Leistung des Sportlers zu nehmen, wenn er dies wünscht. Diese Funktionalität bieten Motivationsmodule, die zum Beispiel die Anzahl der noch zu leistenden Wiederholungen über eine Sprachausgabe ausgeben. Somit kann eine kontinuierliche Verbesserung der geleisteten Übungen erzielt werden, ohne dass die Person das Smartphone-Display im Blick behalten muss.

In dieser Bachelorarbeit wurde ein System entwickelt, das in der Lage ist Aktivitäten auf den Trimm-dich-Pfaden durch Fitnessmonitoring während der Durchführung zu erkennen. Hierzu wurden drahtlose Beschleunigungssensoren mit einer Bluetooth-Kommunikationseinheit an ein Android Smartphone angebunden. Die Sensoren werden am Handgelenk, am Knöchel und an der Brust getragen. Für das Smartphone wurde eine Applikation entwickelt, die in der Lage ist, die aktuell durchgeführte Übung, sowie die Anzahl der Wiederholungen automatisiert zu erkennen. Zudem kann der Verlauf aller vergangenen Übungsdurchführungen erfasst werden, um eine sinnvolle Motivationsunterstützung geben zu können. Hierbei wurde eine Sprachausgabe implementiert, die die Anzahl der noch durchzuführenden Übungen herunter zählt. Da die Übungen von Person zu Person in der Art und Weise der Ausführung variieren können, erlaubt das entwickelte Programm das personenbezogene Training der Übungen. Damit wird die erreichte Erkennungsgenauigkeit gesteigert.

Diese Arbeit ist wie folgt strukturiert: Kapitel 2 zeigt „Related Work“ im Bezug auf diese Arbeit auf. Dabei werden bereits bestehende Android-Apps für den Fitness-Bereich und deren Anwendungsmöglichkeiten beschrieben. Des Weiteren wird auf Arbeiten eingegangen, die sich bereits mit beschleunigungssensorbasierter Aktivitätserkennung im Fitnessbereich befassen. Diese Arbeiten bilden die Grundlage, auf der diese Arbeit aufbaut. Kapitel 3 geht auf die Programmierung der Beschleunigungssensoren für die Nutzung auf einem Trimm-dich-Pfad ein. Kapitel 4 beschäftigt sich mit dem Aufbau der entwickelten Android-App. Das Kapitel beschreibt, wie die App zu nutzen ist und welche Verfahren zur Speicherung, Erkennung und Zählung von Übungen genutzt werden. Kapitel 5 befasst sich mit dem Aufbau und der Nutzung des entwickelten Motivations- und Datenbankmoduls, das die entwickelte App erweitert und auch für die Nutzung in zukünftigen Fitness-Anwendungen für Android-Smartphones konzipiert wurde.

2 Related Work

Das folgende Kapitel beschäftigt sich mit Related Work zu dieser Arbeit. Im ersten Abschnitt werden bereits bestehende Smartphone-Apps behandelt, die sich im Fitnessbereich ansiedeln. Es wird einen Überblick gegeben, welche Funktionalitäten diese Apps zu Verfügung stellen und wo ihre Grenzen liegen. Im zweiten Abschnitt werden Arbeiten behandelt, die sich mit der Aktivitätserkennung mittel Beschleunigungssensoren und deren Nutzen im Fitnessbereich befassen. Diese Arbeiten bilden eine Grundlage für diese Arbeit.

2.1 Bestehende Fitness Smartphone-Apps

Bisher bekannte Smartphone-Apps, die den Anwender beim Absolvieren von Fitnesseinheiten unterstützen, lassen sich grundlegend nach den drei folgenden Kategorien einteilen:

Die erste Kategorie umfasst Apps, die mit Hilfe des GPS-Empfängers die Strecke und Durchführungszeit für Ausdauersportarten aufnehmen. Zu diesen Sportarten zählen unter anderem Laufen, Rad fahren und Walken. Die aufgenommenen Daten werden gespeichert, ausgewertet und schließlich dazu verwendet, den Nutzer auf Basis von bereits erbrachten Leistungen bei einer Übung zu motivieren. Eines der bekanntesten Beispiele hierfür ist *RunKeeper* von FitnessKeeper, Inc.[1]. Wie Abbildung 2.1 und Abbildung 2.2 zeigen, ermöglicht die App eine statistische Auswertung der Trainingserfolge, indem sie Strecken, Zeiten und absolvierte Distanzen aufzeichnet und zudem kartenbasiert darstellt.

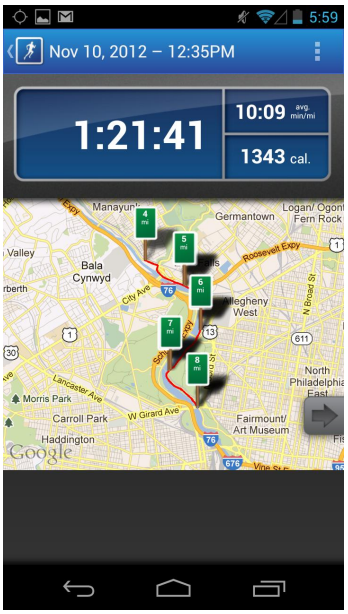

Abbildung 2.1: Kartenbasierte Übersicht der Trainingseinheiten in *RunKeeper*[2]

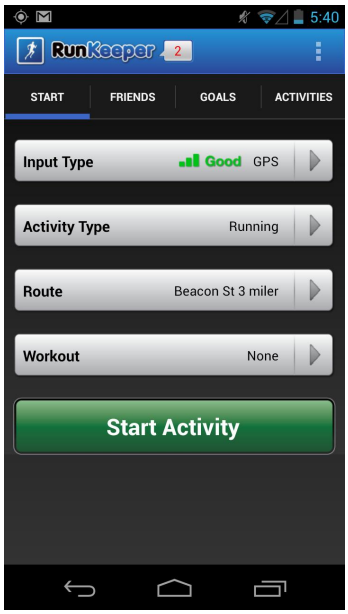

Abbildung 2.2: Menü zum Starten der Aufnahme einer neuen Aktivität in *RunKeeper*[2]

Die Apps der zweiten Kategorie werden zum Erstellen von Trainingsplänen und zum Aufzeichnen von Trainingserfolgen verwendet. Zudem geben sie in Text-, Bild- oder Videoform Anleitungen und Hilfestellungen zur Durchführung von Übungen. Allerdings ist es erforderlich, dass Trainingspläne manuell

1 http://runkeeper.com/
2 https://play.google.com/store/apps/details?id=com.fitnesskeeper.runkeeper.pro

zusammengefügt werden. Ebenso müssen Trainingserfolge per Hand eingetragen werden. Dies erfordert eine stetige Interaktion mit dem Smartphone und stellt einen zeitlichen Mehraufwand dar. Beispielhaft für diese Kategorie ist *JEFIT* von Jefit Inc.[3]. Abbildung 2.3 und Abbildung 2.4 zeigen den Startbildschirm sowie eine Ansicht, in der der User seine Fortschritte bei einer Kraftsportübung einträgt.

Abbildung 2.3: Startbildschirm der App *JEFIT*[4]

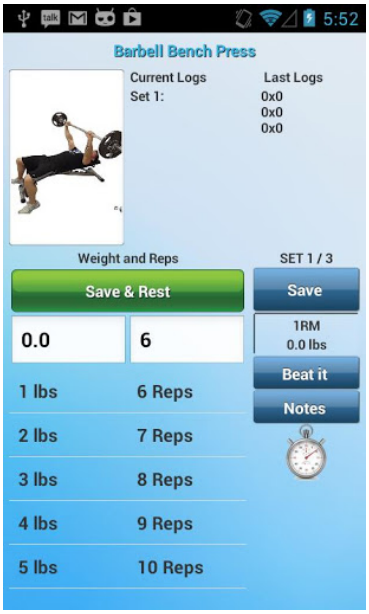

Abbildung 2.4: Trainingsergebnisse mit *JEFIT* speichern[4]

Die dritte App-Kategorie unterstützt einzelne Fitnessübungen und nutzt Bedienelemente und Sensoren des Smartphones, um die Anzahl an Wiederholungen während Durchführung der Übung aufzunehmen. Je nach Art der Übung werden der Touchscreen, eine Neigungserkennung des Smartphones oder ähnliches zur Wiederholungszählung genutzt. *Push ups pro*[4] von Northpark ist ein Beispiel für diese Art von App. Die App zählt die Wiederholungen beim Liegestütz, indem das Smartphone unter dem Gesicht auf den Boden gelegt und bei jeder Durchführung mit der Nase oder dem Kinn der Touchscreen berührt wird. Abbildung 2.5 und Abbildung 2.6 zeigen den Startbildschirm und die Anleitung der App. Jede Berührung wird als Wiederholung gezählt.

Die vorgestellten Apps sind in ihren jeweiligen Funktionen eingeschränkt und konzentrieren sich auf definierte Anwendungsfälle. Daher eignen sie sich nur bedingt oder gar nicht für die Aufzeichnung der Übungen, die auf einem Trimm-dich-Pfad durchgeführt werden. Mit ihnen ist es nicht möglich die Art der Übung während der Durchführung zu erkennen und gleichzeitig die Wiederholungen zu speichern. Der Nutzer muss der App die Umstände der durchgeführten Übung mitteilen und/oder die Trainingsergebnisse manuell einpflegen.

[3] http://www.jefit.com/
[4] https://play.google.com/store/apps/details?id=je.fit

Abbildung 2.5: Startansicht von *Push up pro*[5]

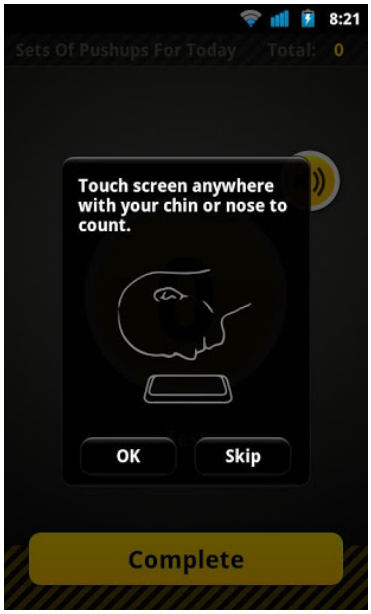

Abbildung 2.6: *Push ups pro* Liegestütz Anleitung[5]

2.2 Entwicklungen in der Forschung

Wie sich bereits in vorangegangenen Arbeiten von Chang et al. [2] und Seeger et al. [4] gezeigt hat, sind Beschleunigungssensoren für die Erkennung und das Zählen der Wiederholungen von Fitnessübungen eine gute Wahl. In beiden Veröffentlichungen werden Mobiltelefone bzw. Smartphones als Aggregator genutzt, nicht jedoch die Sensoren des Smartphones. Die zur Bewegungsdetektion genutzten Beschleunigungssensoren werden in beiden Ansätzen an fest definierten Postionen am Körper des Nutzers befestigt. Dabei wurden bei Chang et al. für die Aktivitätserkennung zwei Methoden untersucht. Die beiden Ansätze nutzen einen Naive Bayes Classifier sowie ein Hidden-Markov Model. Zur Wiederholungszählung werden ein “Peak-Counting”-Algorithmus und ein Viterbi-Algorithmus genutzt.

