SPEEDOMETER

Verbesserung der GPS-Messung mittels zusätzlicher Sensoren

im Auftrag der runtastic GmbH

Diplomarbeit für die Reife- & Diplomprüfung im Fachgebiet
Projektentwicklung
an der Höheren Technischen Bundeslehranstalt Perg
Abteilung: Höhere Lehranstalt für Informatik

Autor: **Stefan Mittermayr**
Prüfer: **Prof. Dipl.-Ing. Richard Kainerstorfer**
Abgabedatum: 13. Mai 2015

Inhaltsverzeichnis

Eidesstattliche Erklärung

Die unterfertigten Kandidaten / Kandidatinnen haben gemäß § 34 (3) SchUG in Verbindung mit § 22 (1) Zi. 3 lit. b der Verordnung über die abschließenden Prüfungen in den berufsbildenden mittleren und höheren Schulen, BGBl. II Nr. 70 vom 24.02.2000 (Prüfungsordnung BMHS), die Ausarbeitung einer Diplomarbeit mit der umseitig angeführten Aufgabenstellung gewählt.

Die Kandidaten / Kandidatinnen nehmen zur Kenntnis, dass die Diplomarbeit in eigenständiger Weise und außerhalb des Unterrichtes zu bearbeiten und anzufertigen ist, wobei Ergebnisse des Unterrichtes mit einbezogen werden können.

Die Abgabe der Diplomarbeit hat bis spätestens **13.05.2015** beim zuständigen Betreuer / der zuständigen Betreuerin zu erfolgen.

Die Kandidaten / Kandidatinnen nehmen weiters zur Kenntnis, dass gemäß § 9 (6) der Prüfungsordnung BMHS nur der Schulleiter bis spätestens Ende des vorletzten Semesters den Abbruch einer Diplomarbeit anordnen kann, wenn diese aus nicht beim Prüfungskandidaten (bei den Prüfungskandidaten) gelegenen Gründen nicht fertiggestellt werden kann.

Kandidaten / Kandidatinnen inkl. Unterschrift:

_____

Stefan Mittermayr

1. Einleitung

1.1. Kurzfassung

Die Diplomarbeit "Speedometer – Verbesserung der GPS-Messung mittels zusätzlicher Sensoren" ist von Stefan Mittermayr während des fünften Jahrganges an der Höheren Technischen Bundeslehranstalt Perg für die Reife- und Diplomprüfung verfasst worden. Als Auftraggeber dieser wissenschaftlichen Arbeit tritt die Firma Runtastic GmbH auf, welche auf die Entwicklung von Fitness-Apps sowie dem Vertrieb von Fitness-Hardware spezialisiert ist.

Im Zuge der wissenschaftlichen Arbeit wurden Methoden zur genaueren Geschwindigkeitsmessung erarbeitet. Dabei wird insbesondere die Verwendung von Sensoren im Mobiltelefon-Bereich herausgearbeitet und die Interpretation der bereitgestellten Daten erklärt.

Es wird aufgezeigt, dass der Beschleunigungssensor im Smartphone sowohl zur Geschwindigkeitsmessung als auch zur Erkennung, ob ein Sportler pausiert oder einer Aktivität nachgeht, verwendet werden kann.

1.2. Abstract

The thesis "Speedometer – Improvement of the GPS measurement by use of additional sensors" is written by Stefan Mittermayr during the 5th class in order to complete the diploma examination at the vocational high school for IT in Perg. The principal of this dissertation is the Runtastic GmbH, which is known for the development of fitness apps as well as the distribution of training hardware.

In the course of this thesis, methods to improve the actual speed measurement have been developed. Especially, the use of sensors in mobile phones is carved out and the interpretation of provided data is explained.

It is shown, that the accelerometer in the smartphone can be used for measuring speed and for detection whether the athlete is moving or resting.

1.3. Vorwort

Im Vorfeld dieser wissenschaftlichen Arbeit möchte ich mich einerseits bei Dipl.-Ing. Stephan Brunner, Android-Teamleiter bei Runtastic, für die technische Begleitung der Diplomarbeit bedanken. Egal welches Problem während dem Programmieren zum Vorschein gekommen ist, mit Herrn Brunner konnte stets eine Lösung gefunden werden.

Zum anderen gilt mein Dank auch Dipl.-Ing. Rene Giretzlehner, der einer von vier Gründern des Unternehmens ist und mir die Absolvierung eines Praktikums im Sommer 2014 ermöglichte sowie die Diplomarbeit vermittelte.

Zu guter Letzt möchte ich Prof. Dipl.-Ing. Richard Kainerstorfer für die Betreuung der Diplomarbeit danken. Er stand mir im Projektentwicklungsunterricht immer mit Rat und Tat zur Seite.

1.4. Diplomand

Abbildung 1: Stefan Mittermayr

1.4.1. Persönliche Daten

Name	Stefan Mittermayr
Geburtsdatum und -ort	4. September 1995, Linz
Religionsbekenntnis	Römisch-katholisch

1.4.2. Ausbildung

Seit 09/2010	HTL Perg Höhere Lehranstalt für Informatik
09/2006 – 07/2010	BG/BRG Ramsauerstraße Linz Network-Realgymnasium
09/2002 – 07/2006	Volksschule Langenstein

1.4.3. Berufserfahrung

Seit 04/2013	Hornbach Baumarkt GmbH Samstagsaushilfe Verkauf
07/2014 – 08/2014	Runtastic GmbH Praktikant IT & Support
02/2014 & 04/2014	Synthesa Chemie GmbH Praktikant IT
07/2013	Gebauer & Griller Metallwerk GmbH Praktikant IT
07/2012 – 08/2012	ASFINAG Bau Management GmbH Praktikant Büro

1.4.4. Kontakt

Telefon	+43 664 104 72 80
E-Mail	stefan.mittermayr@outlook.com

2. Entstehung

2.1. Ausgangssituation & Aufgabenstellung

Derzeit erfolgt die Geschwindigkeitsmessung bei Aktivitäten, die mit Runtastic-Apps ausgeführt werden, rein mit den Daten des Global Positioning Systems (kurz GPS), welches beispielsweise auch bei Navigationsgeräten im Auto zum Einsatz kommt.

Nach einem Gespräch mit der Support-Abteilung hat sich abgezeichnet, dass manche Sportler bzw. Handys kurzfristig Probleme mit dem GPS-Empfang haben. Häufig konnte für dieses Problem keine logisch anmutende Erklärung gefunden werden.

Die Aufgabenstellung der Diplomarbeit liegt in der Entwicklung von Methoden für die Verbesserung der GPS-Geschwindigkeitsmessung mittels zusätzlicher Sensoren, speziell im Bereich unter 15 km/h. Die Realisierung ist durch das Sensor-Framework auf der Hardwareabstraktionsschicht der Android-Architektur zu lösen.

Etwas neuer als die GPS-Technologie ist die Verbreitung von Sensoren in Mobiltelefon. Der ansteigende Verkauf von Smartphones machte auch diese Technik alltagstauglich.

Die wissenschaftliche Arbeit hat die Auswahl des richtigen Sensors zu untersuchen und dessen Daten, falls nötig, zu filtern, aufzubereiten und für die Geschwindigkeitserrechnung zu verwenden.

2.2. Auftraggeber

Auftraggeber dieser wissenschaftlichen Arbeit ist die Runtastic GmbH.

Abbildung 2: Logo der Runtastic GmbH [1]

Runtastic GmbH

Pluskaufstraße 7

4061 Pasching

Das Unternehmen wurde 2009 gegründet und entwickelt seither Fitness-Apps für die mobilen Plattformen Android, Blackberry, iOS (Apples mobiles Betriebssystem für iPhones, iPods sowie iPads) und Windows Phone. Neben der Softwareentwicklung werden auch hauseigene Fitnessprodukte, wie beispielsweise eine Personenwaage oder ein Aktivitäts-Tracker mit Bluetooth-Anbindung angeboten. Das Angebot des Unternehmens wird abgerundet durch Trainingspläne, die speziell Personen im Abnehmen oder beim Erreichen eines Marathonziels unterstützen.

Seit der Gründung im Jahr 2009 wird das IT-Unternehmen von vier Fachhochschulabsolventen geleitet. Chief Executive Officer (CEO) der Organisation ist Florian Gschwandtner. Das Unternehmen ist im Grundlegenden folgendermaßen strukturiert:

Abbildung 3: Unternehmens-Struktur [2]

2.3. Projektauftrag

Um dem in der Ausgangssituation erwähnten Problem entgegenzuwirken, erfolgte der Projektauftrag seitens der Runtastic GmbH.

