ulm university universität
uulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Entwurf and Implementierung eines Frameworks für Mobile Context-Awareness im Bereich eHealth

Abschlussarbeit an der Universität Ulm

Vorgelegt von:
Peter Hösch
peter.hoesch@uni-ulm.de
856162

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Robin Kraft

2020

Fassung 30. November 2020

Satz: PDF-LaTeX 2ε

Abstract

Das Ziel der vorliegenden Bachelorarbeit war es, eine Middleware zu entwickeln, welche die Sensoren von Android- und iOS-Geräten nutzt, um Informationen über die Situation, in welcher sich das Gerät und sein Nutzer befinden, zu erlangen. Bei diesen Informationen handelt es sich im speziellen um den Ort, an welchem sich das Smartphone befindet, seine Position relativ zum Nutzer und die gegenwärtige Fortbewegungsart des Nutzers. Von den verfügbaren Sensoren wurden das GPS, Gyroskop, Accelerometer und Magnetometer verwendet. Es wurden sowohl frühere Projekte ähnlicher Art betrachtet als auch eigene Messungen in praxisnahen Situationen durchgeführt. Die Umsetzung fand mithilfe von Ionic statt. Die Bachelorarbeit ist interessant für Personen, welche selbst eine derartige Middleware entwickeln oder eine bereits bestehende Middleware für kontextsensitive Anwendungen in ein anderes Projekt integrieren wollen.

Inhaltsverzeichnis

1 Einleitung **1**

1.1 Struktur der Arbeit 2

2 Grundlagen **3**

2.1 Mobile Context 3

2.1.1 Mobile Computing 3

2.1.2 Context-Aware Computing 5

2.1.3 Sensor Fusion 7

2.1.4 Definition von Kontext und Kontextsensitiviät 8

2.1.5 Middleware für kontextsensitive Anwendungen 9

2.2 eHealth 11

2.2.1 Allgemeines 11

2.2.2 mHealth 12

2.2.3 eHealth und Context Awareness 13

3 Anforderungen **15**

3.1 Funktionale Anforderungen 15

3.1.1 Sensorauswertung 15

3.1.2 Ortserkennung 16

3.1.3 Fortbewegungsart 16

3.1.4 Geräteposition 16

3.1.5 Rückmeldung der Zuverlässigkeit 17

3.1.6 Kommunikation mit Anwendungsprogramm 17

3.2 Nichtfunktionale Anforderungen 17

3.2.1 Performanz 17

3.2.2 Energieeffizienz 18

3.2.3 Modularität 18

3.2.4 Erweiterbarkeit 18
3.2.5 Wartbarkeit 19
3.2.6 Intuitivität der Schnittstellen 19
3.2.7 Zielplattformen 19

4 Konzeption **20**
4.1 Plattform-Auswahl und -Analyse 20
4.2 Architektur 21
4.3 Sensoren 22
4.3.1 GPS 23
4.3.2 Rotationssensor 23
4.3.3 Beschleunigungssensor 24
4.3.4 Magnetometer 24
4.3.5 Umgebungslichtsensor 24
4.4 Auswertung 24
4.4.1 Ortserkennung 25
4.4.2 Fortbewegungsart 26
4.4.3 Geräteposition 28

5 Implementierung **30**
5.1 Sensormodule 30
5.1.1 Elternklasse 30
5.1.2 GPS 31
5.1.3 Rotationssensor 31
5.1.4 Beschleunigungssensor 32
5.1.5 Magnetometer 32
5.1.6 Umgebungslichtsensor 33
5.2 Datensammlung 33
5.2.1 Hilfsprogramm 34
5.2.2 Fortbewegungsart 34
5.2.3 Geräteposition 35
5.2.4 Weitere Verarbeitung der gesammelten Daten 36
5.3 Analysemodule 39
5.3.1 Ortserkennung 39
5.3.2 Fortbewegungsart 40

5.3.3 Geräteposition 40

6 Evaluation **41**
6.1 Erfüllung der Anforderungen 41
6.1.1 Funktionale Anforderungen 41
6.1.2 Nichtfunktionale Anforderungen 42
6.2 Mögliche Fehlerquellen 43
6.2.1 Messfehler 43
6.2.2 Auswahl der Sensoren 43
6.2.3 Verarbeitung der Rohdaten 43
6.2.4 Klassifikatoralgorithmus 44
6.2.5 Trainingsdaten 44
6.3 Anwendungsbeispiel 44

7 Zusammenfassung und Ausblick **46**
7.1 Ergebnis 46
7.2 Ausblick 47

Literatur **48**

1 Einleitung

Die Idee der Kontextsensitivität in mobilen Anwendungen, also der Verwendung von Informationen über die Umgebung des Nutzers zur Optimierung von Anwendungsabläufen, beschäftigt Informatiker schon seit Ende des zwanzigsten Jahrhunderts [41]. Hierbei sind mobilen Computersysteme wie Smartphones oder Personal Digital Assistents (PDAs) für die Anwendung dieses Konzeptes prädestiniert. Dies liegt darin begründet, dass sich die Umgebung des Nutzers während der Verwendung in einem deutlich stärkeren Maße verändert als bei feststehenden Computersystemen und sie über eine große Anzahl an Sensoren zur Ermittlung dieser verfügen. Mit der wachsenden Verbreitung von mobilen Computersystemen ist daher auch die Relevanz dieses Themas heute höher als je zuvor [27].

Kontextsensitive Programme werden in einer großen Anzahl an Anwendungsfeldern eingesetzt, etwa in der Tourismusbranche [42] und der Lehre [1], aber auch im Gesundheitswesen, wo diese Technik zum Beispiel eingesetzt wird um die Überwachung stationärer Patienten zu vereinfachen [21].

Da nicht jeder Entwickler einer kontextsensitiven Anwendung die Erkennung des Kontextes selbstständig programmieren will oder kann, existieren einige Bibliotheken, welche in neue Projekte eingebunden werden können und derartige Funktionen bereit stellen, wie zum Beispiel AWARE [18]. Hierbei werden vor allem die beiden am weiten verbreiteten mobilen Betriebsysteme unterstützt - Android und iOS [36].

Das Ziel dieser Arbeit ist es, selbst eine solche Bibliotheken zu entwerfen und implementieren. Hierbei soll der Fokus auf der Verwendung für eHealth-Anwendungen, also Anwendungen aus dem Bereich des Gesundheitswesens, liegen. Deshalb soll zum Testen der Bibliothek auch noch eine prototypische eHealth-Anwendung geschrieben werden.

Hierbei bestehen vielfältige Herausforderungen. Zum einen ist es schwierig, die

richtigen Schlussfolgerungen über die Umgebung des Nutzers aus den begrenzten Daten von Smartphone-Sensoren zu ziehen, was die Entwicklung und Kalibrierung des dafür zuständigen Algorithmus zu einem komplexen Unterfangen macht. Zum anderen verfügen Smartphones nur über begrenzte Ressourcen, sowohl was Rechenleistung als auch was Speicherplatz und Energieversorgung angeht, welche mit viel Bedacht verwaltet werden müssen, um die praktische Nutzbarkeit des Systems zu erhalten [55]. Besonders die beständige Benutzung der Sensorik kann hierbei zu einem Problem werden, da diese das Potential eines sehr hohen Energieverbrauchs hat [24].

1.1 Struktur der Arbeit

In dieser Arbeit werden in Kapitel 2 zunächst einmal 'Kontextsensitivität' und 'eHealth' sowie Verknüpfungen zwischen den beiden im Allgemeinen thematisiert. Hierbei werden diese Begriffe zunächst definiert und eingegrenzt, verwandte Themenbereiche angesprochen und Beispiele aus Forschung und Praxis vorgestellt. Danach werden in Kapitel 3 die Anforderungen an das selbstgeschriebene Framework festgelegt, bevor der grundlegende Aufbau geplant wird. Dann wird in Kapitel 4 auf die Schwierigkeiten während der Implementierung eingegangen, und schließlich werden in Kapitel 5 einige Test geplant und durchgeführt um die Funktionalität zu überprüfen und im Falle von Fehlern noch einmal Anpassungen vornehmen zu können. Im letzten Kapitel 6 werden dann die Erkenntnisse noch einmal zusammengefasst und es wird über Ausbaumöglichkeiten nachgedacht.

2 Grundlagen

In diesem Kapitel werden für diese Arbeit relevante Begriffe und Konzepte erklärt. Hierbei werden zwei große Themenkomplexe beschrieben: Mobile Context, mit dessen Ermittlung und Analyse sich der Rest dieser Arbeit beschäftigen wird, und eHealth, das Anwendungsfeld für welches das Ergebnis dieser Arbeit letztendlich gedacht ist.

2.1 Mobile Context

Das Thema des mobilen Kontextes beschäftigt Informatiker schon seit Ende des zwanzigsten Jahrhunderts. Worum genau es sich dabei handelt ist eine viel diskutierte Frage [41]. Im Folgenden wird der Begriff eingegrenzt und einige konkrete Beispiele für Umsetzungen dieses Konzeptes betrachtet.

2.1.1 Mobile Computing

Die ersten modernen Computer waren gewaltige Apparate, welche von Experten bedient wurden um diverse wissenschaftliche oder militärische Aufgabenstellungen zu lösen. Diese Geräte waren oftmals schrankgroß und hätten höchstens in Einzelteilen an einen anderen Ort verlagert werden können [37].

Auch später, als die Miniaturisierung weiter voranschritt, Röhren durch Transistoren abgelöst und Computer auch an Firmen oder sogar Privatpersonen vermarktet wurden, war eine mobile Nutzung von Computern zunächst nur recht eingeschränkt denkbar. Zwar wäre es durchaus möglich gewesen, die frühen PCs (Personal Computer) mit sich herumzutragen und unterwegs zu benutzen - aufgrund ihres noch recht hohen Gewichts und ihrer Unhandlichkeit wäre dies nur recht unangenehm

und aufgrund der mangelnden integrierten Stromversorgung auch von geringer Effektivität gewesen [37].

Dies änderte sich mit der Entwicklung der ersten Notebooks beziehungsweise Laptops. Erstmals war es möglich, einen Computer ohne großen Aufwand zu transportieren und an verschiedenen Orten zu nutzen [37]. Während bei Notebooks der Übergang zwischen Transport und Nutzung noch recht zeitaufwendig ist - man muss erst eine passende Oberfläche zum Abstellen finden, ihn aufklappen und hochfahren bevor man in der Lage ist Anwendungsprogramme zu starten - sieht dies bei Mobiltelefonen ganz anders aus.

Auf diese kann sehr schnell zugegriffen werden, sodass auch bei kurze Interaktionen nur ein geringer Overhead entsteht, etwa um schnell eine Nachricht zu lesen beziehungsweise zu verfassen oder einzelne Informationen im Internet nachzuschlagen. Bei einigen Funktionen überlappen sich sogar die aktive Nutzung und der bloße Transport - zum Beispiel verfügen moderne Smartphones einen Schrittzähler, welcher, anhand der Erschütterungen die das Gerät erfährt, versucht die gelaufene Strecke einer Person anzunähern [4].