Bei Seeger et al. ist die Grundlage für die Erkennung der Art der Fitnessübung ein Gauß-Modell. Mit Hilfe von optimal bestimmten 18-dimensionalen Gauß-Modellen werden extrahierte Sensor-Features der Übung zugeordnet, deren Gauß-Modell den geringsten Abstand zum Feature-Vektor aufweist. Zur Wiederholungszählung werden zwei Peak-Counting Algorithmen eingesetzt. Der erste Algorithmus nutzt eine Peak-Detection auf dem Smartphone. Die für die Zählung der Wiederholungen relevanten Peaks werden dabei, aus den vom Sensor übertragenen Mittelwerten und Varianzen der Beschleunigungsdaten, ermittelt. Die Peak-Detection des zweiten Algorithmus wird bereits auf dem Sensor ausgeführt. Hierzu werden vom Sensor zusätzlich zu den Mittelwerten und Varianzen der Beschleunigungsdaten deren Maxima und Minima bestimmt und übertragen.

Die vorangegangene Arbeit von Pignede [3] beschäftigt sich bereits mit der Aktivitätserkennung und Wiederholungszählung von Übungen an einem FitnessTrail (“Trimm-dich-Pfad”). Pignedes Arbeit untersucht dazu folgende Aspekte:

- Positionierung der Sensoren am Körper des Nutzer
- Gewinnung der relevanten Features vom Sensor
- Algorithmus für die Wiederholungszählung

Die hier gewonnenen Erkenntnisse diesen als Grundlage für diese Arbeit.

[5] https://play.google.com/store/apps/details?id=com.northpark.pushups

3 Sensoren

Das folgende Kapitel beschäftigt sich mit den in dieser Arbeit verwendeten Beschleunigungssensoren. Zunächst wird die verwendete Hardware beschreiben. Im Anschluss erfolgt eine Beschreibung über die Verarbeitung der Beschleunigungsdaten. Hier wird auf die Berechnung des Mittelwertes und der Varianz der Daten, die für die spätere Aktivitätserkennung relevant sind, eingegangen. Darauf folgend wird die Peak-Detektion erläutert, die für die Wiederholungszählung von Fitnessübungen benötigt wird. In Listing A.1 schließt sich der C-Programmcode für die Programmierung der Sensoren an, der die beschriebenen Berechnungen durchführt.

3.1 HedgeHog

Als Sensoren werden zwei oder drei *HedgeHog*-Sensoren[1] verwendet. Die gemessenen Beschleunigungsdaten werden auf dem Mikrocontroller, wie in Abschnitt 3.2 beschrieben, verarbeitet und mittels eines Bluetooth-Moduls an das Smartphone gesendet.

3.2 Programmierung der Sensoren

Die Bachelor Arbeit von Thomas Pignede [3] hat sich mit der Bestimmung geeigneter Merkmale zur Aktivitätserkennung und Wiederholungszählung beschäftigt. Dabei haben sich folgende Merkmale als geeignet herausgestellt:

- Mittelwert der Beschleunigungsdaten (Erkennung)
- Varianz der Beschleunigungsdaten (Erkennung)
- Erkannte Spitzenpaare (Peaks) in den Beschleunigungsdaten (Wiederholungszählung)

Bei dem Ansatz von Pignede werden Mittelwert und Varianz über einen Zeitraum von jeweils vier Sekunden berechnet. Dieser Zeitraum bietet sich an, da die Features so eine ausreichende Genauigkeit bei einem möglichst kleinen Aufnahmezeitraum aufweisen. Zur Berechnung wird ein Sliding-Window Verfahren eingesetzt, das dementsprechend zusätzlich die Werte der letzten drei Sekunden vorhält. Die Berechnung sowie das Senden der Daten erfolgt sekündlich.

3.2.1 Varianz- und Mittelwertberechnung

Da der Sensor bzw. die Sensoren mit einer Frequenz von $100\,Hz$ arbeiten, würde sich mit den exakten Formeln zur Berechnung des Mittelwertes nach Formel 3.1 und zur Berechnung der Varianz nach Formel 3.2 ein Speicheraufwand von insgesamt 400 Werten pro Richtungsachse ergeben, die persistent im Speicher gehalten werden müssten.

Mittelwert:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{3.1}$$

[1] http://www.ess.tu-darmstadt.de/hedgehog

Varianz:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 \tag{3.2}$$

Bei der Verwendung von drei Achsen ergäbe sich somit ein Gesamtanzahl von 1200 Werten. Dies ist jedoch aufgrund der beschränkten Speicherkapazität des *HedgeHog*-Sensors nicht realisierbar und wäre auch im Hinblick auf Berechnungszeit und Anzahl der auszuführenden Berechnungen pro Zeitschritt sehr kostenintensiv.

Zur Lösung des Problems wurde zur Berechnung der Varianz eine Näherung verwendet, die das Ergebnis kontinuierlich approximiert und daher den Speicheraufwand damit beträchtlich reduziert. Der Mittelwert nach Formel 3.3 und die angenäherte Varianz nach Formel 3.4 lassen sich auf die Berechnung der Summe "**sum**" und der Summe der Quadrate "**sumsq**" aller eingehenden Werte pro Achse herunter brechen.

Mittelwert:

$$\overline{x} = \frac{1}{n}\underbrace{\sum_{i=1}^{n} x_i}_{sum} \tag{3.3}$$

Varianz:

$$\sigma^2 \approx \frac{1}{n}(\underbrace{\sum_{i=1}^{n} x_i^2}_{sumsq} - \frac{1}{n}(\underbrace{\sum_{i=1}^{n} x_i}_{sum})^2) \tag{3.4}$$

Somit werden für neu eingehende Sensorwerte deren Summe und deren Summe der Quadrate berechnet und diese dann nach Ablauf einer Sekunde gespeichert. Durch Aufsummieren der Summen (Formel 3.5) und der Summen der Quadrate (Formel 3.6) der vergangenen vier Sekunden, lassen sich nun Mittelwert nach Formel 3.3 und Varianz nach Formel 3.4 berechnen.

Summe:[2]

$$sum = sum_1 + sum_2 + sum_3 + sum_4 \tag{3.5}$$

Summe der Quadrate:[2]

$$sumsq = sumsq_1 + sumsq_2 + sumsq_3 + sumsq_4 \tag{3.6}$$

Der Fehler, der durch die Näherung entsteht, betrifft lediglich die Nachkommastellen der Varianz. Um die via Bluetooth zu übertragende Datenmenge möglichst gering zu halten, werden die Nachkommastellen ohnehin vernachlässigt. Dies bietet sich aufgrund der Größe der Varianzwerte an, da dadurch die Nachkommastellen in späteren Berechnungen ohnehin nur sehr gering ins Gewicht fallen. Daher kann dieser Fehler vernachlässigt werden.

[2] $sum_1, ..., sum_4$ und $sumsq_1, ..., sumsq_4$ stehen für die Summen bzw. Summen der Quadrate jeweils einer der vier Sekunden

3.2.2 Peak-Detection

Während der Durchführung einer Übung entstehen, durch Beschleunigung und Abbremsen der Bewegung, in den vom Sensor gemessenen Daten positive und negative Ausschlagspitzen. Diese Spitzen sind relevant für die Wiederholungszählung, da durch ein Positiv-Negativ-Paar mit einer bestimmten Ausschlagsstärke eine durchgeführte Wiederholung festgestellt werden kann. Das Detektieren dieser Paare übernimmt die Peak-Detection.

In der Arbeit von Seeger [4] wurde die Peak-Detection in einem Ansatz über den Mittelwert- und Varianzdaten ausgeführt, indem nach Spitzenpaaren in Verlauf des Mittelwerts gesucht wurde. Dies ermöglicht durch die $1\,Hz$ Übertragung nur die Detektion von langsam durchgeführten Übungen. Im zweiten Ansatz wurden vom Sensor zusätzlich zu den Mittelwert- und Varianzdaten auch noch bereits auf dem Sensor detektierte Minima und Maxima übertragen, anhand derer dann auf dem Smartphone die Wiederholungsdetektion und schließlich die Zählung erfolgte. Der Zweite Ansatz ermöglicht auch Wiederholungserkennung von schneller durchgeführten Übungen.

In Thomas Pignedes Arbeit [3] wird nun ein Algorithmus beschrieben der auch die Wiederholungserkennung auf den Sensor ausgelagert. Dazu werden bei dem von ihm beschriebenen Peak-Detection-Algorithmus die nur für die Wiederholung relevanten Spitzenpaare erkannt. Dies geschieht, indem bei der Erkennung auch der zeitliche Abstand der der Positiv- und Negativ-Spitzen sowie die Ausschlagsstärke mit einbezogen wird.

Der Algorithmus aus Pignedes Arbeit wurde in dieser Arbeit für die durch die Sensoren gegebene Situation angepasst und für das Sensormodul implementiert. Somit werden nun nicht mehr Maximum- und Minimumpaare pro Achse vom Sensormodul übertragen, sondern nur noch ob ein geeignetes Paar erkannt wurde. Dies verringert die via Bluetooth zu übertragene Datenmenge und erspart eine spätere Verarbeitung auf dem Smartphone. Ein erkanntes Paar wird als ’1’, kein erkanntes als ’0’ kodiert.