2.3.1. Auftraggeber

Name	Dipl.-Ing. Stephan Brunner
Funktion	Technische Unterstützung
Telefon	+43 660 1472219
E-Mail	stephan.brunner@runtastic.com

2.3.2. Projektbetreuer

Name	Prof. Dipl.-Ing. Richard Kainerstorfer
Funktion	Betreuung der Arbeit
E-Mail	r.kainerstorfer@htl-perg.ac.at

2.3.3. Diplomand

Name	Stefan Mittermayr
Telefon	+43 664 104 72 80
E-Mail	stefan.mittermayr@outlook.com

3. Planung

3.1. Pflichtenheft

3.1.1. Einleitung

Einführung

In diesem Pflichtenheft werden sämtliche programmierspezifische Details der Diplomarbeit "Speedometer" von Stefan Mittermayr abgeklärt. Es ist zu untersuchen, inwiefern die jetzt im Einsatz befindliche Geschwindigkeitsmessung, die rein auf den Daten des Global Positioning Systems (kurz GPS) basiert, um die Funktionalität eines weiteren Sensors erweiterbar ist (hinsichtlich Verbesserung der jetzigen Situation bzw. Lieferung von Daten bei Ausfall).

3.1.2. Zielbestimmung

Ziel der Diplomarbeit ist, basierend auf der bisherigen Geschwindigkeitsmessung eine Verbesserung zu erzielen. Die gemessene Geschwindigkeit ist durch Kombination mit den Daten der Sensoren des Mobiltelefons zu optimieren. Es ist eine Annäherung an die reale Geschwindigkeit zu implementieren. Als reale Geschwindigkeit ist eine Testfahrt mit einem Cadence Sensor zu machen. Die ermittelte "reale" Geschwindigkeit ist der gemessenen Geschwindigkeit dann gegenüberzustellen und mit den Möglichkeiten der Sensoren des Mobiltelefons zu optimieren.

3.1.3. Grenzkriterien

Runtastic definiert als fixen Teil der Aufgabenstellung den Soll-/Ist-Vergleich des GPS-Speeds mit "echtem" Speed, die Evaluierung der unterschiedlichen Berechnungsmethoden (Distanz/Zeit oder Doppler-Effekt) sowie eine Kombinationsstrategie von GPS & zusätzlichem Sensor (z. B. Accelerometer – Beschleunigungssensor)

Wunschkriterien

Hinsichtlich der Effizienz des zu implementierenden Algorithmus ist es gewünscht, eine effizientere Stillstands-Erkennung (Pause) mittels des Beschleunigungssensors zu ermöglichen und eine Evaluierung des zusätzlichen Akkuverbrauchs.

Abgrenzungskriterien

Nicht Teil der Diplomarbeit ist die Verwendung der optimierten Geschwindigkeitswerte für die Berechnung von Distanz oder sonstiger Werte einer Aktivität.

3.1.4. Produkteinsatz

Anwendungsbereiche

Zur Anwendung vorgesehen ist der zu programmierende Algorithmus in einer für die Android-Plattform zur Verfügung stehenden Applikation.

Zielgruppen

Von der optimierten Messmethode könnten die Kunden des IT-Unternehmens profitieren, welche sowohl Gelegenheitssportler als auch Profiathleten sind.

3.1.5. Produktkonfiguration

Software, Hardware, Orgware

Die zu entwickelnde Software muss bei Fertigstellung der Diplomarbeit auf denselben Android-Handys, auf denen Runtastic-Apps lauffähig sind, ohne Probleme funktionieren, so dass das Produkt auf allen bis jetzt unterstützten Geräten problemlos verwendet werden kann.

Produkt-Umgebung

Um die Funktionalität von GPS zu gewährleisten, ist ein Einsatz innerhalb eines Gebäudes oder eines zum Himmel verschlossenen Raumes nicht geeignet. Um die volle Funktionalität des Produktes sicherzustellen, ist also ein Einsatz im Freien vorausgesetzt.

Schnittstellen

Nach Ende einer Aktivität wird die gerade durchgeführte sportliche Leistung mit dem hauseigenen Webserver abgeglichen. Die Synchronisierung dieser Schnittstelle ist nicht Teil der Diplomarbeit und daher nicht zu berücksichtigen.

Betriebsbedingungen

Das Produkt wird im Freien bei unterschiedlichen Wetterbedingungen und Tageszeiten eingesetzt. Zudem kontrolliert der Sportler auch ab und an, wie schnell er gerade ist bzw. wie lange er für die bereits absolvierte Strecke gebraucht hat.

3.1.6. Datenlexikon

Laufende Nummer	1	2	3
Bezeichnung kurz	timestamp	speed	speedKmH
Bezeichnung lang	Zeitpunkt in Millisekunden (ms)	Geschwindigkeit in Metern/Sekunde (m/s)	Geschwindigkeit in Kilometern/Stunde (km/h)
Allgemeine Bemerkung	Vergangene Millisekunden seit 01.01.1970	—	—
Obergrenze	9.223.372.036.854.775.807	+/-3,4E+38	+/-3,4E+38
Untergrenze	−9.223.372.036.854.775.808	+/-1,4E-45	+/-1,4E-45
Standardwert	0	0.0f	0.0f
Einheit	ms	m/s	km/h
Datentyp mit Genauigkeit	Long 64-Bit	Double 64-Bit	Double 64-Bit
Abtastrate	~1 Wert/Sekunde		
Quelle	CSV-Datei, die der Speed & Cadence Sensor liefert		
Mengengerüst	Abhängig von der Dauer einer Aktivität (bei einer Stunde 3600 Messpunkte)		

Tabelle 1: Datenlexikon Cadence (Speed & Cadence Sensor/Bike-Sensor)

Laufende Nummer	4	5	6	7	8
Bezeich-nung kurz	gpsTi-mestamp	systemTi-mestamp	speed	speedKmH	location-Provider
Bezeich-nung lang	Zeitpunkt in ms	Zeitpunkt in ms	Geschwin-digkeit in Metern/Se-kunde (m/s)	Geschwin-digkeit in Kilome-tern/Stunde (km/h)	Quelle
Allgemeine Bemer-kung	Erfassungs-zeitpunkt des aktuel-len Mess-punktes	Verarbei-tungszeit-punkt des aktuellen Messpunk-tes	Geschwin-digkeit zum Erfassungs-zeitpunkt in m/s	Geschwin-digkeit zum Erfassungs-zeitpunkt in km/h	GPS, Mobil-funk oder WLAN (drahtloses Netzwerk)
Ober-grenze	9.223.372.036.854.775.807	9.223.372.036.854.775.807	+/-3,4E+38	+/-3,4E+38	—
Unter-grenze	−9.223.372.036.854.775.808	−9.223.372.036.854.775.808	+/-1,4E-45	+/-1,4E-45	—
Standard-wert	0	0	0.0f	0.0f	null
Einheit	Ms	ms	m/s	km/h	–
Datentyp mit Genau-igkeit	Long 64-Bit	Long 64-Bit	Float 32-Bit	Float 32-Bit	String
Abtastrate	~1 Wert/Sekunde				
Quelle	- CSV-Datei, die die angepasste Runtastic-App liefert - CSV-Datei, die der Speedometer liefert				
Mengen-gerüst	Abhängig von der Dauer einer Aktivität (bei einer Stunde 3600 Mess-punkte)				

Tabelle 2: Datenlexikon GPS-Sensor

Laufende Nummer	9	10	11	12	13	14
Bezeichnung kurz	timestamp	original	gravity	linear	magnitude	velocity
Bezeichnung lang	Zeitpunkt in ms	Beschleunigungswerte (Rohdaten)	Beschleunigungswerte (Gravitation)	Beschleunigungswerte (lineare Beschleunigung)	Gesamtlänge der linearen Beschleunigung	Geschwindigkeit
Allgemeine Bemerkung	Verarbeitungszeitpunkt des aktuellen Messpunktes	—	—	—	$M = SQRT(x^2 + y^2 + z^2)$	—
Obergrenze	9.223.372.036.854.775.807	+/- 3,4E+38	+/- 3,4E+38	+/- 3,4E+38	+/- 3,4E+38	+/- 3,4E+38
Untergrenze	−9.223.372.036.854.775.808	+/-1,4E-45	+/-1,4E-45	+/-1,4E-45	+/-1,4E-45	+/-1,4E-45
Standardwert	0	0.0f	0.0f	0.0f	0.0f	0.0f
Einheit	ms	m/s^2	m/s^2	m/s^2	m/s^2	m/s
Datentyp mit Genauigkeit	Long 64-Bit	Float-Array 32-Bit	Float-Array 32-Bit	Float-Array 32-Bit	Float 32-Bit	Float 32-Bit
Abtastrate	~50 Werte/Sekunde					
Quelle	- CSV-Datei, die die angepasste Runtastic-App liefert - CSV-Datei, die der Speedometer liefert					
Mengengerüst	Abhängig von der Dauer einer Aktivität (bei einer Stunde 180000 Messpunkte)					

Tabelle 3: Datenlexikon Accelerometer (Beschleunigungssensor)

3.1.7. Produkt-Funktionen

Kundenspezifische Funktionen

- Soll/Ist Vergleich GPS Speed mit "echtem" Speed
- Evaluierung des unterschiedlichen Berechnungsmethoden (Distance/Time oder Doppler Effekt)
- Kombinationsstrategie GPS & zusätzlicher Sensoren (z. B. Accelerometer)
- Präsentation des Ergebnisses
- Effizientere Stillstands-Erkennung (Pause) mittels Beschleunigungssensoren [optional]
- Evaluierung des zusätzlichen Akkuverbrauchs [wünschenswert]

Branchenspezifische Funktionen

Da die Diplomarbeit in der Praxis keine gängige Methode ist, kann nicht auf branchenspezifische Funktionen zurückgegriffen werden.