Diese Verschmelzung zwischen Computernutzung und anderen Abläufen wird mit den seit einigen Jahren aufkommenden *Wearables* noch weiter verstärkt. Hierbei handelt es sich um Computersysteme, welche während der Anwendung am Körper des Benutzers getragen werden oder in dessen Kleidung integriert sind. Einige Beispiele wären Smartwatches, Datenbrillen oder Fitnessarmbänder. Die Integration der Nutzung dieser in andere Tagesabläufe wird durch verschiedene Strategien gefördert: Die passive Sammlung von Sensordaten, welche entweder erst später vom Nutzer analysiert werden oder über die er nur bei besonderen Ereignissen informiert wird (zum Beispiel ein Fitnessarmband, welches die Vitalwerte des Trägers protokolliert und ihm hilft innerhalb des gewünschten Bereichs zu bleiben), eine Benutzeroberfläche die möglichst unkomplizierten Zugriff auf Funktionen und Informationen gewährt (zum Beispiel eine Smartwatch welche die vom aktuellen Träger meistgenutzten Funktionalitäten direkt am Handgelenk anbietet) oder durch die Anpassung von bereitgestellten Optionen an die aktuelle Umgebung des Nutzers (zum Beispiel eine Datenbrille welche dem Träger ohne Aufforderung Informationen über Objekte im Sichtfeld präsentiert) [28].

Ein zentraler Unterschied zwischen mobilen und nicht-mobilen Computern ist es,

dass bei ersteren das Umfeld des Gerätes/Nutzers sowie deren Zustände, der sogenannte *Kontext*, deutlich stärker variieren als bei letzteren (für eine genauere Definition des Begriffes siehe 2.1.4). Feststehende Computer sind für gewöhnlich auf einen einzelnen, relativ konstanten Raum beschränkt, während Smartphones und Wearables häufig von ihren Nutzern überall hin mitgenommen werden [8]. Auch werden feststehende Computer meistens auf die selbe Art genutzt - der Nutzer befindet sich vor einem fest installierten Monitor, fast immer sitzend, und konzentriert sich auf die aktive Anwendung. Dies ist bei ihren mobilen Gegenstücken anders. Zum Beispiel kann ein Smartphone komplett passiv als Schrittzähler, nebenbei als Taschenlampe oder Musikspieler, oder mit vollem Fokus für Videos oder Spiele verwendet werden.

Ein weiter Punkt, in dem sich die beiden Computerkategorien unterscheiden, ist die Art auf welche sie neue Informationen erhalten. Bei feststehenden Computern liegt der Fokus klar darauf, direkt Daten übertragen zu bekommen, entweder von anderen Computersystemen oder vom Benutzer. Hierfür verfügen sie über diverse optische Laufwerke sowie Steckverbinder [38]. Bei mobilen Computern sind diese Hardwarebestandteile zwar auch vorhanden, allerdings in geringerer Quantität, da diese Geräte um ihre Mobilität zu erhalten deutlich kleiner sein müssen, was die Verwendung von großen optischen Datenträgern, Kabelverbindungen oder Peripheriegeräten erschwert beziehungsweise sogar ausschließt. Stattdessen verfügen mobile Computer meistens über Sensoren, mit denen sie in der Lage sind direkt Informationen über ihre Umgebung zu sammeln. Während sich diese bei Notebooks für gewöhnlich auf Kamera und Mikrophon beschränken, werden in Smartphones normalerweise verschiedenste Sensoren verbaut, welche Informationen wie die Beschleunigung beziehungsweise Rotation des Gerätes, dessen absolute Position und Ausrichtung auf dem Planeten, Temperatur oder Helligkeit messen können [23]. Diese Sensoren verleihen den Geräten die Möglichkeit, den oben genannten Kontext effektiver zu bestimmen.

2.1.2 Context-Aware Computing

Kontextsensitivität (englisch Context Awareness) bezeichnet die Fähigkeit von Programmen, Informationen über ihre Umgebung beziehungsweise ihren 'Kontext' zu benutzen um ihr Verhalten zu optimieren. Ein oben bereits genanntes Beispiel hier-

für wäre die ortsabhängige Darstellung von Informationen, welche heute bereits an manchen Orten aktiv genutzt wird (etwa im Rahmen einer Multimedia-Tour in der Tate Gallery of Modern Art in London [45]).

Es wird zwischen verschiedenen Arten von Kontextsensitivität unterschieden. Eine der Kategorien die hierbei verwendet wird ist der Kontrast zwischen passiver und aktiver Kontextsensitivität. Passive Kontextsensitivität bedeutet, dass das Computersystem den Benutzer zwar mit Informationen über seine Umgebung versorgt und dieser auf der Basis jener Informationen Empfehlungen ausgeprochen bekommt, aber das System sein Verhalten nicht automatisch an den Kontext anpasst. Ein Beispiel hierfür wäre Google Maps, welches nach seinem Start direkt Informationen über den aktuellen Aufenthaltsort des Nutzers anzeigt, aber nur näher auf diese eingeht, falls der Benutzer beschließt sie anzuklicken. [26] Im Gegensatz dazu trifft bei aktiver Kontextsensitivität das System autonom Entscheidungen basierend auf Kontext-Informationen. In diese Kategorie gehört die oben genannte Multimedia-Tour, bei der nach jeder Annäherung an ein Kunstwerk automatisch Informationen über dieses aufgerufen wurden [45]. Dies bedeutet, dass der Nutzer auf der einen Seite entlastet wird da er weniger Entscheidungen eigenständig treffen muss, auf der anderen Seite entsteht dadurch auch das Risiko, dass durch das System Entscheidungen getroffen werden welche gegen die Wünsche des Nutzers gehen. Im schlimmsten Fall wird dadurch das exakte Gegenteil des eigentlich gewünschten Effektes hervorgerufen: Der Nutzer muss Aktionen rückgängig machen und gegen die tatsächlich gewünschten Aktionen substituieren, wodurch er zusätzlich belastet wird [23].

Eine weitere Unterscheidung wird zwischen den verschiedenen Arten von Kontext getroffen, auf die sich ein System berufen kann. Hierbei existieren mehrere Methoden der Kategorisierung. Vorstellbar wäre zum Beispiel eine Unterteilung in physischen Kontext (Temperatur, Lautstärke, Helligkeit, etc), Benutzerkontext (Profil, biometrische Informationen, geographische Position, etc) und temporalen Kontext (Uhrzeit, Datum, Wochentag, Jahreszeit, etc) [23]. Manche Quellen führen als weitere Kontextart den Gerätekontext ein, welcher Dinge wie Netzwerkverbindung oder Akkuladestand umfasst [55].

Kontext kann generell auf zwei verschiedene Arten gewonnen werden: Das Auslesen von Sensoren und das Ziehen von Schlüssen aus bereits bekannten Informationen. Hierbei werden rohe Sensordaten als 'low-level' Kontext bezeichnet, wäh-

rend die fertig ausgewerteten Informationen, welche dann auch direkt zur Entscheidungsfindung genutzt werden können, 'high-level' Kontext genannt werden. Dieser Prozess der Auswertung von Rohdaten ist eine komplexe Aufgabe, für die ein App-Entwickler unter Umständen weder Interesse noch Kapazitäten hat. Aus diesem Grund existiert das Konzept einer so genannten 'Middleware', ein Programm welches Sensorinformationen vom Betriebssystem ausliest, verarbeitet und dann die so aufbereiteten Daten an Anwendungsprogramme weitergibt.

2.1.3 Sensor Fusion

Die Sensor Fusion ist ein Prozess in dem die Daten von verschiedenen Sensoren auf eine Art verarbeitet werden, sodass das Endergebnis mehr Informationen enthält als die vereinigten Einzeldaten. Das menschliche Gehirn kann als der Gold-Standard für dieses Konzept betrachtet werden: Es verwendet Informationen aus verschiedenen Informationsquellen (Augen, Ohren, Nase, Zunge und Haut), welche verschiedenartige Informationen liefern (Visuell, Akustisch, Olfaktorisch, Gustatorisch, Taktil), um ein Gesamtbild seiner Situation zu erschaffen und komplexe Entscheidungen zu treffen [13].

Hierbei kann zum einen einfach die Qualität der Daten verbessert werden, indem die Informationen aus mehreren gestörten, gleichartigen Quellen verrechnet werden um eine Approximation der originalen, ungestörten Daten zu erhalten [23]. Sensorfusion dieser Art wird bereits standardmäßig von jedem Android-Smartphone verwendet, in Form der Fused Location Provider API, welche Signale aus mehreren Quellen verwendet (etwa GPS und Wi-Fi) um die Position des Gerätes zu ermitteln [22]. Zum anderen ist es auch möglich, komplett neue, höherstufige Daten zu erlangen. Vorstellbar wäre hierbei etwa ein Roboter oder selbstfahrendes Auto, welcher beziehungsweise welches durch die Kombination von mehreren visuellen oder Ultraschall-Sensoren ein 3D-Abbild seiner Umgebung generiert [11].

Diese zweite Verwendungsmöglichkeit ist auch für die Ermittlung des Kontextes sehr interessant. So kann man etwa mithilfe des GPS feststellen, mit welcher Geschwindigkeit sich das Gerät und somit auch der Nutzer bewegt. Wenn man nun feststellen möchte, mithilfe welches Fortbewegungsmittels dies geschieht, reicht diese einzelne Information aber nicht aus. Die Geschwindigkeit beim schnellen Laufen und beim langsamen Fahrradfahren können gleich sein, ebenso beim schnellen

Fahrradfahren und beim langsamen Autofahren [32]. Hierbei können nun zusätzliche Sensoren, etwa das Accelerometer, hinzugezogen werden: Die Verbindung aus gemessener Geschwindigkeit und Erschütterungen erlaubt eine recht präzise Identifizierung der Fortbewegungsart [56].

2.1.4 Definition von Kontext und Kontextsensitiviät

Es herrscht keine generelle Einigkeit darüber, was 'Kontext' und 'Kontextsensitiviät' im Bezug auf Computersysteme genau bedeutet. Dies geht so weit, dass einige in dem Feld tätige Personen sich sogar weigern, diese Begriffe zu definieren [18].

Im folgenden Abschnitt werden verschiedene bereits existierende Versuche einer Definition betrachtet, diese miteinander verglichen und schließlich versucht, eine eigene, umfassende Definition aufzustellen.