3.2.3 Übertragungsformat

Die Daten, die durch die vorgestellten Wege berechnet wurden, werden nun in der folgenden Reihenfolge über die Bluetooth-Verbindung übertragen:

- Mittelwert X-Achse
- Mittelwert Y-Achse
- Mittelwert Z-Achse
- Varianz X-Achse
- Varianz Y-Achse
- Varianz Z-Achse
- Peakdetected X-Achse
- Peakdetected Y-Achse
- Peakdetected Z-Achse

Die Mittelwerte sind jeweils ein Byte lang, genauso wie die Peakdeteced-Werte. Die Varianzen haben eine Länge von 2 Bytes (Int16) wobei das jeweils erste übertragene Byte dem niederwertigen Anteil entspricht (Little Endian). Die übertragene Nachricht hat somit eine Länge von 12 Byte.

3.2.4 Testumgebung

Zum Testen der Programmierung wurde das PIC18F Starter Kit (Abbildung 3.1) als Testbed genutzt.

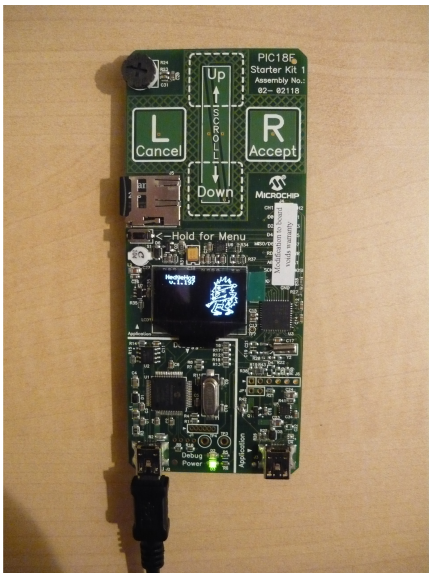

Abbildung 3.1: PIC18F Starter Kit Testbed

Hierzu wurden die eingehenden Beschleunigungswerte des Sensors durch festgelegte Eingangswerte ersetzt und anhand dieser die Ausgabewerte überprüft.

4 Die Android App

Dieses Kapitel beschäftigt sich mit der in dieser Arbeit entwickelten *MyFitnessTrail* App, die dem Nutzer Möglichkeiten bietet während des Trainings auf einem Trimm-dich-Pfad die ausgeführten Übungen automatisch zu erkennen und die Wiederholungen zählen zu lassen. Zudem speichert die App den Trainingsfortschritt und motiviert den Nutzer auf Basis vorangegangener Trainingsergebnisse.

Die App erhält die notwendigen Sensordaten von der MyHealthHub (MHH) App, der zunächst eingehend beschrieben wird. Darauf aufbauend wird die *MyFitnessTrail* App näher beschrieben.

4.1 MyHealthHub

4.1.1 Aufgabe von MyHealthHub

MHH ist eine von Christian Seeger entwickelte Middleware für den Datenfluss von Daten im Bereich der Körper- und Umgebungssensorik. Die eventbasiert gesteuerte Middleware für Android Smartphones empfängt via Bluetooth Sensordaten, die in einem Sensormodul verarbeitet werden. Aus den verarbeiteten Daten wird ein Sensor-Event erzeugt. Dieses wird von MHH über den *BroadcastReceiver* von Android veröffentlicht. Durch MHH wird es Anwendungen, die sich für bestimmte Sensordaten interessieren, ermöglicht, sich für Events ausgewählter Sensoren anzumelden. Wird nun ein solches Sensor-Event auf dem *BroadcastReceiver* veröffentlicht, erhalten alle interessierten Anwendungen die Daten des Sensors automatisch zur Weiterverarbeitung. Das Verfahren folgt einem themenbasierten Publish-Subscribe Pattern.

4.1.2 Anbindung einer Anwendung an MyHealthHub

Wie in Abbildung 4.1 dargestellt, stellt MHH für verschiedene Eventtypen jeweils einen Kanal zu Verfügung. Über diese Kanäle werden Events, mit Hilfe des *BroadcastReceivers*, verteilt. Ist eine Anwendung an

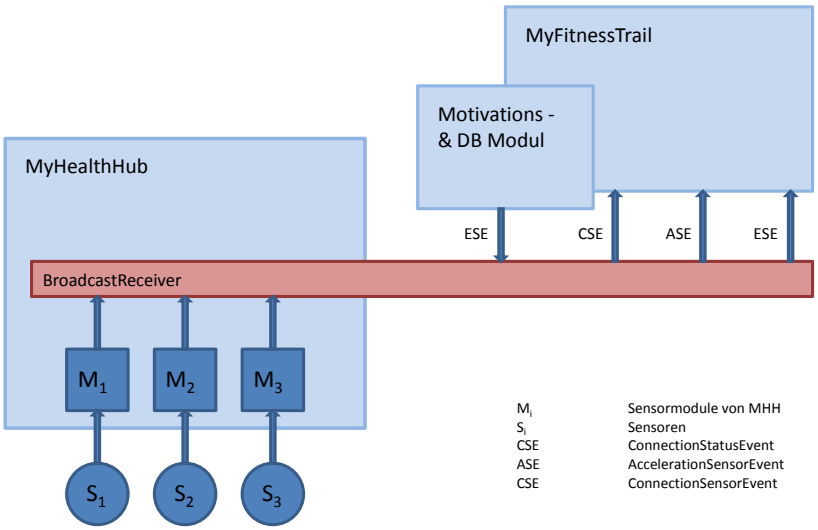

Abbildung 4.1: Anbindung der von MyFitnessTrail an MyHealthHub

einem bestimmten Eventtyp interessiert, subskribiert sie einen *EventReceiver* für den Kanal des Eventtyps.

Der *EventReceiver* empfängt nun alle Events des Eventtyps, die über den Kanal von MHH veröffentlicht werden. MHH ermöglicht es den Anwendungen ebenfalls selbstständig Events zu erzeugen und diese zu veröffentlichen. Der Eventtyp dieser Events muss MHH bekannt sein, sodass diese auch über den korrekten Kanal veröffentlicht werden können.

4.2 MyFitnessTrail App

Für die *MyFitnessTrail* App wird die von Thomas Pignede in seiner Arbeit [3] beschriebenen Positionierungen der Sensoren verwendet. Daher wird angenommen, dass zwei bis drei HedgeHog Beschleunigungssensoren in folgenden Positionsvarianten von einem Nutzer getragen werden:

- Handgelenk-Sensor, Brust-Sensor, Knöchel-Sensor (siehe Abbildung 4.2)
- Handgelenk-Sensor, Knöchel-Sensor (siehe Abbildung 4.3)

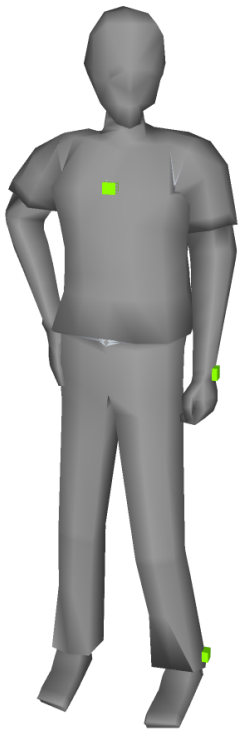

Abbildung 4.2: Positionierung für drei Sensoren

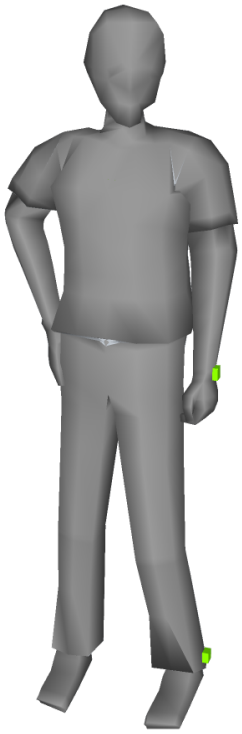

Abbildung 4.3: Positionierung für zwei Sensoren

Die App ermöglicht es für beide Sensorkombinationen Aktivitäten für die spätere Erkennung zu trainieren. Trainierte Aktivitäten werden einem Standort, bspw. einem bestimmten Trimm-dich-Pfad, zugeordnet. Für einen Standort kann eine Menge an trainierten bzw. durchführbaren Aktivitäten in einer Aktivitätsliste als Datei gespeichert werden. Diese gespeicherten Aktivitätslisten können dann zu Beginn eines Trainings standortabhängig geladen und bei der Trainingsdurchführung zur Aktivitätserkennung genutzt werden.

4.2.1 FitnessTrailActivityList

Eine *FitnessTrailActivityList* beinhaltet eine Menge an erkennbaren Aktivitäten, die typischerweise bezogen auf einen Trimm-dich-Pfad gruppiert und gespeichert werden. Für jede Trimm-dich-Pfad Übung, die als Aktivität gespeichert werden soll, werden der Name, das trainierte und sie repräsentierende Gauß-Modell, die Information ob die Übung zählbar ist und der für ihre Zählung genutzte Sensor (Brust, Hand oder Knöchel) als Information für die späteren Berechnungen benötigt.

Die *FitnessTrailActivtiyList*-Klasse (Abbildung 4.4) bietet die notwendige Abstraktion. In einem Objekt dieser Klasse werden alle Aktivitäten hinterlegt, die für den jeweiligen Trimm-dich-Pfad gespeichert werden sollen.

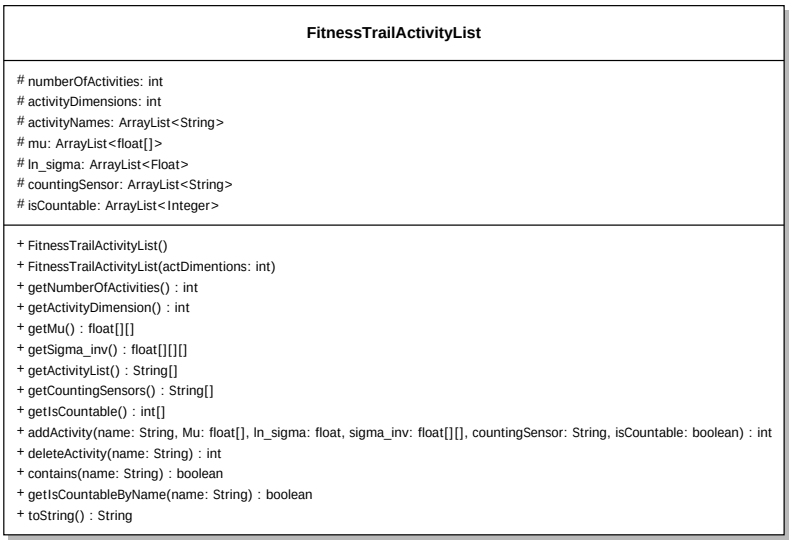

Abbildung 4.4: Klassendiagramm *FitnessTrailActivityList*

In einem solchen Objekt werden sequenziell die relevanten Features aller erkennbaren Aktivitäten abgelegt. Diese Datenstruktur erlaubt während der späteren Klassifikation einen performanten Zugriff. Als zusätzliche Information wird hier noch die Dimension der Gauß-Modelle gespeichert. Diese ist abhängig von der verwendeten Anzahl an Sensoren. Da für eine konkrete FitnessTrailActivityList die Anzahl der Sensoren konstant ist, bleibt die Dimension für eine Liste ebenso konstant. Als Zusatzinformation wird die Anzahl der Aktivitäten, die in dem jeweiligen Objekt hinterlegt sind gespeichert. Dies ist notwendig, um die sequenziell als Arrays abrufbaren Feature Werte korrekt adressieren zu können.