Funktion 1: Kombinationsstrategie von GPS & zusätzlicher Sensoren (z. B. Accelerometer – Beschleunigungssensor)

Diese Funktion ist Hauptaufgabe der Diplomarbeit und ist möglichst effizient zu implementieren. Für den Benutzer entsteht in der Benutzeroberfläche kein Unterschied, die Änderungen, die das Produkt bewirkt, laufen im Hintergrund ab.

Funktion 2: effizientere Stillstands-Erkennung (Pause) mittels des Beschleunigungssensors

Durch die Verzögerung von GPS erfolgt die automatische Pause-Funktion, welche in den Einstellungen der Runtastic-App aktiviert werden kann, sehr verzögert. Diese soll als Wunschkriterium verbessert werden.

Funktion 3: Evaluierung des zusätzlichen Akkuverbrauchs

Durch das Hinzunehmen der Sensordaten ist mit einem erhöhten Akkuverbrauch zu rechnen. Eine Ermittlung, wie viel Akku die Verwendung des zusätzlichen Sensors benötigt, ist durchzuführen.

3.1.8. Produkt-Leistungen

Trotz der Vielzahl der generierten Daten ist darauf zu achten, dass die Darstellung sämtlicher anderer Werte einer Aktivität ohne Verzögerung erfolgt. Die Aktualisierung der Geschwindigkeitswerte ist auf eine angemessene Darstellung auszurichten.

3.1.9. Qualitäts-Zielbestimmung

	Sehr wichtig	Von Bedeutung	Weniger wichtig	Unwichtig
Zuverlässigkeit	X			
Benutzerfreundlichkeit				X
Effizienz	X			
Portierbarkeit			X	
Kompatibilität		X		
Erweiterbarkeit			X	
Übersichtlichkeit			X	

Tabelle 4: Qualitäts-Zielbestimmung

3.1.10. Globale Testfälle

Testfall 1: Geschwindigkeitsmessung bei niedriger Geschwindigkeit (< 15km/h)

Es ist zu testen, ob der Algorithmus bei langsamer Geschwindigkeit funktioniert.

Testfall 2: Geschwindigkeitsmessung bei höherer Geschwindigkeit (> 15 km/h)

Dieser Testfall soll den Algorithmus bei mittlerer und schnellerer Beschleunigung sowie Geschwindigkeit überprüfen.

Testfall 3: Geschwindigkeitsmessung bei Aufbewahrung des Mobiltelefons am Arm/in der Hose

Im Gegensatz zu Radfahrer bewahren Läufer ihr Mobiltelefon während des Sports meistens am Arm oder in der Hosentasche auf. Es ist zu testen, ob es möglich ist, auch in diesem Bereich einen Sensor einzusetzen.

3.1.11. Entwicklungs-Konfiguration

Software, Hardware, Orgware

Als Software wird das Android Developer Toolkit, kurz ADT, der Fa. Google verwendet, um für die Android-Plattform geeigneten Code zu erstellen.

Zum Testen der Funktionen wird auf ein physisches Android-Mobiltelefon und nicht auf einen Emulator zurückgegriffen.

3.1.12. Musterergebnisse

Runtastic kann bis jetzt nicht auf Musterergebnisse zurückgreifen. Eine Recherche im Internet ergab wenige Projekte in diese Richtung, jedoch keine konkreten Ergebnisse.

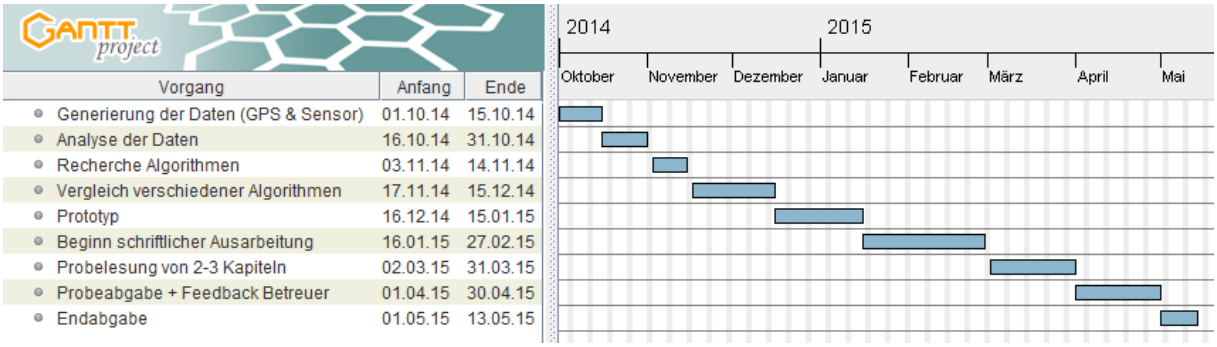
3.1.13. Abnahmen & Garantien

Die Diplomarbeit wird von Runtastic, im speziellen durch den Betreuer des Unternehmens, Stephan Brunner, abgenommen. Garantie sowie einen Wartungsvertrag für die einwandfreie Funktionalität der Software ist nicht gegeben.

3.1.14. Literatur

Als technische Dokumentation für die Android-Basis stellt Google auf der Website developer.android.com eine Referenz für das Android Source Development Kit zur Verfügung.

3.2. Projektablauf



Vorgang	Anfang	Ende
Generierung der Daten (GPS & Sensor)	01.10.14	15.10.14
Analyse der Daten	16.10.14	31.10.14
Recherche Algorithmen	03.11.14	14.11.14
Vergleich verschiedener Algorithmen	17.11.14	15.12.14
Prototyp	16.12.14	15.01.15
Beginn schriftlicher Ausarbeitung	16.01.15	27.02.15
Probelesung von 2-3 Kapiteln	02.03.15	31.03.15
Probeabgabe + Feedback Betreuer	01.04.15	30.04.15
Endabgabe	01.05.15	13.05.15

Abbildung 4: Gantt-Plan

Datum	Meilenstein/Tätigkeit
Mitte Oktober 2014	Generierung der Daten (GPS & Sensor)
Ende Oktober 2014	Analyse der Daten
Mitte November 2014	Recherche Algorithmen
Mitte Dezember 2014	Vergleich verschiedener Algorithmen
Mitte Jänner 2015	Prototyp
Ende Februar 2015	Beginn schriftlicher Ausarbeitung
Ende März 2015	Probelesung von 2-3 Kapiteln
Ende April 2015	Probeabgabe + Feedback Betreuer
Mitte Mai 2015	Endabgabe

Tabelle 5: Projektablauf

3.3. Projektstrukturplan

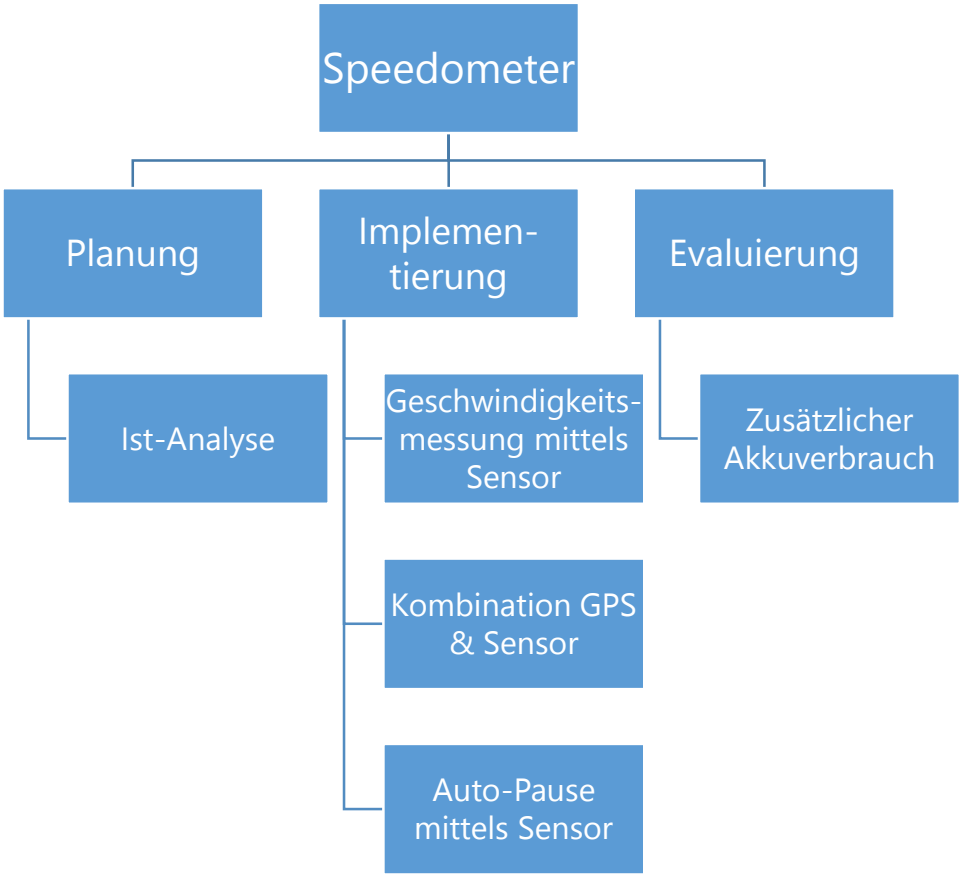


Abbildung 5: Projektstrukturplan

3.4. Ressourcenplan

3.4.1. Hardware

Zur Entwicklung dieser Android-Anwendung wurde ein Windows-Notebook mit aktueller Hardware (Vierkernprozessor, zwölf Gigabyte Arbeitsspeicher, Grafikkarte) verwendet. Als Testgerät wurde das eigene Android-Smartphone (Modell: Samsung Galaxy S4) verwendet.