Die Idee der Kontextsensitivität wurde erstmals 1994 charakterisiert. Hierbei wurde der Begriff 'Kontextsensitive Software' erklärt als solche Programme, welche in der Lage sind sich an die Position des Geräts, die Objekte und Personen in der Nähe des Geräts, sowie Veränderungen dieser anzupassen [41]. Man beachte, dass im Kontrast zu späteren Verwendungen des Begriffs der 'Kontext' lediglich Informationen über die Positionen von Objekten umfasst, nicht jedoch über ihre sonstigen Eigenschaften. Dies ist schlüssig, wenn man betrachtet, welches konkrete Projekt in diesem Artikel beschrieben wurde: Eine elektronische Karte, für die nun mal Positionsinformationen am relevantesten sind. Diese Praxis, die Definition von 'Kontext' an das jeweilige Projekt anzupassen, kann immer wieder angetroffen werden: Bei Untersuchungen welche sich mit PDAs beschäftigen umfasst der Kontext vor allem Dinge welche die Sensorik dieser Geräte direkt messen kann [53], während bei der Entwicklung von Interfaces für Datenbrillen der Kontext in seinem Kern als die Aktivitäten des Nutzers charakterisiert wird.

Wie in früheren Analysen der Begriffe 'Kontext' und 'Kontextsensitiv' bereits angemerkt wurde [16], gibt es trotz der häufig unterschiedlichen Schwerpunkte durchaus Aspekte, welche in jeder Definition gleich sind. Eine Anwendung kann immer dann als 'Kontextsensitiv' bezeichnet werden, wenn ihr Verhalten vom Kontext beeinflusst wird. Wie genau diese Beeinflussung aussieht unterscheidet sich von Programm zu Programm - von der einfachen Speicherung relevanter Informationen bis hin zur au-

tomatischen Durchführung von Routinen. 'Kontext' in diesem Zusammenhang ist, um eine möglichst weite Anzahl an Optionen abzudecken, jegliche Information welche die Umstände beschreibt, in denen sich eine Entität wie eine Person (inklusive des Benutzers), ein Gerät (inklusive des ausführenden Gerätes), ein Ort oder ein Objekt mit Relevanz für die Anwendung befindet.

Definition 1 (Kontext) *Kontext ist jegliche Information, welche die Umstände beschreibt, in denen sich eine Entität wie eine Person, ein Gerät, ein Ort oder ein Objekt mit Relevanz für die Anwendung befindet.*

Definition 2 (Kontextsensitivität) *Eine Anwendung ist genau dann kontextsensitiv, wenn ihr Verhalten vom Kontext beeinflusst wird.*

2.1.5 Middleware für kontextsensitive Anwendungen

Wie oben erwähnt gibt er bereits einige Frameworks, welche in Anwendungsprogramme eingebunden werden und für diese den Kontext ermitteln können. Diese Middlewares operieren auf unterschiedliche Arten und für verschiedene Anwendungen gedacht sind.

Ein Beispiel wäre AWARE [18]. Dieses Open-Source Framework für Android und iPhones verfügt über die Möglichkeit, eine weite Anzahl an Quellen für den Kontext zu nutzen, die hier in drei Kategorien unterteilt werden: Software (andere Anwendungen, das OS oder Netzwerkübertragungen), Hardware (klassische Sensoren wie Accelerometer, Barometer oder GPS) und Menschlich (Touchscreen, Text-to-speech oder Tastatur). Es verfügt über eine Anzahl an Plugins, welche verschiedene Arten an höherleveligem Kontext aus diesen Quellen abstrahieren, wie zum Beispiel die Fortbewegungsart des Nutzers oder ob der Nutzer momentan eine Unterhaltung führt. Diese Informationen können dann auf drei Arten an andere

Anwendungen weitergegeben werden: Broadcasts (eine Funktion der Anwendung wird vom System aufgerufen sobald sich der Kontext ändert), Providers (die Kontext wird auf dem Gerät gespeichert und kann von anderen Anwendungen aktiv abgefragt werden) und Observer (welche auf bestimmte Kontextänderungen reagieren). AWARE sieht sich hierbei als ein "Context Instrumentation Framework For Everyone", welches von verschiedenen Zielgruppen gleichermaßen verwendet werden kann. Als Beispiele werden von den Entwicklern Privatpersonen angegeben, welche sich dafür interessieren welche Informationen ihr Smartphone über sie sammeln kann oder Teile dieser Daten verwenden wollen, Anwendungsentwickler, welche eine kontextsensitive Anwendung schreiben und die Datenermittlung sowie Auswertung nicht selbst implementieren wollen, und Wissenschaftler, welche Smartphone-basierte Studien durchführen. Auf die letztere Gruppe wird hierbei ein besonderes Augenmerk gelegt - die Entwickler des Frameworks geben den Wunsch nach der Wiederverwertbarkeit von Forschungsergebnissen und nach der Kooperation zwischen Wissenschaftlern als eine ihrer Motivationen an [18].

Ein weiteres Beispiel ist CASS (Context-Awareness SubStructure). Dieses ist serverbasiert, das heißt während die Informationen auf mobilen Computersystemen gesammelt werden, findet die Verarbeitung auf einem anderen, mit der Hilfe von drahtloser Netzwerkverbindung verknüpften Computer statt. Dies entlastet die Akkus der mobilen Computer, erlaubt die Verwendung von komplexeren Algorithmen und ermöglicht die Sammlung von Informationen von einer Anzahl physisch getrennter Geräte. Auf der anderen Seite macht dies das System allerdings anfällig für Störungen der Verbindung. Dies soll teilweise durch lokales Caching von Informationen ausgeglichen werden. Zudem könnte es durch die Übertragung zu Problemen mit dem Schutz der persönlichen Daten des Nutzers kommen, falls der Betreiber des Servers sich vom Besitzer des Mobilgerätes unterscheidet. Derartige Bedenken werden allerdings von den Entwicklern nicht angesprochen. Höherleveliger Kontext wird mithilfe einer ’Inference Engine’ aus bereits bekannten Fakten generiert, indem eine Ansammlung auf Regeln befolgt wird welche die Zusammenhänge zwischen Zuständen repräsentiert. Dies ist eines der großen Features von CASS - durch die Trennung zwischen dem Anwendungscode und der Regelsammlung ist es möglich, die Kontextanalyse anzupassen ohne das Framework neu kompilieren zu müssen. Zudem erlangen Personen mit nur geringen Programmierkenntnissen die Möglichkeit, diese Kontextanalyse zu verändern und erweitern. Es existieren eine Reihe

von Anwendungen, welche CASS verwenden, so zum Beispiel MALLET (Maintence Assignment Listing Lightweight Electronic Tool), welches die Organisation von Wartungsaufgaben im häuslichen Bereich auf eine kontextsensitve Art erlaubt [17].

2.2 eHealth

2.2.1 Allgemeines

Laut der WHO ist eHealth die Benutzung von Informations- und Kommunikationstechnologien zur Förderung der Gesundheit [47]. Es handelt sich hierbei um ein sehr weites und diverses Feld. Ein einfaches und weitverbreitetes Beispiel wäre hierbei die Elektronische Krankenakte (eKA), eine institutionsinterne Sammlung der medizinischen Daten eines Patienten, als das elektronischer Äquivalent der traditionellen, in Papierform existierenden Krankenakte [50]. Auch schon simple Praktiken wie die Kommunikation zwischen Patient und Arzt über E-Mails können als Beispiel für eHealth angesehen werden.

Auf der anderen Seite umfasst eHealth auch einige deutlich komplexere und außergewöhnlichere Technologien. Ein prestigeträchtigeres Exemplar davon ist die Möglichkeit, mit der Hilfe von Robotern eine Operation ohne physisch anwesenden Arzt durchzuführen [9]. Ein anderes Beispiel wäre eine App, welche medizinische Notfälle des Trägers detektiert und automatisch den Notarzt ruft [2].

Bis auf einige grundlegende Beispiele werden die Möglichkeiten, welche eHealth in der Theorie bietet, praktisch eher selten genutzt. Dafür gibt es verschiedene Gründe. Einer davon ist, dass viele der Technologien nur als *Proof of Concept* oder Prototyp existieren und noch nicht marktreif sind. Häufig ist die Technologie noch zu teuer um die Nutzung zu rechtfertigen oder das Verfahren ist nicht zuverlässig genug und muss noch weiter getestet werden [50]. Aber nicht nur von der Seite der Entwickler, auch von der Seite der potentiellen Nutzer treten Probleme auf. Eines der größten Bedenken ist der Datenschutz - gesundheitliche Daten sind äußerst persönlich und es kann potentiell zu katastrophalen Konsequenzen kommen, sollten diese in die falschen Hände geraten. Aus diesem Grund ist die digitale Erfassung, Speicherung und Übertragung von Patientendaten ein brisantes Thema, welches sowohl von Gesetzgebern als auch Bevölkerung äußerst kritisch betrachtet

wird [50]. Ein weiteres Bedenken ist die Sicherheit und Zuverlässigkeit von derartigen Technologien, gerade bei lebenswichtigen Geräten wie etwa Implantaten. Eine Sicherheitslücke, zum Beispiel bei einem Herzschrittmacher, kann hier zu schweren Verletzungen bis hin zum Tod führen [14].

Dem gegenüber steht die Vielzahl an Perspektiven, die dem Gesundheitssystem durch die Einbindung der elektronischen Datenverarbeitung eröffnet wird. Durch die Verwendung von Telemedizin könnten dünn besiedelte Gebiete, in denen der nächste Arzt oft weit entfernt ist, besser versorgt werden. Eine verbesserte Kommunikation zwischen verschiedenen Gesundheitsdienstleistern würde einen tieferen Einblick in die medizinische Historie des Patienten geben und somit Zeit sparen und Fehlbehandlungen verringern. Die großflächige Sammlung und Analyse von Informationen über die Effekte welche verschiedene Therapien auf verschiedene Menschen haben könnte in Zukunft eine Verbesserung der Behandlungen ermöglichen [50].

Es bleibt abzuwarten welchen Kompromiss unsere Gesellschaft zwischen den Schwierigkeiten und den Möglichkeiten dieser Technologien findet.

2.2.2 mHealth

mHealth (mobile health) ist eine Unterkategorie der eHealth. Es geht hierbei um die Nutzung mobiler Computersysteme für medizinische Verfahren. Zu diesen Computersystemen zählen sowohl Smartphones und Tablets, aber auch eigens für diesen Zweck entwickelte Sensorik, etwa in der Form von Fitnessarmbändern [34].

Es besteht eine Vielzahl technischer Anwendungsmöglichkeiten. Neben dem Auslesen und Sammeln von gesundheitsbezogenen Daten wie Blutzuckerstand oder der Fernüberwachung chronisch Kranker besteht auch die Möglichkeit, mobile Interventionen durchzuführen. Hierbei wird versucht Patienten zu unterstützen indem diesen regelmäßig Erinnerungen an das Einnehmen ihrer Medikation, Abfragen über ihren derzeitigen Gesundheitszustand oder Vorschläge für privat durchzuführende Übungen geschickt werden [6]. Hierbei scheint es möglich zu sein, mobile Behandlungsformen zu entwickeln, welche eine ähnliche Effektivität aufweisen wie persönliche Behandlungen [12].