4.2.2 Training neuer Aktivitäten

Zum Training von neuen Aktivitäten werden über einen bestimmten Zeitraum die Mittelwerte und Varianzen, die von der Sensorkonstellation empfangen werden, aufgenommen. Aus diesen Werten wird dann ein Gauß-Modell berechnet, welches die Aktivität klassifiziert. Der Algorithmus hierfür wurde aus Christian Seeger's *MyFitnessDiary*-App übernommen und wird hier als Black-Box behandelt. Die vom Training errechneten Werte werden zu einem Objekt der Klasse *FitnessTrailActivtiyList* hinzugefügt.

4.2.3 Speicherung der Aktivitätslisten

Die *FitnessTrailActivityList*-Objekte, in denen Übungen zum jeweiligen Trimm-dich-Pfad hinterlegt sind, werden zur Speicherung mittels *XStream*[1] als XML-Datei gesichert. Die in Java verfügbare Library ermöglicht es Java Objekte im XML-Format abzuspeichern. Ebenso können die gespeicherten Objekte wieder zur Laufzeit in ein Java Programm geladen werden. *XStream* wurde aufgrund seiner einfachen Handhabung ausgewählt. Außerdem erlaubt das lesbare XML-Format eine manuelle Überprüfung der gespeicherten Daten.

Für jede der beiden möglichen Sensorkonstellationen steht ein Dateisystem Ordner bereit, der automatisch auf der Speicherkarte des Smartphones nach folgendem Schema angelegt wird:

[1] http://xstream.codehaus.org/

- **/Speicherkarte/MyFitnessTrail/ActivityLists3Sensors/**
- **/Speicherkarte/MyFitnessTrail/ActivityLists2Sensors/**

4.2.4 Aktivitätserkennung

Mit Hilfe eines Algorithmus zur Aktivitätserkennung wird entschieden, ob die übermittelten Mittelwerte und Varianzen einer trainierten Aktivität aus der aktuell ausgewählten *FitnessTrailActivtiyList* entspricht. Dies geschieht, indem aus den eingegangenen Werten zu jedem hinterlegten Gauß-Modellen die räumlichen Distanzen errechnet werden. Anhand der errechneten Distanzen wird anschließend entschieden, ob die erfasste Aktion einer bekannten Aktivität zugehörig ist oder nicht. Der Algorithmus wurde aus Christian Seeger's *MyFitnessDiary*-App mit der Klasse *FitnessTrailActivityDetection* als Black-Box übernommen.

4.2.5 Wiederholungserkennung

Zur Erkennung von Wiederholungen einer Aktivität ist es notwendig die Ergebnisse, die von einer Peak-Detection bestimmt werden, auszuwerten. Der ursprüngliche Ansatz dazu wurde, wie bereits in Abschnitt 2.2 vorgestellt, von Seeger et al. vorgeschlagen [4]. Bei diesem Ansatz werden die zur Peak-Detection notwendigen Features auf den verwendeten Sensoren berechnet und per Bluetooh an das Smartphone übertragen. Die eigentliche Peak-Detection wird auf dem Smartphone ausgeführt. Bezogen auf diese Bachelorarbeit bestand die Herausforderung darin den Peak-Detection Algorithmus direkt auf Beschleunigungssensoren durchzuführen.

Obwohl der entsprechende Algorithmus, wie in Kapitel 3 beschrieben, erfolgreich auf dem HedgeHog Evaluationsboard implementiert werden konnte, war die Portierung auf die eigentlichen Sensoren auf Grund von Kompatibilitätsproblemen zwischen der zur Verfügung stehenden Sensor Hard- und Software nicht möglich. Daher wurde als Fallback Lösung der in Christian Seeger's Arbeit [4] vorgestellte 'Exercise Counting on Phone' Ansatz genutzt. Hierbei wird anhand des Verlaufs der Mittelwerte festgestellt, ob eine Wiederholung der erkannten zählbaren Übung durchgeführt wurde.

4.2.6 Wiederholungszählung

Mit Hilfe der, aus Aktivitätserkennung und Wiederholungserkennung sekündlich eintreffenden, Informationen bezüglich erkannter Aktivität und möglicher Wiederholung wurde ein Wiederholungszähler für die Übungen implementiert, der eine Filterung für zwischenzeitlich falsch klassifizierte Aktivitäten beinhaltet. Das Verfahren wird in Abbildung 4.5 veranschaulicht.

Der Wiederholungszähler nutzt zwei Zähler. Der erste Zähler zählt die aktuelle Hauptaktivität. Der zweite Zähler zählt im Hintergrund auch eine vermeintlich falsch erkannte Aktivität. Dabei wird die Hauptaktivität ermittelt, indem die letzten fünf eingetroffenen Aktivitätsmeldungen gespeichert werden und, immer wenn eine neue Aktivitätsmeldung eintrifft, überprüft wird ob zwei der vier gespeicherten Aktivitäten die gleiche Aktivität repräsentieren. Dies formuliert die Forderung, dass drei der letzten fünf Aktivitätsmeldungen identisch sein müssen, um als Hauptaktivität gezählt zu werden. Ist diese Bedingung nicht erfüllt, erhöht die eingehende Aktivität lediglich den Hintergrundzähler. Hauptaktivitäts- und Hintergrundaktivitätszähler sind jeweils mit einer Aktivität verknüpft. Die verknüpfte Aktivität des Hintergrundzählers ändert sich, wenn die eingehende Aktivität nicht der mit dem Hauptzähler verknüpften Aktivität entspricht. Die mit dem Hintergrundzähler verknüpfte Aktivität wird zur Aktivität des Hauptzählers, wenn die bereits beschriebene „drei-aus-fünf" Bedingung erfüllt wurde. Wird die Bedingung verletzt, wird die aktuell eingehende Aktivität zur Aktivität des Hintergrundzählers. Die Ausgabe der Wiederholungszählers wird vom Hauptzähler bereitgestellt.

Für nicht zählbare Aktivitäten, also Übungen für die nicht die Anzahl an Wiederholungen, sondern die Durchführungsdauer relevant ist, wird durch den Wiederholungszähler jede erkannte Aktivität gezählt.

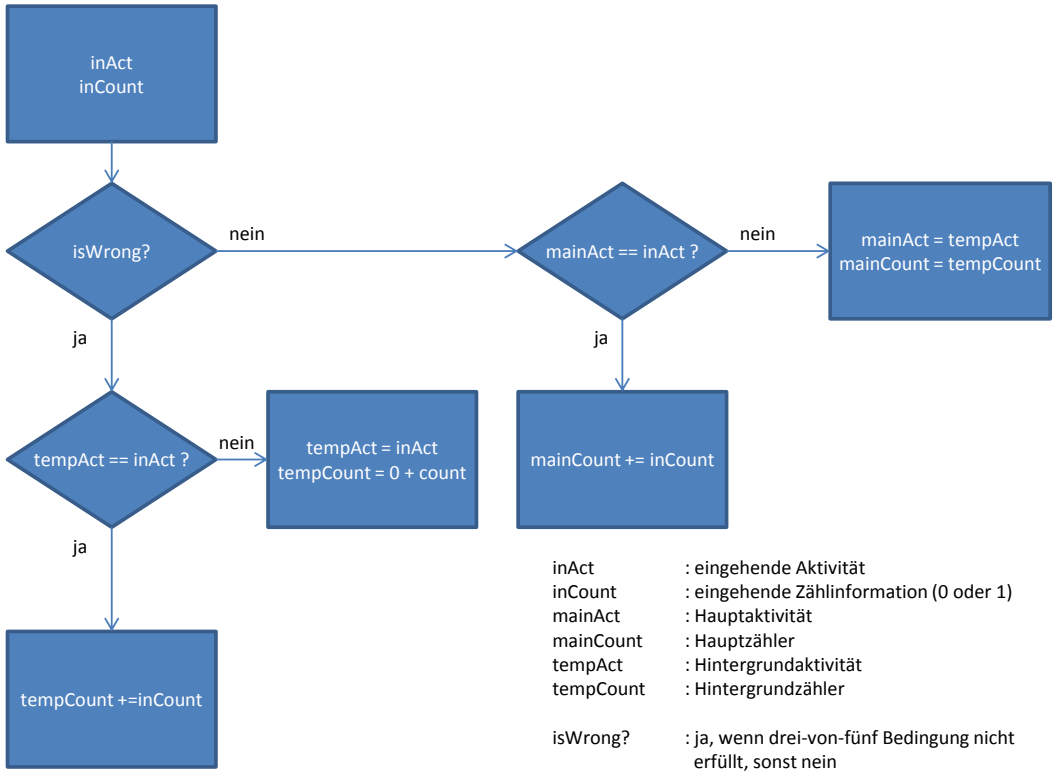

Abbildung 4.5: Flussdiagramm Veranschaulichung des Wiederholungszählers

Somit wird aus den sekündlich eintreffenden Aktivitätsmeldungen aus der Peak-Detection die Dauer der Durchführung ermittelt.

4.2.7 Benutzeroberfläche

Die Benutzeroberfläche von *MyFitnessTrail* gliedert sich in die im Folgenden vorgestellten Benutzerdialoge bzw. Android Activities.