Abbildung 6: Verwendetes Mobiltelefon (Samsung Galaxy S4) [3]

Angefallene Daten wurden in CSV-Dateien am Notebook abgelegt.

3.4.2. Software

Zum Entwickeln der Software wurde die freie Entwicklungsumgebung Eclipse mit dem Android Toolkit verwendet, welches eine Entwicklung von Android-Apps in einfacher Weise zulässt. Die Java-Basis für die Android-Entwicklung stellte zur Zeit der Entwicklung das Java Development Kit in der Version 8.

Zum Analysieren der aufgenommen Daten wurde Microsoft Excel in der Ausgabe 2013 verwendet.

Für alle grafischen Zwecke wurde das freie Bildbearbeitungsprogramm GNU Image Manipulation Program (kurz GIMP) verwendet.

4. Derzeitige Geschwindigkeitserfassung

4.1. Speed & Cadence Sensor (Bike-Sensor)

Um eine Analyse der Genauigkeit der GPS-Daten zur tatsächlichen Geschwindigkeit zu realisieren wird der Speed & Cadence Sensor der von Runtastic verwendet. Der Speed & Cadence Sensor ist ein auf dem Rahmen eines Fahrrades befestigter Sensor, der anhand jeweils eines Sensors auf einer Speiche und dem Pedal eine zuverlässige Geschwindigkeitsmessung ermöglicht. Die Daten werden dann bei einer Fahrt mit dem Rad aufgenommen.

Abbildung 7: Runtastic Speed and Cadence Sensor [4]

Ausgelesen werden diese Daten mit der Runtastic-Mountainbike-App, welche die Werte in einer CSV-Datei ablegt. Neben dem Geschwindigkeitswert steht auch der exakte Aufnahmezeitpunkt zur Verfügung.

4.2. GPS-Sensor

Generell unterscheidet man bei der Erfassung mittels GPS zwischen zwei Arten der Geschwindigkeitsermittlung:

- Distance/Time
- Doppler-Effekt

Während die Ermittlung der Geschwindigkeit über die zurückgelegte Distanz und vergangener Zeit erst ab zirka zehn Kilometern pro Stunde funktioniert und das auch ziemlich unstabil, ist die Geschwindigkeitserfassung mittels Doppler-Effekt genauer und auch im langsamen Bereich anwendbar. Die Funktionsweise bei der Erfassung mit dem Doppler-Effekt ist ähnlich wie man es beim Vorbeifahren eines Autos kennt. GPS kann durch das Messen der Frequenzverschiebung bei diesem Vorgang die Geschwindigkeit ermitteln.

Im Durchschnitt erfasst der GPS-Sensor pro Sekunde einen Wert mit exaktem Erfassungszeitpunkt und Geschwindigkeitswert (in m/s). Diese Implementation inklusive Filterung von fehlerhaften Werten ist bereits in der Runtastic-App vorhanden und kann in einer CSV-Datei ausgegeben werden.

Im Gegensatz zum Bike-Sensor stehen die Daten dieses Sensors nicht zur Millisekunde 0 zur Verfügung, sondern müssen zuerst umgerechnet werden, um einen vergleichbaren Wert mit dem Bike-Sensor zu erhalten.

4.2.1. GPS unter Android

Die GPS-Datenerfassung ist über die von der Android API bereitgestellte Klasse LocationListener implementiert. Diese Klasse ist seit API 1 fester Bestandteil von Android und somit auf jedem Gerät dieses Betriebssystems verfügbar.

GPS-Daten können von drei verschiedenen Quellen gewonnen werden:

- Satelliten
- WLAN-Netzwerke
- Mobilfunk

Berücksichtigt werden bei der Erfassung dabei nur die reinen Satelliten-Daten. Standortdaten eines Netzwerkproviders (Mobilfunk und WLAN) sind zu ungenau und daher nicht relevant.

Die Schnittstelle LocationListener stellt folgende Methoden bereit:

- public void onLocationChanged(Location location) { }
- public void onProviderDisabled(String provider) { }
- public void onProviderEnabled(String provider) { }
- public void onStatusChanged(String provider, int status, Bundle extras) { }

Von Bedeutung für die Geschwindigkeitsermittlung ist hierbei nur die Methode onLocationChanged(Location location). Über das Location-Objekt können schließlich sämtliche Informationen eines Messpunktes ausgelesen werden.

4.3. Vergleich von GPS- und Bike-Sensor

Die mit beiden Sensoren erfassten und kalkulierten Daten werden nun gegenübergestellt, um eine Analyse des Ist-Zustands zu erfassen. Auf der x-Achse ist die Fahrzeit in Sekunden und auf der y-Achse die Geschwindigkeit in Kilometern pro Stunde angegeben.

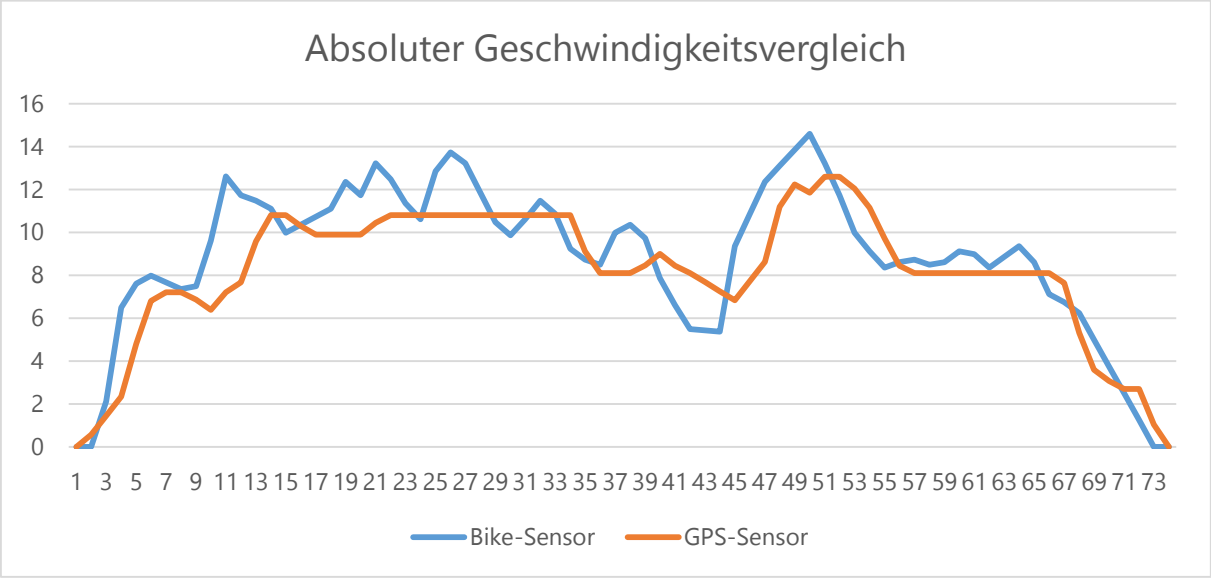


Abbildung 8: Absoluter Geschwindigkeitsvergleich (Bike-Sensor & GPS-Sensor)

Bei näherem Betrachten der Grafik fällt sofort auf, dass der GPS-Sensor leicht verzögert reagiert und bei starken Geschwindigkeitsänderungen nur einen Durchschnitt erfasst, jedoch nicht die näheren Veränderungen.

In dieser Grafik wird noch deutlicher, dass die zuvor erwähnten schnellen Geschwindigkeitsänderungen vom GPS-Sensor sehr ungenau erfasst werden. Auf der x-Achse ist wieder die Fahrzeit in Sekunden und auf der y-Achse die Geschwindigkeitsänderung innerhalb dieser Sekunde angegeben.