Von besonderem Interesse für das Feld der mHealth ist die Nutzung von Smartphones. Weltweit beläuft sich die Anzahl der Smartphone-Nutzer auf 3,2 Milliarden (Stand: 2019) [27], dies ist mehr als jeder dritte [48]. In einigen Ländern ist der Anteil noch bedeutend höher, so gibt es etwa in Deutschland rund 58 Millionen Nutzer von Smartphones (Stand: 2019) [51], was fast 70% der Bevölkerung entspricht [48]. Die Vorteile, welche die Nutzung dieser Geräte anstatt der Verwendung neuer Systeme bietet, liegen auf der Hand. Die Anschaffungskosten entfallen, die Patienten sind bereits mit der Steuerung ihrer Smartphones vertraut und benötigen keine komplexen Instruktionen und sie sind daran gewöhnt ihr Smartphone mit sich zu tragen was das Risiko verringert, dass sie vergessen das Gerät mit sich zu führen.

2.2.3 eHealth und Context Awareness

Es gibt einige Konzepte um mit der Hilfe von Kontextsensitivität bessere eHealth-Anwendungen zu ermöglichen.

Das LoCa project (A Location and Context-aware eHealth Infrastructure) dient dazu, die Überwachung von chronisch kranken Patenten (welche bisher häufig von Hand durchgeführt wird) zu automatisieren, mit den Ziel sowohl den Komfort des Patienten zu erhöhen als auch die Arbeitslast des behandelnden Arztes zu reduzieren. Hierfür werden die Patienten mit Sensorik ausgestattet und ein überwachendes Programm eingerichtet, welches Prozesse wie das Auslesen, Verarbeiten und Vergleichen von Daten sowie das Benachrichtigen von Patienten und Ärzten automatisch je nach Nutzerkontext durchführt [21].

Ein weiteres solches Konzept wäre die Idee der 'Smart Health', bei der Informationen von elektronischer Stadt-Infrastruktur und mHealth-Anwendungen verknüpft werden um die Gesundheitsversorgung der Stadtbewohner zu verbessern. Beispiele dafür wie das konkret aussehen kann sind etwa stationäre Sensoren, welche die Luftqualität in verschiedenen Stadtteilen messen und dann je nach deren Anfälligkeit und Position Warnungen an Personen in der Stadt schicken, oder eine Fitness-App welche automatisch Unfälle entdeckt und einen Krankenwagen ruft, welcher dank permanent gesammelten Verkehrsdaten sehr schnell zum Unfallsort kommen kann [43].

Auch in dem in Abschnitt 2.2.2 bereits genannten Bereich der mobilen Interventio-

nen kann der Kontext genutzt werden, um die Effizienz einer Behandlung zu steigern. Ein Beispiel hierfür wäre die SleepCare-App, welche bei Schlafproblemen im Allgemeinen und Schlaflosigkeit im Besonderen unterstützen soll. Die Anwendung sammelt während ihrer Benutzung Informationen über den Nutzer durch dessen Verwendung von Teilfunktionen der App, etwa eines Schlaftagebuchs oder eines Tools für Entspannungsübungen. Diese Informationen werden im späteren Verlauf verwendet, um dem Patienten bessere Empfehlungen auszusprechen und konkret an den noch vorhandenen Problemen zu arbeiten [54].

3 Anforderungen

Ziel dieses Projektes ist es, ein Programm zu erstellen, welches die Sensoren von Smartphones auslesen kann und in der Lage ist, auf der Basis dieser Informationen den Kontext des Smartphone-Benutzers zu bestimmen. Der Kontext soll dann an Anwendungsprogramme weitergegeben werden, welche diesen als Basis für die Entscheidungsfindung verwenden können. In diesem Kapitel wird dieses Ziel konkretisiert. Dies dient zum einen dazu, die Konzeption auf die Zielsetzung abzustimmen, und erlaubt zum anderen nach Fertigstellung des Projektes zu urteilen wie erfolgreich die Umsetzung war.

3.1 Funktionale Anforderungen

Funktionale Anforderungen beschreiben direkt, welche Aufgaben ein System erfüllen können soll. Es handelt sich im Grunde um eine Liste von Fähigkeiten, welche implementiert werden müssen [3] .

3.1.1 Sensorauswertung

Das Framework soll in der Lage sein, die folgenden Sensoren und Informationsquellen regelmäßig auszulesen, ihre Werte und deren Veränderung über einen längeren Zeitraum zu speichern, sowie diese für die Ermittlung der weiter unten genannten Arten von hochleveligem Kontext zu verwenden:

- Satellitennavigation
- Rotationssensor
- Beschleunigungssensor

- Magnetometer
- Umgebungslichtsensor

3.1.2 Ortserkennung

Das Framework soll in der Lage sein die aktuelle Position des Nutzers mit einer Liste von bekannten Orten abzugleichen und festzustellen, an welchem davon (wenn überhaupt) er sich aufhält.

Zusätzlich soll es über die Fähigkeit verfügen, aus über einen längeren Zeitraum hinweg (etwa ein Tag bis eine Woche) gesammelten Daten Schlüsse auf Orte mit Wichtigkeit für den Nutzer zu ziehen. Hierbei soll zum einen die Existenz und Position dieser Orte herausgefunden, zum anderen ein ungefährer Zweck festgestellt werden - zum Beispiel ’Zuhause’ oder ’Arbeit’.

3.1.3 Fortbewegungsart

Das Framework soll feststellen können, ob der Nutzer unterwegs ist und wenn ja, mit welchem Fortbewegungsmittel. Hierbei soll zwischen den folgenden Optionen unterschieden werden:

- Keine Bewegung
- Fußgänger
- Fahrrad
- Kraftfahrzeug
- Flugzeug

3.1.4 Geräteposition

Das Framework soll in der Lage sein festzustellen, wo sich das Smartphone in Relation zu seinem Benutzer befindet. Hierbei sollen drei generelle Situationen unterschieden werden:

- In aktiver Benutzung (Der Nutzer hält das Gerät in der Hand)
- Am Körper getragen (Der Nutzer hat das Gerät in einer Tasche/Rucksack)
- Nicht in Benutzung (Das Gerät befindet sich nicht in der Nähe des Nutzers)

3.1.5 Rückmeldung der Zuverlässigkeit

Sämtliche höherleveligen Kontextinformationen sollen von einer Abschätzung begleitet sein, welche aussagt wie zuverlässig diese Information ist beziehungweise mit welcher Wahrscheinlichkeit sie korrekt ist.

3.1.6 Kommunikation mit Anwendungsprogramm

Das Anwendungsprogramm soll in der Lage sein zu jedem Zeitpunkt den aktuellen Kontext-Zustand abzufragen. Es soll die Fähigkeit haben, einzustellen, in wie großen zeitlichen Abständen das Framework auf die Sensoren zugreift um neue Informationen zu erlangen und das eigene Wissen zu den Kontext zu aktualisieren. Es soll hierbei auch festlegen können, welche Sensoren und Auswertungsmodule verwendet werden, um den Energieverbrauch zu regulieren und die Privatsphäre des Nutzers zu sichern.

3.2 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen sind allgemeine Qualitätseigenschaften, die das Produkt einhalten soll. Häufig werden diese verwendet um Ressourcenbeschränkungen (wie zum Beispiel Geld/Zeit bei der Implementierung oder Rechenleistung/-Energieverbrauch während der Laufzeit) in den Anforderungen festzuhalten [3].

3.2.1 Performanz

Da das Framework parallel zu dem einbindenden Anwendungsprogramm aktiv sein wird, besteht auch ohne zeitaufwendige Berechnungen durch das Framework potentiell eine hohe Auslastung der Recheneinheiten des Smartphones. Zudem ist

nicht bekannt wie wichtig es für die Anwendung ist verzögerungsfrei zu arbeiten, es besteht also die Möglichkeit, dass dies eine hohe Priorität hat. Somit ist es wichtig für das Framework möglichst effizient mit Rechenleistung umzugehen. Es soll durch die Aktivität des Frameworks nicht zu Verzögerungen im Betriebsablauf des Smartphones kommen welche über das von durchschnittlichen Nutzer akzeptierte Maß hinaus gehen.

3.2.2 Energieeffizienz

Die häufige Verwendung von Sensoren, insbesondere des GPS, sorgt für einen hohen Stromverbrauch [24]. Auch in vorherigen Projekten zu Thema Kontext wurde eine übermäßige Belastung des Akkus als ein starkes Problem festgestellt [55]. Der Energieverbrauch dieses Frameworks soll so gering sein, dass der Akku eines durchschnittlichen Smartphones bei passiver Nutzung einer App welche es verwendet mindestens zehn Stunden hält. Dies stellt sicher, dass es möglich ist eine App über einen vollen Arbeitstag hinweg aktiv zu halten, vorausgesetzt man beginnt den Tag mit einem vollen Akku.

3.2.3 Modularität

Das Projekt soll nach Zweck geordnet in verschiedene Einzelmodule unterteilt werden, welche seperat ausgeführt und getestet werden können. Dies dient dazu, die Fehlersuche zu vereinfachen, da Probleme stets lokalisiert werden können, und vereinfacht die Nutzung, Wartung und Erweiterung, da für einen Programmierer nicht mehr relevant ist wie Module, die er nicht bearbeitet, intern funktionieren, sondern er nur die Schnittstelle kennen muss.

3.2.4 Erweiterbarkeit

Das Framework soll derart gestaltet werden, dass es möglichst einfach ist neue Module einzubinden. Dies gilt sowohl für die Integration neuer Sensoren in die Auswertung eines bereits bestehenden höherleveligen Kontext-Moduls, als auch für das Erschaffen neuer höherleveliger Kontext-Module.

3.2.5 Wartbarkeit

Das Projekt soll umfassend dokumentiert und sauber geschrieben sein, um Anpassungen durch Dritte möglichst einfach und erfolgreich zu gestalten.

3.2.6 Intuitivität der Schnittstellen

Das Framework soll so gestaltet werden, dass es Anwendungsentwicklern möglichst leicht fällt zu verstehen wie das Interface zwischen Framework und Anwendung funktioniert. Die Einbindung des Frameworks soll ohne viel Wissen über seine interne Funktionsweise möglich sein.

3.2.7 Zielplattformen

Das Projekt soll sowohl unter aktuellen Android-Versionen als auch unter aktuellen iOS-Versionen benutzbar sein, da ein Großteil der sich aktuell in Betrieb befindenden Smartphones (über 99 Prozent) eines dieser beiden Betriebssysteme verwendet [36].

4 Konzeption

In diesem Kapitel wird der geplante Aufbau des Projektes erörtert. Hierbei wird zuerst Ionic [29] vorgestellt, das System auf dem diese Arbeit basieren wird, bevor dann allgemeine Designentscheidungen erörtert und schließlich die einzelnen Bestandteile betrachtet werden.

4.1 Plattform-Auswahl und -Analyse

Um die Ausführbarkeit auf Android und iOS zu ermöglichen, wurde bereits relativ früh beschlossen für diese Arbeit ein Cross-Platform-Framework zu verwenden. Hierbei wurden einige Optionen in Betracht gezogen: Flutter, Xamarian und Ionic, welche zu den beliebtesten App-Entwicklungs-Frameworks zählen [39].