4.2.7.1 Startbildschirm

Nach dem Start der *MyFitnessTrail* App hat der Nutzer auf dem Startbildschirm nach Abbildung 4.6 die Möglichkeit eine zuvor erstellte Übungsliste zu wählen. Alternativ kann er eine neue Übungsliste erstellen, um Aktivitäten von Grund auf anzutrainieren. Zuvor müssen allerdings mindestens zwei Sensoren erfolgreich verbunden worden sein.

Unter Zuhilfenahme der von MHH gelieferten *ConnectionStatusEvents*, wird für die einzelnen Sensoren ermittelt, ob eine der genannten Sensorkonstellationen zur Verfügung steht. Solange dies nicht der Fall ist, kann keine weitere Aktion durchgeführt werden. Um dem Nutzer den aktuellen Verbindungsstatus der Sensoren mitzuteilen, wird dieser durch farbliche Indikatoren dargestellt. Dabei sind verbundene Sensoren grün, nicht verbundene Sensoren rot markiert. Sobald eine der beiden Sensorkonstellationen erkannt wurde, wird aus dem jeweiligen Dateisystem-Ordner eine Liste der beinhalteten Dateien eingelesen, in denen entsprechende *FitnessTrailActivityLists* gespeichert wurden. Die eingelesenen Dateiobjekte werden in einer Drop-down Komponente (Android Spinner Komponente) zur Auswahl gestellt, mit deren Hilfe nun die *FitnessTrailActivityLists* ausgewählt werden kann. Diese kann nun bearbeitet oder zur Durchführung eines Workouts genutzt werden. Für den Fall, dass eine neue Aktivitätsliste angelegt wer-

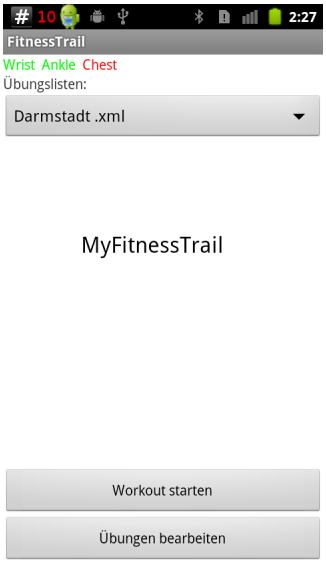

Abbildung 4.6: Startbildschirm

den soll, beinhaltet die Drop-down Komponente einen zusätzlichen Eintrag “Neu”, der nur das Erstellen und Bearbeiten einer neuen Aktivitätsliste ermöglicht.

Mit Hilfe der Buttons “Übungen bearbeiten” und “Workout starten” ist dem Anwender die Entscheidung überlassen, ob eine Aktivitätsliste bearbeitet oder ein Workout auf dem Trimm-dich-Pfad durchgeführt werden soll.

4.2.7.2 Aktivitätsaufnahmebildschirm

Das Training und das Speichern von Aktivitäten findet mit Hilfe der *EditExercisesActivity*, wie in Abbildung 4.7 zu sehen, statt.

Abbildung 4.7: Aufnahmebildschirm

Aufgrund der in der *HomeActivity* Ansicht getroffenen Auswahl wird ein *FitnessTrailActivityLists* Objekt aus dem Speicher geladen oder ein neues Objekt erzeugt. Ist das Objekt neu, muss für die Speicherung ein eindeutiger Name vergeben werden.

Das entsprechende Drop-down Menü beinhaltet alle Namen der Aktivitäten, die bereits in der *FitnessTrailActivityLists* enthalten sind, sowie einen Eintrag für das Erstellen einer neuen zu trainierenden Aktivität. Für eine neue Aktivität muss ein nicht bereits vorhandener Name vergeben werden. Des Weiteren lässt sich auf dem Bildschirm der für die spätere Wiederholungszählung zu nutzende Sensor auswählen und festlegen ob es sich um eine zählbare oder nicht zählbare Aktivität handelt. Nicht zählbare Aktivitäten sind Aktivitäten, wie 'Gehen' oder 'Laufen', für die während des Durchführens des Workouts nicht die Anzahl an Wiederholungen, sondern die Dauer der Durchführung relevant ist.

Unter Countdownzeit lässt sich die Länge des Countdowns festlegen, der vor der Aufnahme der Aktivität über den Lautsprecher herunter gezählt ausgegeben wird. Ebenso lässt sich unter Aufnahmezeit die Zeitspanne festlegen, während der später Sensordaten gesammelt werden. Die gesammelten Daten werden zur Bestimmung des Gauß-Modells benötigt, welches die Aktivität mathematisch repräsentiert.

Der Button "Übung einfügen/bearbeiten" startet die eigentliche Aufnahme bzw. das Training der Aktivität. Der Countdown wird herunter gezählt, bevor die Aufnahme beginnt. Nach dem Countdown folgt ein Tonsignal, welches signalisiert, dass die Aufnahme startet. Nun werden die Sensordaten für die Berechnung gespeichert. Nach Ablauf der Aufnahmezeit, ertönt ein weiterer Signalton, der das Ende der Aufnahme signalisiert. Nun erfolgt die Berechnung bzw. das Training der Aktivität wie in Abschnitt 4.2.2 beschrieben. Nach erfolgreicher Berechnung wird die Aktivität zur Aktivitätsliste hinzugefügt und diese als Datei gespeichert.

Der Button "Lösche aktuelle Übung" löscht die im Drop-down Menü ausgewählte Aktivität aus der Aktivitätsliste und überschreibt die alte Datei im Speicher. Der Button "Lösche Übungsliste" löscht die Datei aus dem Speicher. Nach erfolgter Fertigstellung aller Aktionen erfolgt die Rückkehr zum Startbildschirm.

4.2.7.3 Workoutbildschirm

Während der Durchführung eines Workouts, in diesem Fall also die sportliche Betätigung auf dem Trimmdich-Pfad, werden die zuvor antrainierten und in Aktivitätslisten gespeicherten Aktivitäten erkannt. Für die korrekte Aktivitätserkennung, muss zuvor eine entsprechend trainierte Aktivitätsliste ausgewählt worden sein. Für zählbare Übungen, wie Liegestütz oder Kniebeuge, werden die Anzahl der durchgeführter Wiederholungen, für nicht zählbare Übungen, wie Laufen und Gehen, die Dauer der Durchführung ermittelt. Des Weiteren erfolgt eine akustische Motivation während des Workouts.

Abbildung 4.8: Workoutbildschirm

Hierfür ist die *WorkoutActivity*, zu sehen in Abbildung 4.8, zuständig. Während eines Workouts werden die, über MHH sekündlich eingehenden, Sensor-Events der angebundenen Beschleunigungssensoren dem Aktivitätserkennungsalgorithmus zugeleitet.

Es werden die aktuelle Übung und die Zahl der absolvierten Wiederholungen dargestellt. Diese Informationen werden dem Motivations- und Datenbankmodul kontinuierlich weitergeleitet. Somit stehen dem Modul die notwendigen Daten zur Verfügung, um, aufbauend auf vorangegangenen Workouts, den Nutzer zu motivieren. Ebenso stellt der Workoutbildschirm eine Liste der bisher im aktuellen Workout beendeten Übungen mit der Dauer oder Wiederholungszahl zur Verfügung. Diese Liste wird mit Hilfe der *ExerciseSet-Events* gefüllt, die nach Beendigung eines Übungssatzes vom Motivations- und Datenbankmodul erzeugt werden. Dieses Modul wird im folgenden Kapitel beschrieben.

5 Motivations- und Datenbankmodul

Die entwickelte Android-Applikation bindet ein eigens entwickeltes Motivations- und Datenbankmodul an, das in diesem Kapitel vorgestellt wird. Dieses Modul kann ebenfalls von weiteren und zukünftigen Anwendungen eingebunden werden. Um die Daten, die im Laufe einer Übung entstehen, für spätere Anwendungen nutzen zu können, ist die Hauptaufgabe des Moduls die Abspeicherung aller abgeschlossenen Übungen. Diese Daten werden im Rahmen einer relationalen Datenbank auf dem Gerät gespeichert. Zusätzlich sorgt eine akustische Lautsprecherausgabe für Motivation des Trainierenden. Hierbei wird zum Beispiel die Anzahl der geleisteten Wiederholungen über eine Text-to-Speech API ausgegeben.

5.1 Datenbank

Dem Modul liegt eine Datenbank zugrunde, in der die Workouts und die darin absolvierten Übungen mit Anzahl der Wiederholungen oder der Durchführungsdauer gespeichert werden. In diesem Fall wird eine leichtgewichtige Datenbank benötigt, die für das Android Betriebssystem geeignet ist.

Aufgrund der Tatsache, dass dies bereits von Android unterstützt wird fiel die Wahl auf *SQLite*[1], eine SQL-Datenbank Engine [1]. Hierbei kommt *SQLite* ohne einen eigenständigen Server aus, der von der Programmbibliothek gestellt wird. Durch das integrierte Transaktionsmanagement haben auch Software- oder Geräteabstürze keinen Einfluss auf die Integrität der Datenbank.

5.1.1 Tabellen der Datenbank

Um die Daten aus den Workouts auf eine relationale Datenbankstruktur abbilden zu können, wurde ein Datenbanklayout mit zwei Tabellen entwickelt. Diese bestehen aus einer *Workout*-Tabelle und einer Exercise-Tabelle. Hierbei sind die Daten der *Exercise*-Tabelle relational mit denen der Workout-Tabelle verknüpft. Somit können mehrere Exercises einfach einem Workout zugeordnet werden. Im Folgenden wird die Datenbankstruktur der beiden Tabellen vorgestellt.

ID	Activity List	Date
1	Darmstadt3	27.12.2012
2	Rossdorf2	29.12.2012
...	...	...

Tabelle 5.1: *WorkoutTbl* Datenbanktabelle

In der *Workout*-Tabelle wird die dem Workout zugrunde liegende Aktivität gespeichert. Jedes Workout besitzt eine eindeutige *ID*, einen Bezeichner im Textformat (*Activity List*) und ein Datum (*Date*), das den Zeitpunkt der Erstellung kennzeichnet. Tabelle 5.1 zeigt eine solche, mit Beispieldaten gefüllte, Tabelle.

In der *Exercise*-Tabelle werden alle durchgeführten Übungen gespeichert. Dabei werden der Name der Übung, die Anzahl der Durchführungen bzw. die Dauer der Durchführung sowie eine Einheit zur Unterscheidung von Anzahl und Dauer gespeichert. Diese Übung wird mittels der *Workout ID* einem bestimmten Workout zugeordnet. Tabelle 5.2 verdeutlicht dieses Schema.