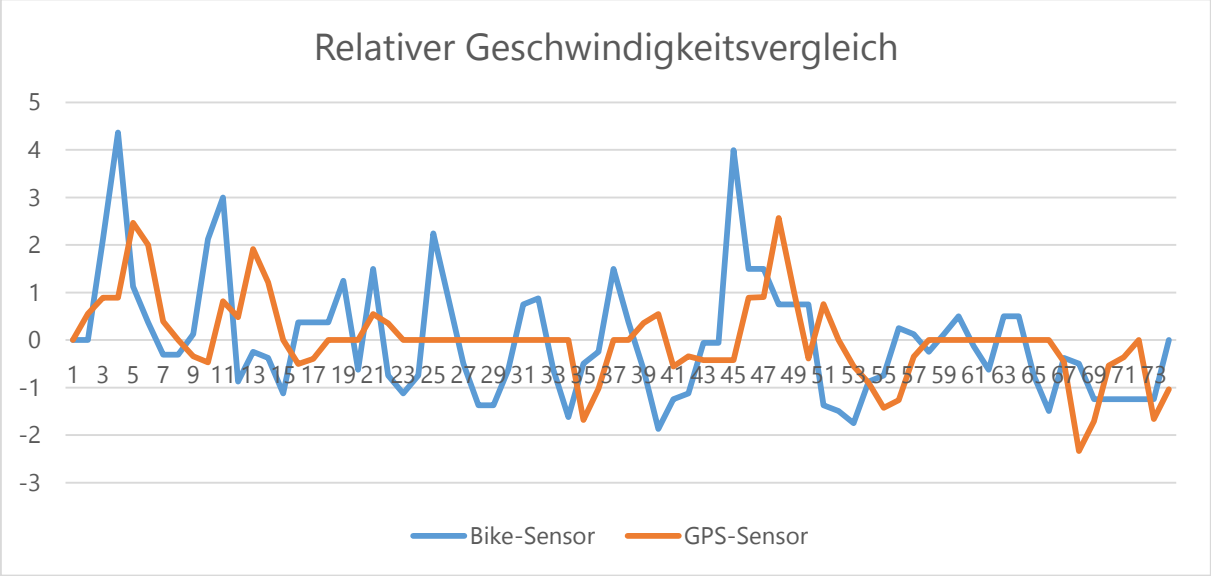


Abbildung 9: Relativer Geschwindigkeitsvergleich (Bike-Sensor & GPS-Sensor)

5. Realisierung

5.1. Technologien

5.1.1. Android-Architektur

Da es in dieser Diplomarbeit hauptsächlich um die Verwendung von Sensoren in Mobiltelefonen geht, möchte ich kurz die Position dieser in der Android-Architektur darstellen.

Sämtliche Sensoren befinden sich auf der Hardwareabstraktionsschicht (englisch Hardware Abstraction Layer), die direkt über dem Linux-Kernel liegt. Ohne hardwarenaher Programmierung kommt mittlerweile keine funktionsfähige Anwendung mehr aus. Auf derselben Schicht sind auch beispielsweise das Audio- und Kamera-Interface zu finden. Sämtliche Komponenten dieser Schicht verdeutlicht folgende Grafik:

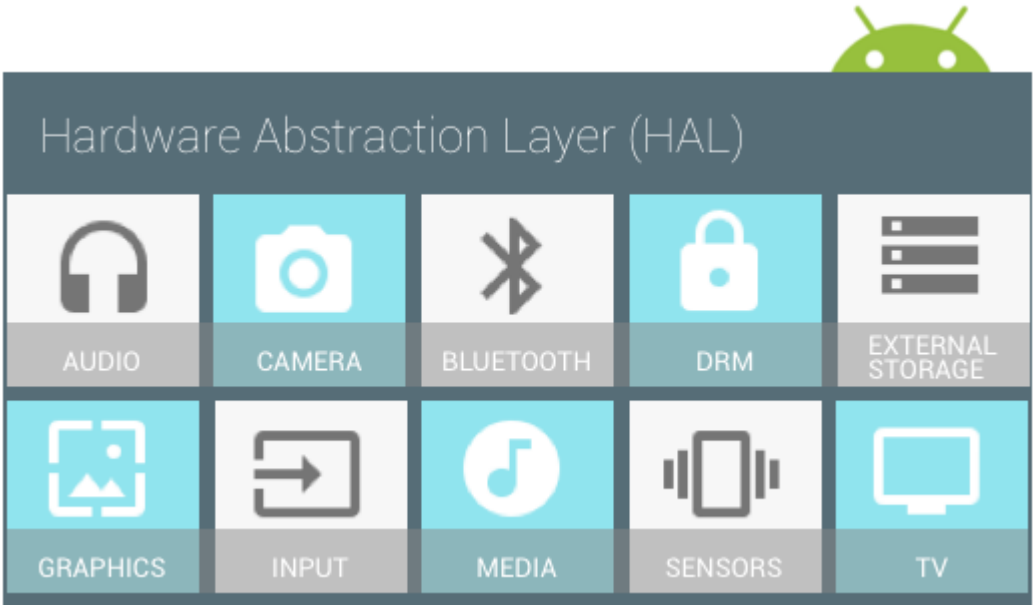


Abbildung 10: Hardwareabstraktionsschicht von Android [5]

5.1.2. Android-APIs

Auch wenn Android, im Vergleich zu Microsofts Windows, als relativ neues Betriebssystem auf dem Markt ist, ist die Anzahl der unterschiedlichen Versionen nicht zu vernachlässigen. Jede neue Version des Android Source Development Kits (Android SDK) bringt ebenfalls wieder neue Funktionen oder Grafik-Möglichkeiten mit sich.

Um hier Ordnung zu halten, hat sich Google für die Einführung von so genannten API-Stufen entschieden. Anhand einer API-Stufe kann man feststellen, ab welcher Betriebssystem-Version eine Klasse bzw. eine Funktionalität unterstützt wird und daher lauffähig ist. Von Wichtigkeit ist dies besonders, wenn in einer App neue Technologien verwendet werden sollen, die Anwendung aber auch auf Smartphones älterer Baujahre noch laufen soll. Ein Blick auf die entsprechende API-Stufe der Klasse gibt Auskunft, welche Version des Betriebssystems mindestens auf dem Ziel-Handy laufen muss, um die Funktionalität sicherzustellen.

Die wichtigsten APIs sind Folgende:

API-Stufe	Android-Version
8	2.2
10	2.3.3
15	4.0.3
16	4.1
17	4.2
18	4.3
19	4.4
21	5.0
22	5.1

Tabelle 6: API-Stufen und Android-Versionen

5.1.3. Android-Manifest bzw. Rechteverwaltung

Wie bereits zuvor erwähnt, muss man bei Verwendung von GPS oder Sensoren auf Elemente der Hardwareabstraktionsschicht zugreifen. Damit eine Anwendung aber überhaupt Hardware-Komponenten benutzen darf, müssen diese im Android-Manifest freigeschalten werden. Das Manifest ist die Konfigurationsdatei für eine App, ohne der keine Android-Anwendung auskommt. Standardmäßig kann eine App gar nichts aus der Hardwareschicht benutzen, erst durch das Erlauben in dieser XML-Datei kann zugegriffen werden. Für meine Diplomarbeit benötige ich folgende Berechtigung in der Manifest-Datei, um den Accelerometer als auch GPS nutzen zu können:

```
<uses-feature android:name="android.hardware.sensor.accelerometer" android:required="true" />

<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
<uses-permission android:name="android.permission.ACCESS_MOCK_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Abbildung 11: Berechtigungen in der Manifest-Datei

Ebenfalls in der Manifest-Datei festgelegt wird beispielsweise, ab welcher API-Stufe eine App generell läuft oder welche Aktivitäten alle zu der Applikation dazugehören.

5.1.4. Android-Aktivitäten

In Android werden Container, auf denen UI-Elemente angezeigt werden, als Aktivität (englisch Activity) bezeichnet. Eine Applikation kann aus mehreren Aktivitäten bestehen, die dann auch untereinander gewechselt werden können. Jede App muss dabei mindestens eine Aktivität als Ausgangslage haben, ähnlich wie bei einer Website, die immer eine Startseite braucht. Dies wird im Manifest im Abschnitt application folgendermaßen definiert:

```
<activity
    android:name="com.example.speedometer.activities.MainActivity"
    android:screenOrientation="portrait"
    android:label="@string/app_name_short" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Abbildung 12: Startaktivität einer Android-App

Um beim Starten oder Schließen bestimmte Aktivitäten durchzuführen stellt Android sechs Methoden bereit, die zu unterschiedlichen Zeiten abgerufen werden. Das Verständnis der Methoden ist essentiell für eine korrekte Programmierweise. Die nachfolgende Grafik zeigt den Zusammenhang der verschiedenen Methoden und jenes Ereignis, welches die Methode aufruft.