Flutter ist ein von Google entwickeltes Open-Source SDK, mit welchem Apps für eine Vielzahl von Betriebssystemen geschrieben werden können, so zum Beispiel Android und iOS, aber für Windows und Linux [19]. Der Code hierfür wird in Dart geschrieben, einer ebenfalls von Google entwickelten, objektorientierten Programmiersprache. Die fertigen Apps werden dann für die jeweilige Architektur kompiliert, sodass es sich hierbei um echte native Anwendungen und nicht interpretierten Code handelt. Es wurde sich schließlich gegen die Verwendung von Flutter entschieden, da die mitgelieferte Dokumentation nicht sehr einsteigerfreundlich ist und es aufgrund des geringen Alters des SDK (2019) wenige Anlaufstellen für Hilfestellung gibt [20].

Xamarian ist ebenfalls ein Open-Source SDK, welches von einer Tochterfirma von Microsoft entwickelt wurde. Die Zielplattformen sind hierbei Android und iOS sowie Windows und Mac, für welche der Code wie bei Flutter kompiliert und nativ ausgeführt werden kann [46]. Die zu verwendende Programmiersprache ist C#, welche

ebenfalls von Microsoft entwickelt wurde und populären objektorientierten Sprachen wie C++ und Java ähnelt [10]. Relevante Nachteile umfassen hier die geringe Größe der Community sowie den Zwang, Teile des Codes plattform-spezifisch schreiben zu müssen [46].

Es existiert von eine Vielzahl weiterer Cross-Platform-Frameworks, von denen auch einige für die Nutzung in dieser Arbeit in Betracht gezogen wurden, doch die Betrachtung aller davon würde den Rahmen dieses Kapitels übersteigen.

Es wurde schließlich beschlossen, dass dieses Projekt mit der Hilfe von Ionic umgesetzt werden soll. Ionic ist ein Open-Source-Frameworks, mit dem Apps für Android und iOS sowie Web Apps entwickelt werden können. Obwohl Ionic eigentlich auf den Aufbau des Frontends fokussiert ist, so stellt es trotzdem dank der einfachen Bedienung, dem bestehenden Vorwissen und des uniformen Zugriffs auf die Hardware verschiedener Plattformen eine gute Wahl für dieses Projekt dar. Ionic verfügt über die Möglichkeit, verschiedene JavaScript-basierte Frameworks zur Entwicklung des Backends zu verwenden, welche dem durch HTML und CSS beschriebenen grafischen Design Funktionalität verleihen. Hierbei stehen Angular und React zur Auswahl, sowie Vue, auch wenn die Unterstützung von letzterem sich momentan noch unter Entwicklung befindet [29]. Aus den zur Verfügung stehenden Backend-Frameworks wurde für dieses Projekt Angular gewählt, da dies am einfachsten zu verwenden zu sein schien und, da es die am längsten implementierte Variante ist, die größte Unterstützung für es besteht.

Zum Zugriff auf Hardwarefunktionalitäten wird bei Ionic Apache Cordova verwendet, ein Framework welches es erlaubt mit dem gleichen Code auf die nativen APIs verschiedener Plattformen zuzugreifen. Hierbei fungiert 'Ionic Native', eine Reihe von Community-geschriebenen Plugins, als eine Art Schnittstelle zwischen Ionic und Cordova. Es existieren Plugins für eine ganze Reihe von Features, unter anderem die für dieses Projekt ausgewählten Sensoren (GPS, Rotation, Beschleunigung, Magnetfeld, Umgebungslicht) [30].

4.2 Architektur

Die Funktionalität des in dieser Arbeit entwickelten Frameworks soll in einzelne, relativ unabhängige Module unterteilt werden, welche jeweils einen Teil der Daten-

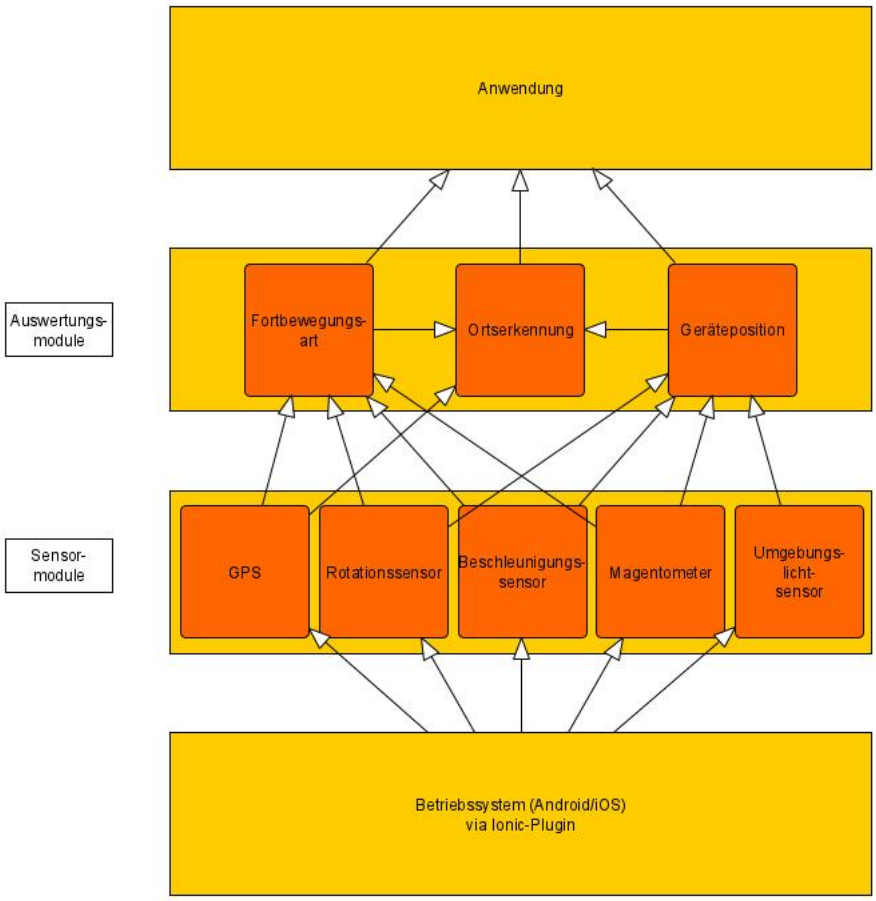

Abbildung 4.1: Architektur des zu entwickelnden Frameworks

sammlung und -auswertung erledigen können. Hierbei soll es zwei verschiedene Typen von Modulen geben: Sensormodule, welche in regelmäßigen Abständen physischen Sensoren abfragen und deren Informationen in verwendbare Form umwandeln, und Auswertungsmodule, welche die Daten von Sensormodulen und eventuell anderen Auswertungsmodulen verwenden um höherleveligen Kontext zu gewinnen.

4.3 Sensoren

Die verschiedenen Sensormodule sollen alle im Prinzip gleichartig funktionieren, nur dass sie Informationen von verschiedenen Quellen sammeln und organisieren. Diese Sensormodule werden über eine Datenbank von bisher gesammelten Infor-

mationen verfügen, da für viele Aufgaben mehrere Messwerte erforderlich sind. Neben Anfragen auf diese gesammelten Daten aus der Vergangenheit soll es auch möglich sein, frische Informationen anzufordern, wobei es die Option geben soll ein kleines Zeitintervall anzugeben, aus dem Werte gesammelt werden, um die Erkennung von Mustern in den Daten zu ermöglichen. Das permanente Auslesen von Sensorinformationen ist zu Energieaufwendig, als dass es sinnvoll wäre dies zu gestatten (siehe hierzu Abschnitt 3.2.2).

4.3.1 GPS

Das GPS (Global Positioning System) ist ein globales Navigationssatellitensystem mit dem die Position von Objekten überall auf der Erdoberfläche festgestellt werden kann. Dies funktioniert, indem die Signale von verschiedenen Satelliten empfangen werden und aus deren Position und der Übertragungszeit des Signals berechnet wird, wo man sich selbst befindet. Smartphones erreichen typischerweise eine Genauigkeit von etwa 5 Metern unter optimalen Bedingungen [44]. Da sich das GPS auf den Empfang von Satellitensignalen verlässt, kann es nur unter freiem Himmel mit maximaler Zuverlässigkeit eingesetzt werden, da das Durchdringen von Hindernissen wie zum Beispiel Wänden das Signal stören kann [25]. In diesem Projekt soll dieser Sensor zur Erfassung von zwei Arten von niedrigleveligem Kontext verwendet werden: Der Position und Geschwindigkeit des Nutzers.

4.3.2 Rotationssensor

Der Rotationssensor (beziehungsweise Gyroskop) erlaubt dem Smartphone, seine genaue Lage im Raum festzustellen. Dies funktioniert, indem ein rotierender Kreisel in ein bewegliches Lager platziert wird. Wenn das System rotiert, tritt eine Coriolis-kraft auf, über die Messung welcher diese Rotation quantifiziert werden kann [49]. In diesem Projekt soll mithilfe dieser Daten abgeschätzt werden, welchen Aktivitäten der Nutzer zu einem bestimmten Zeitpunkt nachgeht.

4.3.3 Beschleunigungssensor

Der Beschleunigungssensor (beziehungsweise Accelerometer) ist in der Lage, Kräfte zu messen welche das Smartphone beschleunigen. Hierbei können die X-, Y- und Z-Komponenten der Beschleunigung einzeln detektiert werden. Dies wird durch eine an Federn befestigten Masse realisiert, welche sich bei einer Beschleunigung des Gerätes innerhalb des Smartphones verschiebt, was mithilfe der sich verändernden Kapazität von Kondensatorplatten gemessen wird [7]. Der Beschleunigungssensor soll hier verwendet werden um die physischen Aktivitäten des Nutzers festzustellen.

4.3.4 Magnetometer

Das Magnetometer ist ein digitaler Kompass. Es benutzt das Magentfeld der Erde, um die Nordrichtung relativ zur Lage des Smartphones zu ermitteln [7]. Dieser Sensor soll ähnlich verwendet werden wie das Gyroskop, das heißt es soll anhand der Veränderung seiner Messungen ermittelt werden, welche physischen Bewegungen der Nutzer ausführt.

4.3.5 Umgebungslichtsensor

Smartphones verfügen für gewöhnlich über einen Umgebungslichtsensor, meist am oberen Rand in der Nähe der Frontkamera. Die Hauptaufgabe dieses Sensors ist es, das Beleuchtungslevel und die Farbtemperatur der Umgebung des Gerätes festzustellen, um die Helligkeit sowie Kontrast und Farbsättigung des Bildschirms entsprechend anzupassen, sollte der Nutzer dies wünschen [7]. In diesem Projekt soll diese Informationen jedoch für andere Zwecke verwenden, zum Beispiel als Indiz dafür, dass sich das Gerät in einer Tasche befindet.