[1] http://www.sqlite.org/

ID	Workout ID	ExerciseName	Unit	Quantity
1	1	Klimmzüge	Wdh	12
2	1	Klimmzüge	Wdh	10
3	1	Laufen	Sec	360
4	2	Klimmzüge	Wdh	12
...	...	...	...	...

Tabelle 5.2: Datenbanktabelle: ExerciseTbl

Abbildung 5.1 zeigt ein Entity-Relationship Diagramm der zugrundeliegenden Datenbankstruktur. Die Relation zwischen *Exercises* und *Workouts* wird deutlich.

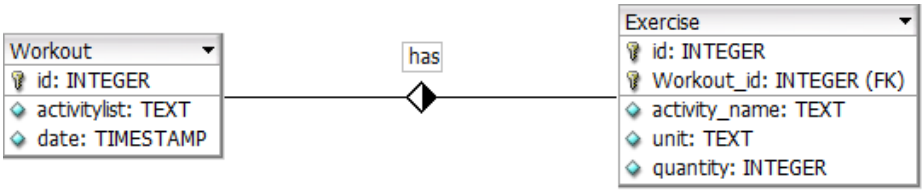

Abbildung 5.1: Entity-Relationship Diagramm für die Datenbank

5.1.2 Funktionen der Datenbank

Um die Datenbankverwaltung übersichtlicher zu gestalten, werden, entsprechend der vorgestellten Tabellen, zwei Objektklassen benutzt, deren Instanzen jeweils einen Eintrag der jeweiligen Datenbanktabellen abbilden. Somit werden die Datenbank-Daten auf eine Objektstruktur gemapped, um die Verwaltung der Daten innerhalb der Applikation wesentlich zu vereinfachen. Die entwickelten Klassen sind **'DBEntryWorkout'** (Abbildung 5.2) und **'DBEntryExercise'** (Abbildung 5.3).

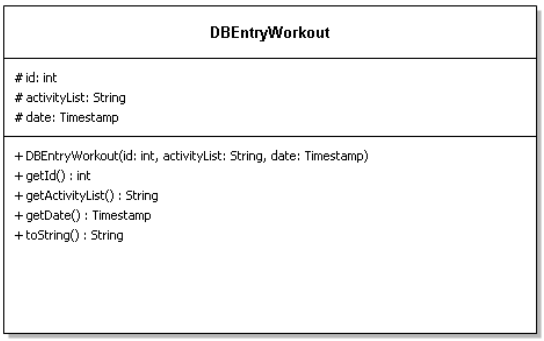

Abbildung 5.2: Klassendiagramm DBEntryWorkout

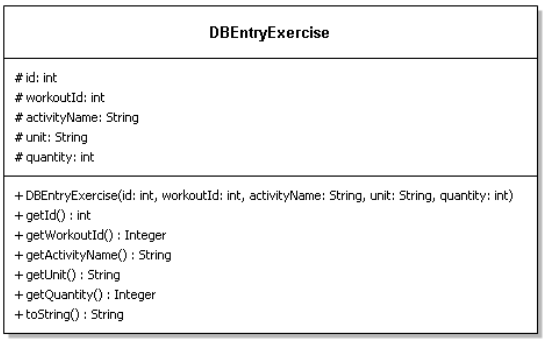

Abbildung 5.3: Klassendiagramm DBEntryExercise

Die im Folgenden beschriebenen Funktionen der Datenbank nutzen diese Klassen zur Ein- und Ausgabe. Folgende Funktionen wurden im Rahmen der Arbeit zur Verwaltung der Trainingsdaten entwickelt:

- ***insertWorkout***
- ***insertExercise***

- ***getBestExerciseQuantityFromWorkout***
- ***getExercisesFromLastWorkout***
- ***getAllWorkouts***
- ***getAllExercises***
- ***dropTable***

Mittels ***insertWorkout*** und ***insertExercise***, denen jeweils Instanzen der oben beschriebenen Klassen übergeben werden, werden neue Einträge zur Datenbank hinzugefügt. Die Funktion ***insertWorkout*** gibt dabei auch noch die für den neuen Eintrag verwendete ID zurück. Diese wird für die Einträge der Übungen bzw. der Verknüpfung der Übung zum Workout benötigt.

Indem ***getExercisesFromLastWorkout*** mit einem String *ActityListname* als Parameter aufgerufen wird, wird das letzte Workout in der Datenbank gesucht, welches den Namen als Eintrag enthält. Zurückgegeben wird außerdem eine Liste von Objekten des Typs 'DBEntryExercise', die zu diesem Workout gehören.

Die Funktion ***getBestExerciseQuantityFromWorkout*** erhält als Eingabe einen String-Paramater *ActivityListname* und einen String-Parameter *ExerciseName*. Sie gibt die Übung mit dem besten Ergebnis zu einem Workout und eines Übungstyps zurück. Die beste Übung ist hierbei diejenige mit dem höchsten Wert für *quantity*, also das bisher beste gespeicherte Zählergebnis für die jeweilige Übung.

Ein Aufruf von ***getAllWorkouts*** und ***getAllExercises*** liefert jeweils eine Liste von Instanzen von 'DBEntryWorkout', bzw. 'DBEntryExercise', zurück. Diese beinhalten im Folgenden alle Einträge der jeweiligen Tabelle in der Datenbank.

Die Funktion ***dropTable*** löscht den gesamten Inhalt beider Tabellen in der Datenbank.

5.2 Motivation

Zur Motivation während des Trainings wird mit der Text-to-Speech API von Android ein Audiofeedback erzeugt. Zu Beginn einer Übung wird hierzu die im letzten Workout für den aktuellen Satz erzielte Wiederholungsanzahl und die bisher beste Wiederholungsanzahl für die Übung sprachlich ausgegeben. Des Weiteren wird, wenn sich die aktuelle Zählung der im letzten Workout erreichten Wiederholungszahl annähert, ein Countdown gestartet. Dieser Countdown repräsentiert die Differenz zwischen den beiden Werten (Standard Threshold-Differenz = 3). Dies geschieht so lange, bis diese Differenz null ist, und genauso viele Wiederholungen geschafft wurden wie beim letzten Workout. Um dies zu ermöglichen, wird das letzte Workout beim Erstellen des Moduls als Instanz der Workout-Klasse, wie in Abbildung 5.4 dargestellt, übergeben.

Für die Ausgabe des besten Ergebnisses muss die Funktion *setBestExercise* für jeden Übungswechsel erneut mit dem aktuellen Wert aufgerufen werden. Hiermit ist gewährleistet, dass die beste Übung im aktuellen Durchlauf genannt wird, auch wenn bei der vorangegangenen Durchführung einer Übung das beste Ergebnis erzielt wurde. Entspricht der Übungsname der aktuell durchgeführten Übung, wird die Sprachausgabe hierfür ausgelassen, wenn diese nicht als beste Übung gespeichert wurde.

Dieses Verhalten ändert sich beim Fehlen einer Übung oder des jeweiligen Satzes im vorherigen Workout. In diesem Fall wird für das letzte Mal eine '0' ausgegeben.

Die aktuelle Übung mit der Wiederholungszahl und Einheit wird über die Funktion *incomingExercise* an das Motivationsmodul übergeben. Dann wird entschieden, ob die eingehende Übung einen neuen Übungsnamen hat oder ob die letzte Zählung fortgesetzt wird. Ist es eine neue Übung, wird für die

Alte ein Satzzähler erhöht (für den Vergleich mit dem letzten Workout). Nun kann der Vergleich mit der Anzahl der im letzten Workout absolvierten Wiederholungen erfolgen. Ist die Differenz geringer oder gleich der Threshold-Differenz, wird sie über die Text-to-Speech Ausgabe wiedergegeben und bis '0' herunter gezählt.

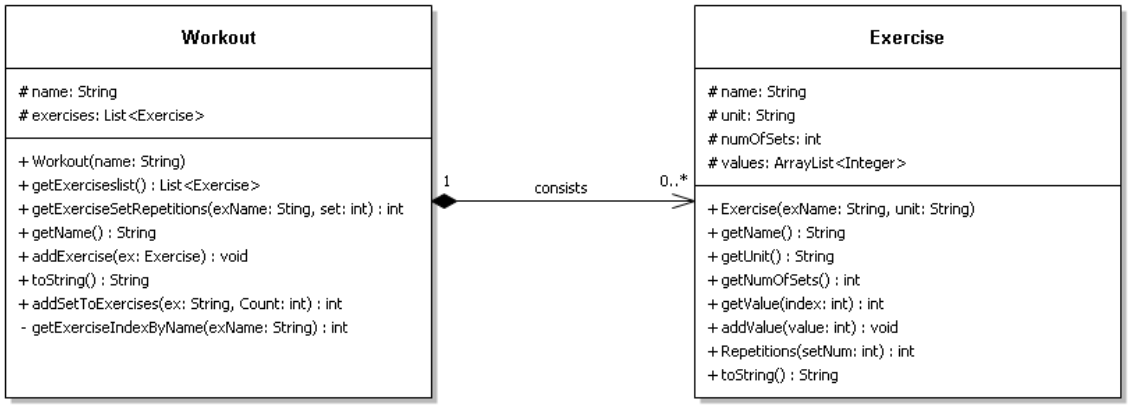

Abbildung 5.4: Klassendiagramm von Workout und Exercise

Um auch bei kurzen Übungssätzen, also Sätzen mit weniger als fünf Wiederholungen, zu gewährleisten, dass das letzte und beste Ergebnis zu Beginn der Übung genannt werden kann, gibt es in diesem Fall keine Countdown Sprachausgabe. Wenn dies der Fall wäre, könnte der Countdown beginnen, bevor die Startausgabe beendet wäre.

Aktuell wird die Motivation nur für Übungen mit der Einheit 'Wdh', also für zählbare Übungen, durchgeführt. Dies ist also nicht praktikabel für Übungen die über die Zeit gemessen werden, also für Übungen mit der Einheit 'Sec'.

5.3 Verknüpfung von Datenbank und Motivation

Die Verknüpfung von Datenbank und Motivation wird von der Klasse *MotivationDBHandle* durchgeführt. Hier wird die Datenbank beim Erstellen einer Instanz der Klasse nach der jeweils aktuellen Aktivitätsliste des letzten Workouts durchsucht. Somit wird dann eine, für die Motivation nutzbare, Instanz der Workout-Klasse erstellt, die die Übungsdaten des Workouts beinhaltet. Mit dieser Instanz kann nun die Motivation gestartet werden und ein Datenbankeintrag für das aktuelle Workout erstellt werden.