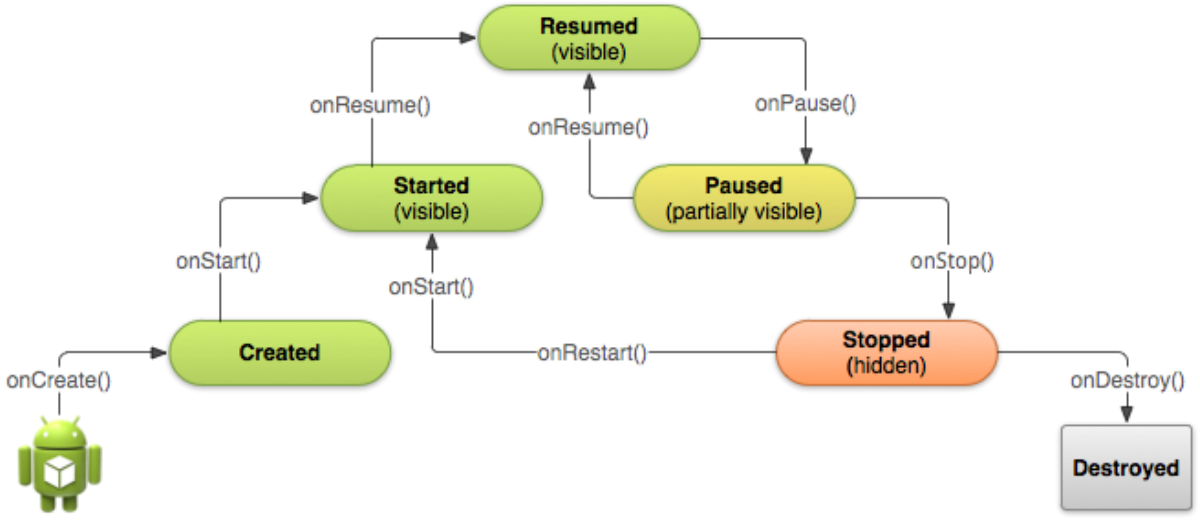


Abbildung 13: Lebenszyklus einer Android-Anwendung [6]

5.1.5. Android: Layouts

Sämtliche Layouts, die der Benutzer schließlich auf dem Bildschirm angezeigt bekommt werden in XML-Dateien beschrieben. Für die Gestaltung eines Layouts bietet Android eine umfangreiche Palette an Elementen und Layouts. Die Entwicklungsumgebung Eclipse bietet zudem einen grafischen Entwurf eines Layouts an und schreibt grundlegenden Code selbst in die XML-Datei dahinter, die auch bearbeitet werden kann. Ein fertig zusammengestelltes Layout schaut beispielsweise so aus:

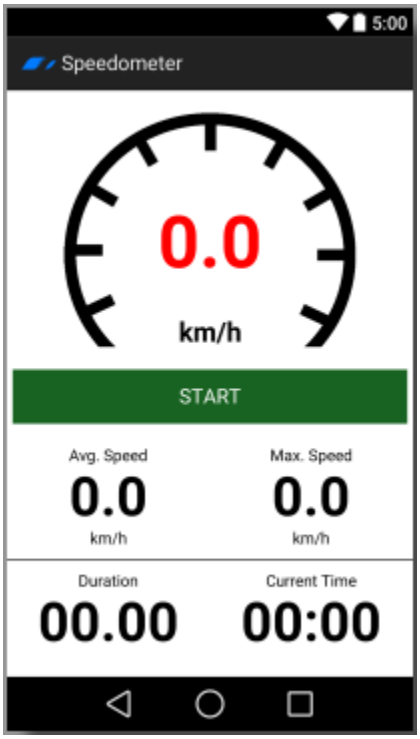


Abbildung 14: Speedometer-Design

5.1.6. Java

Die Grundlage für Android ist das Java Development Kit, welches im Unterricht an der HTL Perg bereits ab der 1. Klasse zum Einsatz kommt und daher keine Sprachschwierigkeiten oder Barrieren gegeben waren.

5.1.7. XML

XML in der Version 1.0 wird in Android für sehr viele Bereiche verwendet. Zu den wichtigsten Verwendungszwecken zählen:

- Layouts
- Zeichenketten
- Themes
- Menüs
- Manifest-Datei

5.2. Tools

5.2.1. Eclipse & Android Toolkit

Zu Beginn der Diplomarbeit stand die Installation und Konfiguration der Entwicklungsumgebung an. Dabei musste zwischen Eclipse mit dem Android Toolkit oder Android Studio entschieden werden. Zu Beginn der Entwicklung war Android Studio noch im Beta-Status, wodurch die Auswahl auf Eclipse fiel. Der Umgang mit Eclipse ist ähnlich einfach wie NetBeans. Zudem bietet das Programm, wie zuvor erwähnt, auch grafische Unterstützung bei der Entwicklung von Designs.

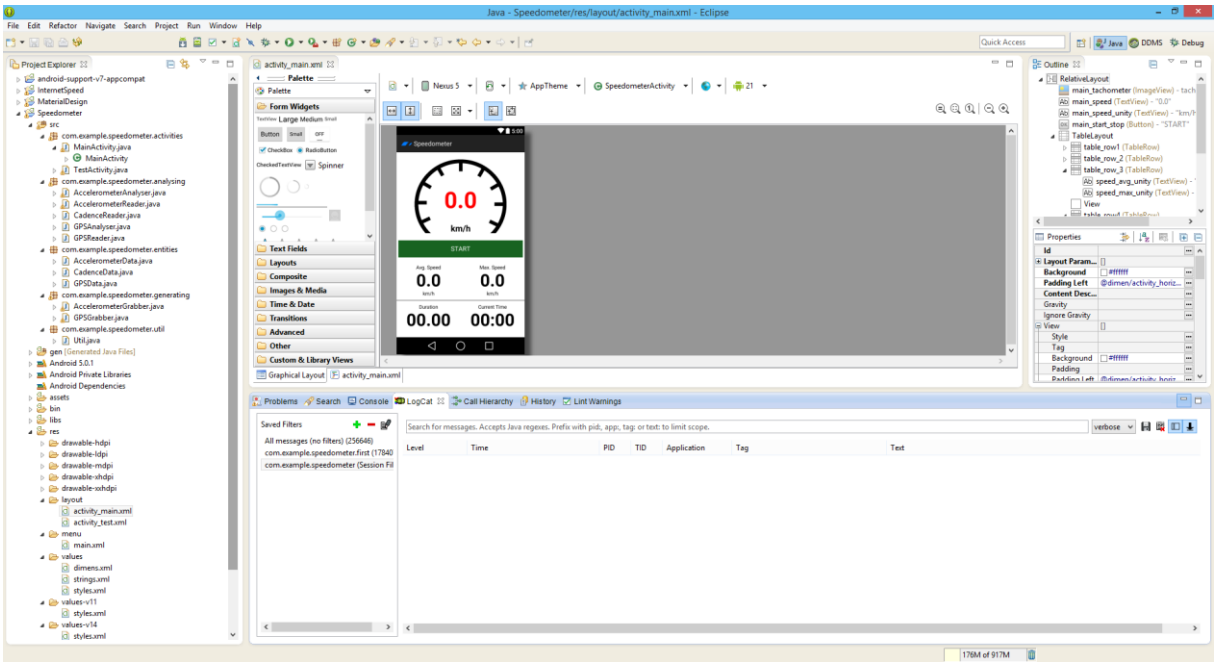


Abbildung 15: Eclipse mit Android Toolkit

5.2.2. Microsoft Excel

Mit Microsoft Excel wurden in CSV-Dateien geschrieben Daten ausgelesen, analysiert und grafisch aufbereitet.

5.2.3. Grafikbearbeitung

Für die Grafikerstellung (z. B. der Tachometer oder die Grafiken in der Diplomschrift) wurde die freie Software GNU Image Manipulation Program (kurz GIMP) verwendet.

5.3. Implementierung

5.3.1. Allgemein

Um eine strukturierte Android-Anwendung zu entwickeln, wurde der Programmcode in fünf Packages aufgeteilt. Das Android-Projekt ist folgendermaßen strukturiert:

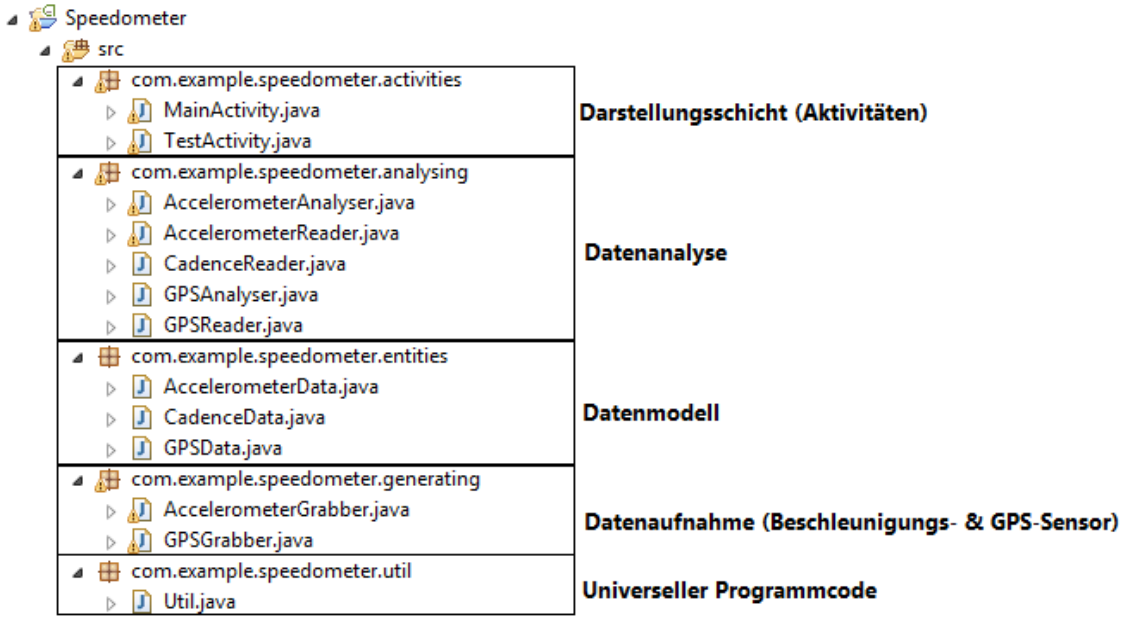


Abbildung 16: Programmaufbau

Beide Aktivitäten sind im Layout-Ordner (res/layout) durch ein Layout-File dargestellt, in welchem die einzelnen UI-Elemente mittels XML beschrieben sind.