4.4 Auswertung

Der Aufbau der Auswertungsmodule soll, wie schon bei den Sensormodulen, im Bezug auf Funktionsumfang und Interface recht gleichartig sein. Es soll möglich sein,

direkte Anfragen über den aktuellen Kontext an die Module zu stellen, wobei es möglich sein soll einzuschränken, welche Sensoren zur Ermittlung des Kontextes verwendet werden dürfen. Für alle drei Module soll ein naiver Bayes-Klassifikator verwendet werden, um von mit der Hilfe von Daten aus verschiedenen Quellen die Situation in eine der Kategorien einzuordnen. Dieser wurde bereits in früheren Projekten als zuverlässig bei geringem Ressourcenbedarf eingestuft [40]. Es handelt sich hierbei um eine Methode, ein Objekt der Klasse zuzuordnen, zu welcher es mit der höchsten Wahrscheinlichkeit gehört. Hierfür wird die Wahrscheinlichkeit, dass ein Objekt zu einer bestimmten Klasse gehört, ohne die Eigenschaften des Objektes zu kennen (die sogenannte A-priori-Wahrscheinlichkeit), multipliziert mit den Wahrscheinlichkeiten, dass ein beliebiges Objekt dieser Klasse die Eigenschaften des zu untersuchenden Objektes hat. Daraus resultiert eine Wertung für jede Klasse, aus welcher man dann diejenige mit der höchsten Wertung wählt [5].

4.4.1 Ortserkennung

Der erste Teil dieses Features, das Erkennen des Ortes an dem sich der Nutzer aufhält, ist relativ simpel realisierbar indem man durch die Liste der bekannten Orte iteriert, deren Koordinaten mit der aktuellen Position des Gerätes vergleicht und ausgibt wenn die Entfernung zu einem Ort unter einen bestimmten Schwellenwert fällt. Die Höhe des Schwellenwertes muss hierbei für jeden Ort eigens festgelegt werden, da einige zusammengehörige Regionen größer sind als andere - die Grundfläche eines Einfamilienhauses ist deutlich geringer als die einer Kaserne oder eines Firmengeländes, auch wenn beide als ein 'Ort' im Sinne dieses Features gelten können. In jedem Fall sollte der Schwellenwert höher sein als 5 Meter, da dies die durchschnittliche Unsicherheit von Smartphone-GPS ist [44].

Ein ähnliches Feature ist in MobiAR, einem AR (Augmented Reality) Touristenguide, zu finden. Das Ziel ist hierbei, den Nutzer über Ort mit Interesse für ihn in seinem Sichtfeld zu informieren, indem das Programm eine aktuelle Aufnahme der Frontkamera des Smartphones zeigt, in welcher über erkannte Bauwerke Informations-Icons gelegt werden. Durch die Nutzung von AR stehen MobiAR hier Optionen zur Ortsidentifizierung zur Verfügung, welche hier nicht verwendet werden können: Es verwendet das Magnetometer, um die Ausrichtung und somit das Sichtfeld des Nutzers herauszufinden, und vergleicht die Kameraaufnahmen mit gespeicherten

Bildern, um beobachtete Orte akkurater zuordnen zu können [33].

Der zweite Teil ist schwieriger umzusetzen. Zur Erkennung der Existenz von Orten mit Wichtigkeit muss eine stichprobenartige Menge an Punkten gespeichert werden, innerhalb welcher Teilmengen von nah beieinander liegenden Punkten detektiert werden können. Wie groß diese Teilmengen sein und wie nah die Punkte aufeinander liegen müssen wird durch Experimente festgelegt werden.

Die Bedeutung dieser Ort für den Nutzer wird notwendigerweise nur unpräzise festgestellt werden können. Im Rahmen dieser Arbeit beschränkt sich die Klassifikation auf zwei Optionen - 'Arbeit' und 'Zuhause'. Die Zuordnung soll sich hierbei an ein paar allgemeinen Indizien orientieren. Es wird erwartet, dass man auf der Arbeit etwa 6 bis 10 Stunden tagsüber verbringt. Dem gegenüber sind die meisten Personen etwa 11 bis 15 Stunden lang Zuhause [35], vor allem abends bis morgens, wovon sie ungefähr 8 Stunden mit Schlafen verbringen [15]. Diese Schlafzeit kann relativ einfach erkannt werden, da während ihr keine Interaktion mit dem Smartphone geschieht. Hierfür kann das Gerätepositionsmodul verwendet werden, wie in Abbildung 4.1 dargestellt. Anhand dieser Informationen sollte es möglich sein, den Arbeitsplatz und die eigene Wohnung zu identifizieren. Selbstverständlich funktioniert das Verfahren nur bei Lebensentwürfen, welche den genannten Parametern entsprechen - eine einzelne, lange Schlafzeit am Stück anstatt über den Tag verteilt zu schlafen, ein einzelner Wohnort, ein Beruf der an einen bestimmten Ort gebunden ist - aber die Unterstützung alternativer Lebensentwürfe geht über den Rahmen dieses Projektes hinaus.

4.4.2 Fortbewegungsart

MobiCon, eine Middleware zur Ermittlung von hochleveligem Kontext, verfügt über die Fähigkeit die physische Aktivität des Nutzers festzustellen, wobei zwischen Stehen, Sitzen, Gehen und Rennen unterschieden wird. Hierbei kann der Zustand 'gehen' sowohl über eine statistische Analyse der GPS-Daten als auch über eine Fourier-Analyse der Beschleunigungsdaten ermittelt werden [31].

In einer Arbeit von 2014 wurde zur Unterstützung eines Navigationssystems ebenfalls versucht, anhand von Kontextinformationen die Fortbewegungsart des Nutzers zu ermitteln. Hierbei wurde unterschieden zwischen Gehen, Rennen, Treppenstei-

gen, Autofahren und Aufzugfahren. Es wurde untersucht, wie die Zuverlässigkeit der Aussagen über diese Zustände von der Verwendung von verschiedenen Sensoren und Datenverarbeitungsmethoden abhängt. Diese Zuverlässigkeit wurde dann mit dem jeweiligen Energie- sowie Rechenleistungsbedarf der verschiedenen Methodiken verglichen um eine Kombination zu finden, welche eine hinreichende Ergebnisqualität bei dennoch akzeptablem Ressourcenverbrauch liefert. Die letztendlich favorisierte Kombination bei den Sensoren war das Accelerometer in Verbindung mit einem Orientierungs-Softsensor, welcher seine Daten aus dem Accelerometer, Magentometer und Gyroskop bezieht. Bei der Verarbeitungsmethodik, wo ein Naiver Bayes-Klassifikator, ein Bayessches Netz und ein Neuronales Netzwerk verglichen wurden, wurde festgestellt, dass während das Neuronale Netzwerk deutlich mehr Rechenleistung benötigt als die anderen Methoden, ihre Zuverlässigkeit ungefähr gleich war [40].

Zur Feststellung der Fortbewegungsart sind sowohl die aktuelle Geschwindigkeit als auch die Art, wie sich das Smartphone bewegt (detektiert durch Rotationssensor, Beschleunigungssensor und Magnetometer) relevant. Bei einer Geschwindigkeit nahe 0 wird davon ausgegangen, dass sich das Gerät nicht bewegt, und ab einer gewissen Geschwindigkeit (200 km/h oder 60 m/s) wird davon ausgegangen, dass sich das Gerät in einem Flugzeug befindet - andere Erklärungen sind zwar möglich (Raumschiff, sehr schnelles Auto), aber unwahrscheinlich [32]. Darunter kann die Geschwindigkeit jedoch nur als Indiz verwendet werden, da es zu große Überlappungen zwischen den möglichen Geschwindigkeiten der drei anderen Fortbewegungsmittel gibt [32]. An dieser Stelle können die restlichen drei Sensoren ins Spiel. Die Bewegungen, welche der Nutzer und damit das Gerät machen, sowie die Ausrichtung des Nutzers und somit Gerätes, hängen von der Fortbewegungsart ab [40]. Wenn es uns gelingt, diese gemessenen Bewegungen und die Ausrichtung einem von drei Mustern (Laufen, Fahrrad, Auto) zuzuordnen, dann kann diese Information zusammen mit der Geschwindigkeit für eine relativ zuverlässige Aussage über das Fortbewegungsmittel genutzt werden. Die genauen Parameter dieser Muster müssen experimentell ermittelt werden.

4.4.3 Geräteposition

In einer Arbeit von 2015 wurde bereits versucht, die Position des Smartphones am Körper anhand von Sensordaten festzustellen. Das Ziel hierbei war es, durch das Wissen über die Position des Smartphones am Körper Sensorwerte besser interpretieren zu können, so wurden verschiedene Arten von Stürzen etwa falsch klassifiziert wenn sich das Gerät nicht an der von der Anwendung erwarteten Stelle befand. Es wurden hierbei die Werte von Accelerometer, Gyroskop, Lichtsensor, Mikrophon, GPS und Magnetometer bei der Lage des Smartphones in sechs verschiedenen Positionen am Körper sowie auf dem Tisch untersucht. Es wurde ermittelt, dass Accelerometer, Gyroskop und Magnetometer in der Lage sind, recht präzise Aussagen über die Smartphoneposition zu tätigen. Der Lichtsensor ist in der Lage, zwischen bedeckten und unbedeckten Positionen zu unterscheiden - er kann allerdings nicht ermitteln, in welcher Position er sich genau befindet. Das GPS erlaubt keine relevanten Rückschlüsse auf die Position des Gerätes [2].

In dem in 4.4.2 erwähnten Navigationssystemsprojekt wurde ebenfalls versucht, die Position des Smartphones in Relation zum Nutzer zu ermitteln. Hierbei wurden deutlich mehr verschiedene Zustände unterschieden als in unserer Zielsetzung, so wurden zum Beispiel drei unterschiedliche Arten das Gerät in der Hand zu halten (baumelnd, lesend, neben dem Ohr) und fünf verschiedene Kleidungstücke (Hose, Gürtel, Handtasche, Rucksack, Jacke) differenziert [40]. Für die Methodik und Ergebnisse dieses Projektes siehe Abschnitt 4.4.2.

Wie weiter oben bereits festgestellt, sind für die Geräteposition vor allem die Sensoren, welche die Bewegung des Handys aufzeichnen (Rotationssensor, Beschleunigungssensor und Magnetometer) interessant. Auf eine Positionierung des Smartphones abseits des Körpers des Nutzers kann geschlossen werden, wenn sich das Gerät nicht bewegt und waagerecht auf einer Fläche liegt [2]. Die Möglichkeit, dass sich ein Smartphone auch ohne menschliche Einwirkung bewegt, besteht zwar (Erdbeben, in Fahrzeugen vergessen), sollte während der normalen Benutzung aber nur selten auftreten. Ebenfalls ist es möglich, dass das Gerät abseits des Körpers des Nutzers schräg an etwas angelehnt wird, aber auch dies sollte erfahrungsgemäß nicht häufig der Fall sein.