Über die Funktion ***incomingExercise*** werden die Übungen mit dem aktuellen Zählerwerten, und der Information ob die Übung zählbar ist, an das Modul übergeben. Anhand der Zählbarkeit wird entschieden welche Einheit für die der Zählerwert einer Übung gilt - 'Sec' für nicht zählbare Übungen und 'Wdh' für zählbare Übungen.

Anhand des Übungsnamens und der Zahl der Wiederholungen fällt die Entscheidung, ob es sich um eine neue Übung, bzw. Übungsdurchführung, handelt, oder ob es eine Wiederholung einer alten Übung ist. Wird eine neue Übung bzw. ein neuer Satz der aktuellen Übung erkannt, werden die alten Daten in die Datenbank geschrieben. Anschließend wird das im folgenden beschriebene *ExerciseSetEvent* erzeugt und über einen Kanal von MyHealthHub gesendet. Schließlich wird die höchste Quantity für die neue Übung aus der Datenbank abgefragt und an das Motivationsmodul für die Ansage des bisher besten Ergebnisses weitergeben. Für alle eingehenden Übungen werden Übungsname, Zähler und Einheit an das Motivationsmodul zur Erstellung der Audioausgabe weitergereicht.

5.4 ExerciseSetEvent

Das ExerciseSetEvent ist ein von der Klasse *MotivationDBHandle* erzeugtes Event, welches an MyHealthHub gesendet wird. Es beinhaltet, neben den für den Hub relevanten Informationen, den Namen der aktuell abgeschlossenen Übung, deren Einheit und die gezählten Wiederholungen.

Das Event wird in der Klasse erzeugt, wann immer ein Satz beendet und in die Datenbank geschrieben wird. An diesen Daten interessierte Anwendungen können sich bei MyHealthHub über den genutzten Kanal anmelden, somit das Event empfangen und die beinhalteten Daten nutzen.

5.5 Schnittstelle

Um das Motivations- und Datenbankmodul auch für andere Anwendungen nutzbar zu machen, ist es erforderlich eine Schnittstelle zu schaffen. Im folgenden wird die Anwendung des entwickelten Moduls beschrieben.

Für die Erzeugung einer Instanz von *MotivationDBHandle* bestehen zwei mögliche Konstruktoraufrufe:

MotivationDBHandle(Context context, String activityListname)

oder

MotivationDBHandle(Context context, String activityListname, int countdownLength)

Im ersten Fall wird die Datenbank, falls diese bereits vorhanden ist, nach dem letzten Workout mit 'activityListname' als ActivityList durchsucht. Auf Basis der Datenbankeinträge für dieses Workout wird nun eine Instanz der Klasse *Workout* erstellt und mit dieser das Motivationsmodul gestartet. Dann wird ein neuer Workout-Eintrag in der Datenbank angelegt. Über den zweiten Konstruktor lässt sich zusätzlich die Länge der Countdowns für die Motivation bestimmen.

Aktuelle Zählergebisse für Übungen werden über die Funktion

incomingExercise(String exerciseName, int exerciseCount, boolean isCountable)

gemeldet. Hier wird anhand der Zählbarkeit der Übung festgelegt, welche Einheit für die Übung gilt ('Wdh' oder 'Sec') und schließlich der Aufruf der Motivation und die Datenbankzugriffe ausgeführt.

Zum Abschluss muss die Funktion

close()

aufgerufen werden, damit der Text-to-Speech-Listener vom Motivationsmodul abgemeldet und auch der letzte Übungseintrag in die Datenbank geschrieben wird.

6 Abschluss

6.1 Zusammenfassung

Im Fokus dieser Arbeit lag die Konzeptionierung und Implementierung eines Aktivitätserkennungssystems für Trimm-dich-Pfade. Hierzu wurde eine Smartphone-Applikation entwickelt, die, aufbauend auf einer bestehenden drahtlosen Bluetooth-Lösung zur Anbindung mehrerer Beschleunigungssensoren, die Erkennung von Fitnessübungen, wie sie auf Trimm-dich-Pfaden durchgeführt werden, ermöglicht. Zudem erlaubt die Anwendung das Zählen der durchgeführten Wiederholungen einer Übung. Um die Erkennung von Wiederholungen während einer Trimm-dich-Pfad Übung zu gewährleisten, ist es möglich die Sensoren in zwei unterschiedlichen Konfigurationen am Körper anzubringen. Diese setzen entweder zwei oder drei Beschleunigungsensoren voraus. Die Beschleunigungssensoren basieren auf der verwendeten HedgeHog-Architektur und wurden mit einer applikationsspezifischen Implementierung eines Peak-Detection-Algorithmus erweitert. Dieser Algorithmus hat den großen Vorteil, dass die Erkennung von Wiederholungen einer durchgeführten Übung direkt auf dem Sensorknoten stattfinden kann. Ebenso wurde das zur Aktivitätserkennung notwendige Verfahren weiterentwickelt, das in seiner ursprünglichen Version anhand der Sensorbeschleunigungsdaten Mittelwert und Varianz für ein Sekunden lange Werteintervalle berechnet. Es werden nun Beschleunigungsdaten über längere Zeiträume betrachtet. Dadurch, dass nun Werteintervalle über vier Sekunden als Berechnungsgrundlage dienen, konnte die Erkennungsgenauigkeit der Aktivitäten auf dem Trimm-dich-Pfad erhöht werden.

Mit *MyFitnessTrail* wurde eine Smartphone-Applikation entwickelt, die die empfangenen Beschleunigungs- und Peak-Detection-Daten auf Basis der MyHealthHub-Middleware verarbeitet. In dieser Applikation lassen sich mehrere Trimm-dich-Pfad Aktivitäten verwalten. Darunter fallen zum Beispiel die Beschreibung, die Auswahl des für die Übung relevanten Sensors sowie die Aufnahmezeit. Ein zentraler Bestandteil der Applikation ist die Möglichkeit einen Workout zu starten und daraufhin Motivation bei der Durchführung der Übung zu erhalten. Diese erfolgt über einen Countdown, der über eine Text-to-Speech Ausgabe synthetisiert wird und, im Bezug auf die Anzahl an durchgeführten Wiederholungen, ausgegeben wird. Die Daten der Applikation werden in einer SQLite Datenbank gespeichert und sind nach einem Neustart des Programms oder einem Systemabsturz weiterhin verfügbar.

6.2 Ausblick

Während dieser Arbeit konnten viele Themen für zukünftige Arbeiten identifiziert werden. Aufgrund von Kompatibilitätsproblemen zwischen der bestehenden Sensorhardware und der Sensorsoftware konnte die entwickelte Peak-Detection auf den Sensoren leider nicht zum Einsatz kommen. Hier sollte in absehbarer Zeit ein Bugfix vorliegen, sodass die in dieser Arbeit entwickelte Sensorprogrammierung auch zum Einsatz kommen kann.

Weiterhin ist es von Interesse die Anzahl der einstellbaren Parameter pro Übung zu erweitern, sodass ein höherer Automatisierungsgrad erreicht wird, aber auch weitere Anwendungsdomänen erschlossen werden können. Somit könnte zum Beispiel die GPS-Position dafür genutzt werden, um die jeweilige Übung auf dem Trimm-dich-Pfad vorzuselektieren. Dies ist insbesondere im Außenbereich sinnvoll, da die Genauigkeit der GPS-Position im Allgemeinen ausreicht um den Aufenthalt an einer Übungsstation eindeutig zu bestimmen. In Indoor-Anwendungsfällen, wie Fitnessstudios, könnte eine auf Signalstärken basierende Lokalisierungstechnologie, beispielsweise anhand von Wireless LAN-Signalen, zum Einsatz kommen.

Der Einsatz des Motivationsmoduls in anderen Anwendungsbereichen stellt ebenfalls einen interessanten Ausblick auf zukünftige Arbeiten dar. Hierbei könnte das Motivationsmodul für Wiederholungsübungen in Fitnessstudios oder im Heimbereich genutzt werden. Es könnte ebenfalls sinnvoll sein die Text-to-Speech Ausgabe mit weiteren motivierenden Features, wie bspw. das Abspielen von MP3-Dateien als Belohnung zu erweitern. Außerdem kann auch die Durchführungszeit, neben der Anzahl der durchgeführten Wiederholungen, in die Motivation mit einfließen.

Zusammenfassend kann gesagt werden, dass sich der Einsatz von Smartphones in Kombination mit drahtlosen Beschleunigungssensoren sehr für den Anwendungsfall von Aktivitätserkennung bei Trimm-dich-Pfaden eignet. Hierzu muss in jedem Fall noch eine Nutzerevaluation erfolgen, die nach Fertigstellung der Sensorumstellung Sinn macht.