Innerhalb der Datenanalyse erfolgen der Hauptteil der Anwendung, nämlich die Kalkulation der Geschwindigkeit und die Ermittlung, ob gerade pausiert wird oder nicht.
Im Entities-Package wurde für jeden Sensor eine Klasse angelegt, die einen Messpunkt dieses Sensors samt seinen Werten repräsentiert.

Das Generating-Package implementiert die Schnittstellen zu den Sensoren. Der AccelerometerGrabber stellt die Abfrage der Daten des Beschleunigungssensors sicher. Der GPSGrabber implementiert die Schnittstelle zur Lokalisierung unter Android und kann so Geo-Daten von GPS, Mobilfunknetzwerken oder WLAN-Netzwerken empfangen.

5.3.2. Zeitaufwand

Task	Stunden
Installation und Konfiguration Android SDK	8
Einlesen in Android	25
Pflichtenheft	7
Ist-Analyse mit Cadence-Sensor & GPS-Sensor	12
Implementierung SensorManager (Accelerometer)	10
Algorithmus-Findung	14
Geschwindigkeitsfunktion und testen	16
Entwicklung Prototyp	20
Accelerometer-Filtering	7
Pausefunktion implementieren und testen	11
Evaluierung zusätzlicher Akkuverbrauch	3
Dokumentation / Diplomschrift	34
Summe	**167**

Tabelle 7: Zeitaufwand

5.3.3. Sensoren unter Android

Das Android SDK stellt seit API 1 verschiedene hardware- und softwarebasierte Sensoren zur Verfügung. Hardwarebasierte Sensoren erfassen Rohdaten. Dies sind unbearbeitete Daten in dem Zustand, wie sie aufgenommen wurden. Softwarebasierte Sensoren dagegen filtern die aufgegriffenen Daten, bevor die Werte dem Programmierer zur Verfügung gestellt werden. Alle Sensortypen werden über die Klasse SensorManager verwaltet. Die Initialisierung des Managers erfolgt folgendermaßen:

```
// Set up sensor stuff
accGrabber = new AccelerometerGrabber(this);
sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
sensorAcc = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

Abbildung 17: Sensor-Initialisierung

5.3.4. Arten von Sensoren

Android stellt grundsätzlich drei verschiedene Arten von Sensoren bereit. Dies sind Bewegungssensoren, Umgebungssensoren und Positionssensoren.

Zu den Bewegungssensoren zählen der "Accelerometer", der "Gravity sensor", der "Gyroscope" und der "Rotational vector sensor".

Mit den Umgebungssensoren kann unter anderem die Lufttemperatur, der Luftdruck, die Ausleuchtung sowie die Luftfeuchtigkeit erfasst werden.

Die Positionssensoren umfassen den Orientierungssensor und den Magnetometer.

5.3.5. Implementierung eines Sensors

Zur Implementierung eines der Sensoren steht das Interface SensorEventListener zur Verfügung. Dieses erfordert bei Implementierung das Überschreiben der Methoden

- onAccuracyChanged(Sensor sensor, int accuracy) und
- onSensorChanged(SensorEvent event).

Die Methode onAccuracyChanged wird aufgerufen, sofern sich die Genauigkeit des Sensors ändert. Vier Statusvariablen geben über die Genauigkeit nach der Änderung Auskunft:

- SensorManager.SENSOR_STATUS_ACCURACY_HIGH – hohe Genauigkeit
- SensorManager.SENSOR_STATUS_ACCURACY_MEDIUM – mittlere Genauigkeit
- SensorManager.SENSOR_STATUS_ACCURACY_LOW – geringe Genauigkeit
- SensorManager.SENSOR_STATUS_UNRELIABLE – Genauigkeit kann nicht gewährleistet werden.

Von Interesse für die Datenerfassung eines Sensors ist die Methode onSensorChanged. Der übergebene Parameter event enthält das Array values, in welchem die Daten eines Messpunktes erfasst werden.

Um zu regeln, wie oft der Sensor Daten abfragen soll, gibt es vier Variablen, die die Verzögerung (engl. Delay) regeln:

- SensorManager.SENSOR_DELAY_FASTEST
- SensorManager.SENSOR_DELAY_GAME
- SensorManager.SENSOR_DELAY_UI
- SensorManager.SENSOR_DELAY_NORMAL

Je nach Option werden die Werte alle 20 bis 200 Millisekunden aktualisiert.

5.3.6. Beschleunigungssensor

Für diese Diplomarbeit von Interesse sind die drei Beschleunigungssensoren Accelerometer, Gravity und Linear Acceleration.

Typ	Daten	Verfügbar seit
TYPE_ACCELEROMETER	Beschleunigung inklusive Gravitation in X-, Y- und Z-Achse	API 1
TYPE_GRAVITY	Gravitation in X-, Y- und Z-Achse	API 9
TYPE_LINEAR_ACCELERATION	Beschleunigung ohne Gravitation in X, Y und Z-Achse	API 9

Tabelle 8: Beschleunigungssensoren

Auf den Werten des Sensors wirkt standardmäßig die Gravitation. Je nachdem, wie das Smartphone positioniert ist, wirkt die Gravitation einmal auf die X-Achse, Y-Achse oder Z-Achse.

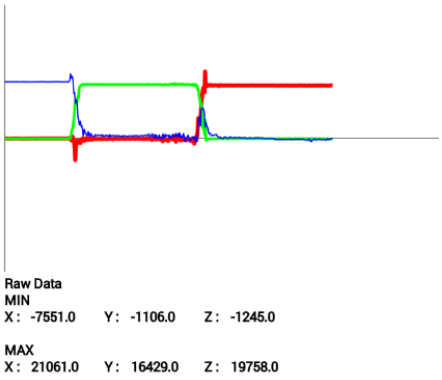


Abbildung 18: Gravitationseinfluss auf den Beschleunigungssensors

Jeder Beschleunigungssensor hat für die Werte der X-, Y und Z-Achse verschiedene Erfassungsbereiche. Das Android SDK gibt zur Vereinfachung die Koordinaten automatisch in der SI-Einheit m/s^2 zurück und nicht die nummerischen Werte des Sensors.

Wichtig für das Verständnis der Werte dieses Sensors ist das Wissen über das dreidimensionale Koordinatensystem. Das Koordinatensystem jedes Beschleunigungssensors arbeitet nach folgendem Muster:

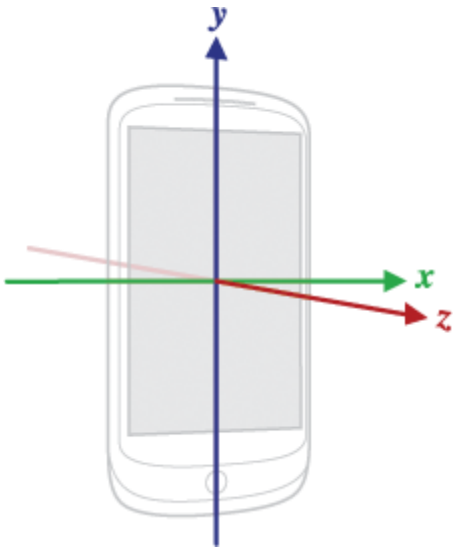


Abbildung 19: Koordinatensystem des Beschleunigungssensors unter Android

Um von einem Beschleunigungssensor Daten zu holen, wird er aktiv gesetzt:

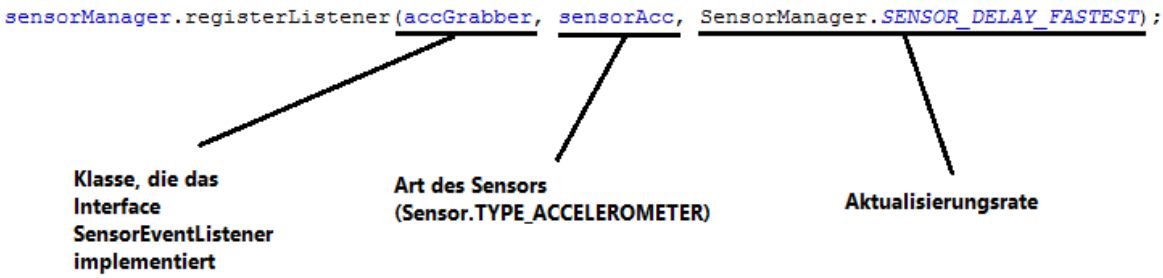


Abbildung 20: Registrierung des Beschleunigungssensors

In der Methode onSensorChanged innerhalb der Klasse, welche das SensorEventListener-Interface implementiert, kann dann auf die Werte zugegriffen werden.

5.3.7. Datenfilterung

Um die Gravitation herauszufiltern, bediene ich mich der Hilfe eines Low-Pass-Filters und eines High-Pass-Filters. Damit isoliere ich die Gravitation von der Beschleunigung und übrig bleibt nur mehr die reine Beschleunigung:

```
// low-pass filter
gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0];
gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1];
gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2];

// high-pass filter
linear[0] = event.values[0] - gravity[0];
linear[1] = event.values[1] - gravity[1];
linear[2] = event.values[2] - gravity[2];
```

Abbildung 21: Filterung der Daten

5.3.8. Geschwindigkeitsermittlung

Die Geschwindigkeitsmessung mit dem Beschleunigungssensor basiert auf der Formel:

$$v(t1) = v(t0) + a * dt$$

v(t1) … Neue Geschwindigkeit

v(t0) … Alte Geschwindigkeit

a … Aktuelle Beschleunigung

dt … Vergangene Zeit zwischen einem Beschleunigungswert und dem Nächsten (t1 – t0)

Mit der Formel wird die Beschleunigung für die Dauer, in der sie aufgetreten ist, der aktuellen Geschwindigkeit hinzugerechnet.

5.3.9. Auto-Pause-Funktion

Um eine Pause-Funktion anhand des Beschleunigungssensors implementieren zu können, habe ich die Y-Achse gewählt, deren lineare Beschleunigung beobachtet wird. Besteht über eine längere Dauert, das heißt mehrere Werte des Sensors in Folge, keine oder nur eine sehr geringe Beschleunigung, so kann man annehmen, dass die Person gerade steht.

Ob gerade der automatische Pause-Modus aktiviert ist, erkennt man an der Farbe der Geschwindigkeit. Ist diese Orange, so ist der Pause-Modus aktiv.

Um wieder aus dem Pause-Modus zu kommen, genügt eine Bewegung nach vorne, die von Dauer ist. Überschreiten die letzten gemessenen Werte einen Schwellenwert, so ist anzunehmen, dass man sich in Bewegung gesetzt hat und die Pause-Funktion wird ausgeschaltet. Daraufhin wird die Aktivität wieder gestartet und der Speed-Zähler erscheint in grüner Farbe.

6. Qualitätssicherung

6.1. Herausforderungen & Probleme

6.1.1. Einstieg in Mobile Computing

Da ich bisher noch nicht im Bereich des Mobile Computings entwickelt habe und Ähnliches noch nicht im Unterricht durchgemacht wurde, bestand der erste Teil der Diplomarbeit aus der Einarbeitung in die neue Materie. Aufgrund der Basissprache Java, auf dem das Android SDK basiert, ist der Einstieg nicht so schwer gewesen.

6.1.2. Gravitation

Anfangs bestand Skepsis, ob es möglich ist, die Gravitation aus den Werten des Beschleunigungssensors zu filtern. Nach ausführlicher Recherche fanden sich aber verschiedenste Algorithmen, wobei zwei in der Diplomarbeit angewendet wurden.

6.2. Testfälle

Um den entwickelten Code zu testen, wurde am Rahmen eines Fahrrades eine Handyhalterung angebracht in welcher das Smartphone mit der Speedometer-App befestigt wurde. Der Testaufbau schaute nach Montage folgendermaßen aus:

Abbildung 22: Testumgebung

Es wurde sowohl bei langsamer Fahrt als auch bei rasantem Bergabfahren getestet und lieferte, verglichen mit GPS, gute Ergebnisse.

7. Evaluierung

7.1. Zusätzlicher Akkuverbrauch

Da durch die Nutzung einer weiteren hardwaretechnischen Komponente, dem Sensor-Framework, zusätzlicher Akkuverbrauch zu erwarten war, wurde in den abschließenden Arbeiten der Diplomarbeit berechnet, wie viel Akku die zusätzliche Komponente erfordert.

Mit der Methode getPower kann von einem Sensor der Verbrauch in mA pro Minute abgerufen werden. Der zusätzliche Verbrauch bei höchster Empfangsrate beträgt 0,25 mA/Minute. Aufgerechnet auf eine Aktivität mit einer Länge von einer Stunde sind das 15 mA mehr, die die Sensorik benötigt. Verglichen mit den heutigen Akkus in Smartphones (zwischen 2000 und 3000 mAh) ist dieser zusätzliche Verbrauch als gering einzustufen.

7.2. Kompatibilität

7.2.1. Abwärtskompatibilität

Seit der Markteinführung von Android im Jahre 2008 in der Version 1.0 sind viele Entwicklungszyklen vergangen, sodass die aktuelle Version des mobilen Betriebssystems die Version 5.1.1 ist. Gewöhnlicher Weiße werden Android-Smartphones aber wenig bis gar nicht upgedatet, sodass es nur wenige Geräte mit der aktuellsten Version gibt. Bei der Entwicklung wurde daher darauf geachtet, dass der implementierte Algorithmus auf Geräten, auf denen die Version 2.3 oder höher läuft, ohne Einschränkung lauffähig ist. Gemäß aktueller Verteilung der verschiedenen Versionen läuft der entwickelte Programmcode daher auf 99,7 % aller mit Android benutzten Geräte.

Version	Codename	API	Distribution
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.3%
4.1.x	Jelly Bean	16	15.6%
4.2.x		17	18.1%
4.3		18	5.5%
4.4	KitKat	19	39.8%
5.0	Lollipop	21	9.0%
5.1		22	0.7%

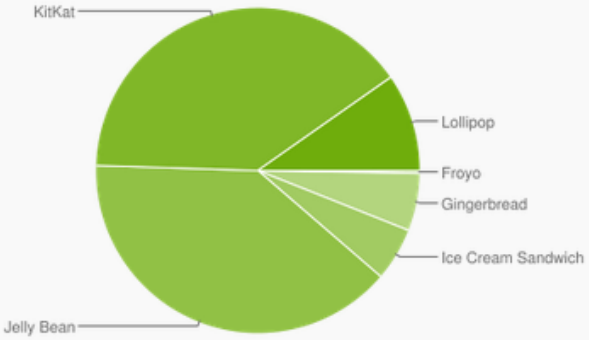


Abbildung 23: Kompatibilität [7]

7.2.2. Kompatibilität zu anderen Plattformen

Da Runtastic auch auf anderen Plattformen verfügbar ist, wurde im Zuge der Diplomarbeit geprüft, ob auf iOS und Windows Phone der verwendete Sensor ebenfalls zur Verfügung steht.

Bei Windows Phone ist der Sensor im Package Microsoft.Devices.Sensors implementiert, während unter iOS die Klasse UIAccelerometer vorhanden ist.

7.3. Teilnahme am P@BS-Wettbewerb

Wie alle Projekte und Diplomarbeiten der Abschlussklassen der Schule nahm auch meine Diplomarbeit am schulinternen Wettbewerb, der vom Absolventenverband der HTL Perg (kurz P@BS) jährlich organisiert wird, teil. Die wissenschaftliche Arbeit wurde der Kategorie Wirtschaft zugeteilt und einer Fachjury bestehend aus Professoren und Absolventen präsentiert. Neben vier weiteren ausgewählten Projekten dieser Kategorie wurde die Diplomarbeit am 28. April 2015 vor Eltern, Professoren und Unternehmensvertretern im Zuge einer Abendveranstaltung präsentiert und der dritte Platz unter den wirtschaftlichen Projekten erreicht.

7.4. Resümee

Durch die Entwicklung einer größeren Android-App konnte ich zum ersten Mal die Entwicklung für mobile Geräte näher kennenlernen. Ich denke, dass speziell durch den Boom von Smartphones und Gadgets wie Smartwatches oder Fitnessbänder das Thema Pervasive Computing, wie der Fachbegriff für die immer weiter fortschreitende digitale Vernetzung lautet, auch in den nächsten Jahren noch eine große Rolle spielt und Knowhow auf diesem Gebiet sicherlich nicht schadet.

8. Abkürzungsverzeichnis & Glossar

Begriff	Erklärung
Accelerometer	Beschleunigungssensor
API	Application Programming Interface, die Programmierschnittstelle
bzw.	beziehungsweise
GPS	Global Positioning System
Gravitation	Erdanziehungskraft, die auf den Werten der Sensoren liegt
IT	Informationstechnologie
SDK	Source Development Kit, Entwickler-werkzeuge
z. B.	zum Beispiel

Tabelle 9: Glossar

9. Abbildungsverzeichnis

10. Tabellenverzeichnis

11. Quellenverzeichnis

[1] http://mediacenter.runtastic.com/

[2] http://www.gruenderservice.at/

[3] http://www.samsung.com/

[4] http://shop.runtastic.com/

[5] http://developer.android.com/

[6] http://developer.android.com/

[7] http://developer.android.com/