Zur Unterscheidung zwischen aktiver Nutzung und passivem Tragen wird eine größere Anzahl von Datenquellen nötig sein. Der Umgebungslichtsensor kann die Dun-

kelheit einer Hosentasche oder eines Rucksacks feststellen [2] - hierfür muss allerdings geprüft werden, ob die Nutzung eines Smartphones in relativer Dunkelheit, zum Beispiel Nachts und/oder in einem licht-isolierten Raum, nicht ähnliche Werte verursacht. Zudem folgen die Bewegungen des Gerätes bei aktiver Nutzung und passivem Tragen verschiedenen Mustern [2], welche mit den drei oben genannten Sensoren unterschieden werden können, wobei allerdings deren genaue Parameter noch ermittelt werden müssen.

5 Implementierung

In diesem Kapitel soll die Implementierung des Frameworks dokumentiert werden. Es soll also festgehalten werden, in welcher Reihenfolge was umgesetzt wurde, warum welche Entscheidungen getroffen wurden, welche Fragen dabei aufgekommen sind und welche Probleme sich gestellt haben, und wie diese beantwortet beziehungsweise gelöst wurden.

Die Funktionalität des Codes wird auf drei verschiedenen Plattformen getestet: Im Browser eines Computers (Windows 10 Pro N x64, Google Chrome Version 86.0.4240.193) sowie auf zwei verschiedenen Smartphones (Huawei WAS-LX1A, Android 8 sowie Fairphone 3, Android 10). Hierbei ist aufgrund der mangelnden Sensoren im Computer der Test an diesem nur beschränkt möglich. Tests an iPhones wären wünschenswert, sind aufgrund einem Mangel an entsprechender Hardware aber momentan leider nicht möglich.

5.1 Sensormodule

Am Anfang der Implementierung steht die initiale Umsetzung der Sensormodule. Da diese Module als relativ gleichartig entworfen wurden, ist es sinnhaft zunächst die übereinstimmenden Codeabschnitte in einer nicht direkt verwendeten Klasse zu schreiben, von welcher die einzelnen Module dann erben, um danach den spezifischen Code in den Modulen zu implementieren.

5.1.1 Elternklasse

Die Elternklasse (SensorModule genannt) verfügt über zwei öffentliche Funktionen (getCurrentData und getDataSequence), von denen eine wie in der Konzeption er-

klärt einmal Daten sammelt und die andere in regelmäßigen Abständen die erstere aufruft. Diese zweite Funktion ist in SensorModule schon vollständig implementiert, während die erstere leer bleibt, da die nötigen Schritte hier vom spezifischen Sensor abhängen. Weiterhin gibt es eine öffentliche Variable, welche ein Array der gesammelten Daten enthalten soll.

5.1.2 GPS

Die Implementierung des GPS-Moduls lief ohne größere Probleme ab.

Das verwendete Cordova-Plugin [58] übergibt eine Vielzahl von Daten, von denen für uns allerdings nur Längengrad, Breitengrad, Genauigkeit und Zeitpunkt der Messung sowie die aktuelle Geschwindigkeit relevant sind. Die restlichen Informationen - Höhe, Genauigkeit der Höhe und Bewegungsrichtung - werden verworfen. Von den verwendeten Werten ist die Geschwindigkeit allerdings ein optionaler Rückgabewert, welcher zudem oftmals unplausible Werte annimmt. Deshalb wird diese von unserem Modul anhand der Differenzen in Position und Zeit während den beiden letzten Messungen berechnet.

Zusätzlich zu den durch SensorModule vorgegebenen Funktionen stellt das GPS-Modul auch eine Funktion bereit, welche die Distanz zwischen zwei Koordinaten als Punkte auf einer Kugel berechnet. Diese wird zum einen intern zur Geschwindigkeitsberechnung verwendet, zum anderen wird diese Funktionalität später vom Ortserkennungs-Modul benötigt werden.

Dieses Modul liefert auf beiden Testsmartphones sowie am Computer plausible Ergebnisse.

5.1.3 Rotationssensor

Das verwendete Cordova-Plugin [52] übergibt den Zeitpunkt der Messung sowie die Rotation in alle drei Dimensionen. Diese Informationen werden unverändert abgespeichert.

Das Modul liefert auf dem Huawai zwar Werte, allerdings sind diese immer sehr klein, und verändern sich nur über lange Zeiträume. Auch starke Rotationen schei-

nen sie nicht zu beeinflussen. Da die Werte auf dem Fairphone plausibel sind, ging man hier ursprünglich von einem technischen Defekt im ersten Smartphone aus. Eine Messung mit einem externen Programm lieferte allerdings plausible Werte, somit ist ein technischer Defekt ausgeschlossen und es könnte sich um ein Kompatibilitätsproblem handeln. Eine mögliche Lösung wäre die Verwendung eines anderen Plugins. Versuche des Umstiegs auf alternative Plugins scheiterten jedoch. Als Konsequenz daraus wird für die Datensammlung ausschließlich das zweite Smartphone verwendet.

5.1.4 Beschleunigungssensor

Die Implementierung des Beschleunigungssensor-Moduls lief ohne größere Probleme ab.

Das verwendete Cordova-Plugin [61] übergibt den Zeitpunkt der Messung sowie die Beschleunigung in alle drei Dimensionen. Diese Informationen werden unverändert abgespeichert.

Dieses Modul liefert auf beiden Testsmartphones plausible Ergebnisse.

5.1.5 Magnetometer

Die Verwendung des Magnetometer-Plugins [60] gestaltete sich zu Beginn schwierig, da der Compiler nicht in der Lage war das in der Dokumentation angegebene Modul zu importieren. Es stellte sich heraus, dass der in der Dokumentation angegebene Pfad inkorrekt war. Sobald dieser Fehler korrigiert war, funktionierte auch das Magnetometer ohne weitere Schwierigkeiten.

Das Plugin übergibt die Stärke des gemessenen Magnetfeldes in alle drei Dimensionen (unveränderte Sensordaten), sowie ein in Software berechneten Wert, welcher die Gesamtstärke des Magnetfeldes angibt. Zusätzlich zu diesen Werten wird auch der Zeitpunkt des Messung gespeichert.

Dieses Modul liefert auf beiden Testsmartphones plausible Ergebnisse. Die Messungen korrelierten sehr genau mit der Ausrichtung des jeweiligen Gerätes.

5.1.6 Umgebungslichtsensor

Es gibt nur sehr wenige Ionic-Plugins, die Zugriff auf den Umgebungslichtsensor gestatten. Auch nach längerer Suche waren nur zwei auffindbar: cordova-plugin-lightsensor, welches nur das Abfragen des Umgebungslichtssensors gestattet [57] und cordova-plugin-sensors, welches Zugriff auf alle durch die Android-API bereitgestellten Sensoren gewährt [59]. Beide Plugins sind allerdings nur für Android implementiert, weshalb es an dieser Stelle andere keine Option zu geben scheint, als auf die Unterstützung des Umgebungslichtsensors für iOS zu verzichten. Es wurde schließlich zunächst beschlossen, das zweite der Plugins zu verwenden, da es besser dokumentiert ist und aufgrund der Breite seiner Möglichkeiten eventuell wiederverwendet werden kann.

Während dieses Modul auf dem Huawai anfangs einwandfrei funktionierte, liefert es auf dem Fairphone keinerlei Daten. Nach einer Weile lieferte das Plugin auch auf dem ersten Smartphone keine Daten mehr, obwohl der Code unverändert blieb. Nach der Untersuchung des Problems konnte der Fehler auf die Pluginfunktion eingegrenzt werden, welche den Sensor aktivieren sollte. Urprünglich hatte diese mit einer Erfolgsmeldung terminiert, im aktuellen Zustand beendet sie sich jedoch auch nach längerem Warten nicht und verhindert somit die weitere Ausführung des Programms. Der ursprünglich funktionale Zustand konnte auch mit vielfältigen Fehleranalysen und Tests nicht wiederhergestellt werden. Es liegt durchaus im Rahmen des Möglichen das Updates des Smartphone-Betriebsystems zu diesem Zustand beigetragen haben. Die Umstände verhinderten eine weitere Integration des Umgebungslichtsensors.

5.2 Datensammlung

Zur Konstruktion der Analysemodule werden Daten benötigt, aus denen die Regeln zur Erkennung von verschiedenen Situationen hergeleitet werden können. Aus diesem Grund wurde zunächst ein Hilfsprogramm entwickelt, welches Sensordaten einliest und in eine Datei speichert, um danach die verschiedenen Situationen nachzustellen und in ihnen die entsprechenden Daten zu sammeln. Hierbei besitzen diese Situationen häufig verschiedene Ausprägungen (im folgenden als Unter-

situationen bezeichnet), welche später in eine gemeinsame Kategorie eingeordnet werden sollen, deren Messwerte sich aber signifikant unterscheiden. Weiter unten wird aufgelistet, welche Situationen unterschieden werden (dies entspricht den Anforderungen aus Kapitel 3) und in welchen exakten Ausprägungen dieser für diese Messungen durchgeführt, sowie wie viele Daten von welchen Sensoren eingelesen werden. In jeder Ausprägung werden jeweils 10 Datenpakete gesammelt. Hierbei werden für die Ortserkennung keine Daten gemessen, da die Features dieses Moduls entweder keine Daten zur Implementierung brauchen oder es erst möglich ist diese Daten zu sammeln, wenn die anderen beiden Module fertig sind. Die relevanten Informationen, welche in dieser initialen Phase gespeichert und später zur Erkennung von Situationen verwendet werden, sind für das GPS die Geschwindigkeit, für das Accelerometer, Magnetometer und Gyroskop die Standardabweichung sowie der Mittelwert der Summe der Absolutwerte der Komponenten und für das Magnetometer zusätzlich die primäre Vektorrichtung. Diese meisten dieser Vergleichspunkte wurden bereits in früheren Projekten mit Erfolg verwendet [2].

5.2.1 Hilfsprogramm

Das Sammeln der Rohdaten wird durch Einbinden und Aufrufen der zuvor geschriebenen Sensormodule erreicht. Für die Berechnung der Standardabweichung sowie Mittelwerte werden neue Funktionen benötigt, welche um sie für die Analysemodule wiederverwenden zu können in eine Elternklasse (AnalysisModule genannt) ausgelagert werden. Für das Speichern der gewünschten Werte wird die Filesystem-API von Ionic verwendet. Hierbei ist die einzige Schwierigkeit, dass Android 10 standardmäßig keinen Zugriff auf öffentliche Teile des Dateisystems gewährt, und erst ein entsprechender Kompatibilitätsmodus aktiviert werden muss, bevor dies möglich ist.

5.2.2 Fortbewegungsart

Für die Ermittlung der Fortbewegungsart werden pro Datenblock zwei GPS-Messungen in einem Abstand von 10 Sekunden eingelesen, um eine möglichst akkurate Berechnung der Geschwindigkeit zu ermöglichen. Magnetometer, Rotationssensor und Beschleunigungssensor sollen mit 10 Hertz für fünf Sekunden abgefragt werden, da

repetitive Bewegungen wie Schritte nach dieser Zeit abgeschlossen sein sollten, und diese Frequenz hoch genug ist um diese zu entdecken [2].