A Quellcode

Listing A.1: Quellcode Sensor Mittelwert- und Varianzberechnung sowie Peak-Detection

```
%\begin{lstlisting}[caption={Poll interface of \acs{VAPI}},label=code:vapi_poll]
/*****************************************************************************
 * Global Variables and Constants
 *****************************************************************************/
// Variables for Mean and Variance Calculation
INT8 array_pos;
INT16 sum_x[4], sum_y[4], sum_z[4];
INT32 sqsum_x[4], sqsum_y[4], sqsum_z[4];

// Variables for Peak Detection
#pragma udata udata_x_array
INT8 x[100];
#pragma udata udata_y_array
INT8 y[100];
#pragma udata udata_z_array
INT8 z[100];
BOOL lookformax_x, lookformax_y, lookformax_z;
INT8 max_val_x, max_val_y, max_val_z;
INT8 max_time_x, max_time_y, max_time_z;
INT8 min_val_x, min_val_y, min_val_z;
INT8 min_time_x, min_time_y, min_time_z;

// Constants for Mean and Variance Calculation and Peak Detection
const INT8 init_max = -128;
const INT8 init_min = 127;
const INT8 sendwindow_ms = 100;
const INT8 recordwindow_s = 4;
const INT8 delta_time_max_min = 50;


/*****************************************************************************
 * Function:        init_mean_var_peak(void)
 * Overview:        Global Variables for Mean and Variance Calculation
 *                  and for Peak Detection are set initially here.
 *****************************************************************************/
void init_mean_var_peak(void) {
  INT8 i;

  for (i= 0; i < 100; i++){
    x[i] = 0;
    y[i] = 0;
    z[i] = 0;
  }

  for (i= 0; i < 4; i++){
     sum_x[i] = 0;
     sum_y[i] = 0;
     sum_z[i] = 0;
     sqsum_x[i] = 0;
     sqsum_y[i] = 0;
     sqsum_z[i] = 0;
  }
```

```
53
54    lookformax_x = TRUE;
55    lookformax_y = TRUE;
56    lookformax_z = TRUE;
57    max_val_x = init_max;
58    max_val_y = init_max;
59    max_val_z = init_max;
60    max_time_x = 0;
61    max_time_y = 0;
62    max_time_z = 0;
63    min_val_x = init_min;
64    min_val_y = init_min;
65    min_val_z = init_min;
66    min_time_x = 0;
67    min_time_y = 0;
68    min_time_z = 0;
69
70    array_pos = 0;
71  }
72
73
74  /*****************************************************************************
75   * Function:        calc_mean_var_peak(void)
76   * Overview:        Calcutes Mean and Variance and performs Peak Detection
77   *****************************************************************************/
78  void calc_mean_var_peak(void) {
79    // local Variables for Calculation of Mean, Variance and for Peak Detection
80    INT8 i;
81
82    // Mean and Variance
83    INT32 t_sum_x, t_sum_y, t_sum_z;
84    INT32 t_sqsum_x, t_sqsum_y, t_sqsum_z;
85    float var_x, var_y, var_z;
86    INT8  mean_x, mean_y, mean_z;
87
88    // Peak Detection
89    INT8 max_x, min_x, max_y, min_y, max_z, min_z;
90    INT8 delta_val_x, delta_val_y, delta_val_z;
91    INT8 cur_val;
92    INT8 peak_x, peak_y, peak_z;
93
94    sum_x[array_pos] = 0;
95    sum_y[array_pos] = 0;
96    sum_z[array_pos] = 0;
97    sqsum_x[array_pos] = 0;
98    sqsum_y[array_pos] = 0;
99    sqsum_z[array_pos] = 0;
100   t_sum_x = 0;
101   t_sum_y = 0;
102   t_sum_z = 0;
103   t_sqsum_x = 0;
104   t_sqsum_y = 0;
105   t_sqsum_z = 0;
106   peak_x = 0;
107   peak_y = 0;
108   peak_z = 0;
109
110   max_x = init_max;
111   min_x = init_min;
112   max_y = init_max;
```

```
113   min_y = init_min;
114   max_z = init_max;
115   min_z = init_min;
116
117   for (i = 0; i < sendwindow_ms; i++) {
118     acc_getxyz(&accval);
119
120     x[i]                =   accval.x;
121     sum_x[array_pos]    = sum_x[array_pos] + x[i];
122     sqsum_x[array_pos]  = sqsum_x[array_pos] + ((INT32)x[i] * (INT32)x[i]);
123
124     y[i]                =   accval.y;
125     sum_y[array_pos]    = sum_y[array_pos] + y[i];
126     sqsum_y[array_pos]  = sqsum_y[array_pos] + ((INT32)y[i] * (INT32)y[i]);
127
128     z[i]                =   accval.z;
129     sum_z[array_pos]    = sum_z[array_pos] + z[i];
130     sqsum_z[array_pos]  = sqsum_z[array_pos] + ((INT32)z[i] * (INT32)z[i]);
131
132     // to calculate MAX and MIN for delta_vals in Peak Detection
133     if(x[i] > max_x)max_x = x[i];
134     if(x[i] < min_x)min_x = x[i];
135     if(y[i] > max_y)max_y = y[i];
136     if(y[i] < min_y)min_y = y[i];
137     if(z[i] > max_z)max_z = z[i];
138     if(z[i] < min_z)min_z = z[i];
139
140     Delayms(9);
141   }
142
143   // Mean and Variance Calculation
144   for (i= 0; i < recordwindow_s ; i++) {
145     t_sum_x += sum_x[i];
146     t_sum_y += sum_y[i];
147     t_sum_z += sum_z[i];
148     t_sqsum_x += sqsum_x[i];
149     t_sqsum_y += sqsum_y[i];
150     t_sqsum_z += sqsum_z[i];
151   }
152
153   mean_x = t_sum_x /(recordwindow_s*sendwindow_ms);
154   mean_y = t_sum_y /(recordwindow_s*sendwindow_ms);
155   mean_z = t_sum_z /(recordwindow_s*sendwindow_ms);
156
157   var_x = (float)(t_sqsum_x - t_sum_x*t_sum_x/(recordwindow_s*sendwindow_ms))/(
          recordwindow_s*sendwindow_ms);
158   var_y = (float)(t_sqsum_y - t_sum_y*t_sum_y/(recordwindow_s*sendwindow_ms))/(
          recordwindow_s*sendwindow_ms);
159   var_z = (float)(t_sqsum_z - t_sum_z*t_sum_z/(recordwindow_s*sendwindow_ms))/(
          recordwindow_s*sendwindow_ms);
160
161   // Peak Detection Begin
162   delta_val_x = (INT8) ((INT16)max_x - (INT16)min_x )/3;
163   delta_val_y = (INT8) ((INT16)max_y - (INT16)min_y )/3;
164   delta_val_z = (INT8) ((INT16)max_z - (INT16)min_z )/3;
165
166   max_x = init_max;
167   min_x = init_min;
168   max_y = init_max;
169   min_y = init_min;
```

```
  max_z = init_max;
  min_z = init_min;

  // x-Direction
  for (i = 0; i < sendwindow_ms; i++) {
    cur_val = x[i];

    if(cur_val > max_val_x){
      max_val_x = cur_val;
      min_time_x = i;
    }

    if(cur_val < min_val_x){
      min_val_x = cur_val;
      min_time_x = i;
    }


    if (lookformax_x == TRUE) {
      if((cur_val < (max_val_x - delta_val_x)) && ((i - max_time_x)> delta_time_max_min))
          {
        min_val_x = cur_val;
        min_time_x = i;
        lookformax_x = FALSE;
      }
    } else {
      if ((cur_val < (min_val_x + delta_val_x)) && ((i - min_time_x)> delta_time_max_min))
           {
        max_x = max_val_x;
        min_x = min_val_x;
        max_val_x = cur_val;
        max_time_x = i;
        peak_x = 1;
        lookformax_x = TRUE;
      }
    }
  }

  // y-Direction
  for (i = 0; i < sendwindow_ms; i++) {
    cur_val = y[i];

    if (cur_val > max_val_y) {
      max_val_y = cur_val;
      min_time_y = i;
    }

    if (cur_val < min_val_y) {
      min_val_y = cur_val;
      min_time_y = i;
    }

    if (lookformax_y == TRUE) {
      if((cur_val < (max_val_y - delta_val_y)) && ((i - max_time_y)> delta_time_max_min))
          {
        min_val_y = cur_val;
        min_time_y = i;
        lookformax_y = FALSE;
      }
    } else {
```

```
227       if((cur_val < (min_val_y + delta_val_y)) && ((i - min_time_y)> delta_time_max_min))
              {
228         max_y = max_val_y;
229         min_y = min_val_y;
230         max_val_y = cur_val;
231         max_time_y = i;
232         peak_y = 1;
233         lookformax_y = TRUE;
234       }
235     }
236   }
237
238   // z-Direction
239   for (i = 0; i < sendwindow_ms; i++) {
240     cur_val = z[i];
241
242     if(cur_val > max_val_z){
243       max_val_z = cur_val;
244       min_time_z = i;
245     }
246
247     if(cur_val <min_val_z){
248       min_val_z = cur_val;
249       min_time_z = i;
250     }
251
252     if (lookformax_z == TRUE) {
253       if((cur_val < (max_val_z - delta_val_z)) && ((i - max_time_z)> delta_time_max_min))
              {
254         min_val_z = cur_val;
255         min_time_z = i;
256         lookformax_z = FALSE;
257       }
258     } else {
259       if((cur_val < (min_val_z + delta_val_z)) && ((i - min_time_z)> delta_time_max_min))
              {
260         max_z = max_val_z;
261         min_z = min_val_z;
262         max_val_z = cur_val;
263         max_time_z = i;
264         peak_z = 1;
265         lookformax_z = TRUE;
266       }
267     }
268   }
269
270   max_time_x -= sendwindow_ms;
271   min_time_x -= sendwindow_ms;
272   max_time_y -= sendwindow_ms;
273   min_time_y -= sendwindow_ms;
274   max_time_z -= sendwindow_ms;
275   min_time_z -= sendwindow_ms;
276
277   // Peak Detection End
278   array_pos = (array_pos + 1) % recordwindow_s;
279   cdc_print_mean_var_peak(mean_x,mean_y, mean_z, (INT16)var_x, (INT16)var_y, (INT16)var_z,
          peak_x, peak_y, peak_z );
280 }
```

Abbildungsverzeichnis

Literaturverzeichnis

[1] Arno Becker and Marcus Pant. *Android 2 Grundlagen und Programmierung*. dpunkt.verlag GmbH, Heidelberg, 2010.

[2] Keng-hao Chang, MikeY. Chen, and John Canny. Tracking Free-Weight Exercises. In *UbiComp 2007: Ubiquitous Computing*, volume 4717 of *Lecture Notes in Computer Science*, pages 19–37. Springer Berlin Heidelberg, 2007.

[3] Thomas Pignede. Development of an Activity Recognition System for Fitness Trails. Bachelor thesis, TU Darmstadt, July 2012.

[4] Christian Seeger, Alejandro Buchmann, and Kristof Van Laerhoven. myHealthAssistant: A Phone-based Body Sensor Network that Captures the Wearerś Exercises throughout the Day. In *The 6th International Conference on Body Area Networks (BodyNets)*, November 2011.

[5] Christian Seeger, Alejandro Buchmann, and Kristof Van Laerhoven. Poster Abstract: Adaptive Gym Exercise Counting for myHealthAssistant. In *The 6th International Conference on Body Area Networks (BodyNets)*, November 2011.

[6] Christian Seeger, Alejandro Buchmann, and Kristof Van Laerhoven. An Event-based BSN Middleware that supports Seamless Switching between Sensor Configurations. In *ACM SIGHIT International Health Informatics Symposium (IHI 2012)*, January 2012.