Für die Erkennung des Laufens sollen folgende Untersituationen differenziert werden:

- Nutzer laufend, Smartphone in der Hosentasche
- Nutzer laufend, Smartphone in Rucksack
- Nutzer laufend, Smartphone in den Händen

Für die Erkennung des Fahrradfahrens sollen folgende Untersituationen differenziert werden:

- Nutzer auf Fahrrad, Smartphone in der Hosentasche
- Nutzer auf Fahrrad, Smartphone in Rucksack

Für die Erkennung der Kraftfahrzeugnutzung sollen folgende Untersituationen differenziert werden:

- Nutzer sitzend in Kraftfahrzeug, Smartphone in der Hosentasche
- Nutzer sitzend in Kraftfahrzeug, Smartphone in Rucksack

Daten über Flugzeuge können aus zeitlichen und finanziellen Gründen nicht gesammelt werden. Diese Option wird automatisch ab einer Geschwindigkeit von 60 m/s zurückgegeben und erfordert somit keine weiteren Informationen. Ebenso wird aus einer Geschwindigkeitsmessung von unter 0,1 m/s geschlossen, dass sich das Gerät nicht bewegt.

5.2.3 Geräteposition

Für die Ermittlung der Geräteposition sollen Magnetometer, Rotationssensor und Beschleunigungssensor wie bei der Fortbewegungsart mit 10 Hertz für fünf Sekunden abgefragt werden.

Für die Erkennung der aktiven Benutzung sollen folgende Untersituationen differenziert werden:

- Nutzer laufend, Smartphone in den Händen

- Nutzer sitzend, Smartphone in den Händen
- Nutzer liegend, Smartphone auf Möbelstück

Für die Erkennung des Tragens am Körper sollen folgende Untersituationen differenziert werden:

- Nutzer stehend, Smartphone in der Hosentasche
- Nutzer laufend, Smartphone in der Hosentasche
- Nutzer sitzend, Smartphone in der Hosentasche
- Nutzer liegend, Smartphone in der Hosentasche
- Nutzer stehend, Smartphone in Rucksack
- Nutzer laufend, Smartphone in Rucksack

Die Untersituationen, welche sowohl hier als auch bei den Messungen zur Fortbewegungsart auftreten, werden doppelt verwendet anstatt mehrere Messungen durchzuführen.

Für die Erkennung der Nichtbenutzung sollen folgende Untersituationen differenziert werden:

- Smartphone auf Möbelstück, Front abwärts
- Smartphone auf Möbelstück, Front aufwärts

5.2.4 Weitere Verarbeitung der gesammelten Daten

Die Messungen beinhalten einzelne Ausreißer, welche als Sensorstörung interpretiert und entfernt wurden.

Um den naiven Bayes-Klassifikator anzuwenden, müssen die Eingabewerte zunächst in diskrete Klassen unterteilt werden. Die genauen Parameter dieser Klassen sind hierbei frei wählbar, allerdings sorgen zu große Klassen dafür, dass die Daten nicht mehr verwendbar sind (da auch stark verschiedene Werte nicht mehr unterschieden werden), während zu kleine Klassen den Rechenaufwand erhöhen und größere Mengen an Trainingsdaten nötig machen (da die Klassen ansonsten

nicht entsprechend ihrer tatsächlichen Wahrscheinlichkeit gefüllt werden). Die getroffenen Wahlen für die Klassenparameter sind in der Tabelle 5.1 dargestellt.

In einem nächsten Schritt muss für jede Klasse/Situations-Kombination die Anzahl der zugehörigen Messwerte aus den Trainingsdaten gezählt werden, um die Wahrscheinlichkeit P(Messwert in Klasse x | Nutzer in Situation y) berechnen zu können, dass ein Messwert in einer bestimmten Klasse liegt, unter der Bedingung, dass sich der Nutzer in einer bestimmten Situation befindet. Hierbei wird zu jeder dieser Klasse/Situations-Kombinationen noch ein α (hier 1) addiert, um zu verhindern, dass P(Messwert in Klasse x | Nutzer in Situation y) zu 0 werden kann und somit einzelne unpassende Werte Situationen komplett ausschließen können. Die daraus resultierenden Wahrscheinlichkeiten sind in Tabelle 5.1 abgebildet.

In der folgenden Tabelle steht 'Mag' für 'Magnetometer', 'Acc' für 'Accelerometer', 'Gyr' für 'Gyroskop', 'Ben' für Benutzung, 'Get' für 'Getragen', 'Unb' für Unbenutzt, 'Mean' für den Mittelwert, 'SD' für die Standartabweichung und 'Ges' für 'Geschwindigkeit'. Die Mittelwerte und Standardabweichungen beziehen sich jeweils auf die Summe der Absolutwerte der Komponenten, während sich die Richtung des Magnetometers auf dessen im Absolutwert größte Messrichtung bezieht.

Tabelle 5.1: Wahrscheinlichkeiten für Messwerte in Abhängigkeit vom Kontext

		Laufen	Fahrrad	KFZ	Ben	Get	Unb
Mag Richtung	x	0,03	0,09	0,17	0,03	0,02	0,04
	y	0,33	0,04	0,13	0,33	0,48	0,04
	z	0,64	0,87	0,7	0,64	0,51	0,91
Mag Mean	<20	0,03	0,04	0,08	0,03	0,02	0,04
	>20 <40	0,03	0,04	0,19	0,03	0,02	0,42
	>40 <60	0,86	0,81	0,62	0,53	0,85	0,42
	>60 <80	0,03	0,04	0,04	0,08	0,09	0,04
	>80	0,03	0,04	0,04	0,19	0,02	0,04
	>100	0,03	0,04	0,04	0,14	0,02	0,04
Mag SD	<0,5	0,21	0,17	0,04	0,21	0,17	0,21
	>0,5 <0,75	0,18	0,58	0,42	0,44	0,39	0,71
	>0,75 <1	0,38	0,13	0,21	0,29	0,23	0,04
	>1	0,24	0,13	0,33	0,06	0,02	0,04
Gyr Mean	<0,1	0,03	0,04	0,7	0,42	0,46	0,91
	>0,1 <1	0,45	0,43	0,22	0,48	0,29	0,04
	>1	0,52	0,52	0,09	0,09	0,25	0,04
Gyr SD	<0,1	0,03	0,08	0,71	0,38	0,48	0,88
	>0,1 <0,5	0,32	0,33	0,17	0,41	0,2	0,04
	>0,5 <1	0,38	0,38	0,04	0,18	0,16	0,04
	>1	0,26	0,21	0,08	0,03	0,16	0,04
Acc Mean	<10	0,03	0,04	0,04	0,03	0,02	0,83
	>10 <12,5	0,03	0,08	0,42	0,32	0,36	0,08
	>12,5 <15	0,65	0,5	0,21	0,62	0,3	0,04
	>15	0,29	0,38	0,33	0,03	0,33	0,04
Acc SD	<0,1	0,03	0,04	0,19	0,06	0,3	0,77
	>0,1 <0,5	0,03	0,04	0,15	0,33	0,26	0,04
	>0,5 <1	0,03	0,15	0,27	0,19	0,02	0,04
	>1 <3	0,03	0,31	0,27	0,08	0,09	0,08
	>3 <6	0,36	0,42	0,08	0,31	0,05	0,04
	>6	0,53	0,04	0,04	0,03	0,29	0,04
Ges	<0,5	0,17	0,08	0,04			
	>0,5 <1	0,28	0,04	0,04			
	>1 <5	0,39	0,27	0,23			
	>5 <10	0,06	0,46	0,12			
	>10 <30	0,08	0,04	0,38			
	>30	0,03	0,12	0,19			

5.3 Analysemodule

Nachdem die Trainingsdaten gesammelt und in die entsprechende Form gebracht wurden, können die Analysemodule implementiert werden. Hierbei sind das Fortbewegungsartmodul und das Gerätepositionsmodul stark ähnlich, während das Ortserkennungsmodul recht einzigartig ist.

5.3.1 Ortserkennung

Die Implementierung des ersten Features des Ortserkennungsmoduls (das Vergleichen der aktuellen Position mit einer Liste an bekannten Orten) ist wie erwartungsgemäß entsprechend der Konzeption in Abschnitt 4.4.3 realisierbar.
Das zweite Feature (das Erkennen von Orten und deren Bedeutung) stellt hingegen eine größere Herausforderung dar. Für die Speicherung der zur Identifizierung eines Ortes relevanten Daten (Uhrzeit, Position, Geräteposition, Fortbewegungsart) wird eine neue Struktur definiert, sowie eine Funktion um diese mit den aktuellen Informationen zu füllen. Die Funktion zur Analyse dieser Daten unterteilt die übergebene Mengen erst in Teilmengen, indem zuerst alle Punkte in 30 Meter zu einem gewissen Ausgangspunkt zusammengefasst werden, bevor der Suchradius für jeden zur Teilmenge hinzugefügten Punkt erweitert wird und der Vorgang wiederholt wird, bis keine neuen Punkte hinzugefügt werden können. Danach wird ein neuer Ausgangspunkt gewählt, der Suchradius wieder auf 30 Meter beschränkt und der erste Schritt wiederholt. Dieser Prozess findet so lange statt, bis alle Punkte einer Teilmenge zugeteilt wurden (die kann auch nur ein Element enthalten). Danach wird für jede der Teilmengen, welche mehr als ein Element enthält, der Mittelpunkt und die Größe (größte Entfernung zwischen Mittelpunkt und einem Element der Teilmenge) berechnet sowie ein Hinweis zu seiner Bedeutung abgegeben. Hierbei erhält jede mögliche Bedeutung (momentan nur Zuhause und Arbeit) eine Wertung, welche für zu dieser Bedeutung passenden Werten gesteigert wird. Die Bedeutung mit der höchsten Wertung wird schließlich zusammen mit den Koordinaten und der Größe zurückgegeben.

5.3.2 Fortbewegungsart

Zur Ermittlung der Fortbewegungsart müssen als erstes Sensordaten gesammelt werden, dies kann durch einen Aufruf der entsprechenden Sensormodul-Funktionen erreicht werden (die Parameter sind hierbei die gleichen wie bei der Sammlung der Trainingsdaten, um die Vergleichbarkeit zu gewährleisten). Aus diesen Daten werden zunächst die relevanten Werte (Mittelwerte, Standartabweichungen etc.) berechnet, bevor die zugehörigen Klassen für den Naiven Bayes-Klassifikator ermittelt werden. Der Klassifikator-Algorithmus soll mehrmals verwendet werden, weshalb er sich in einer eigenen Klasse befindet.
Der Naive Bayes-Klassifikator benötigt zusätzlich zu den Wahrscheinlichkeiten für die Klassen in Abhängigkeit von der Situation, welche experimentell ermittelt wurden, auch die A-priori-Wahrscheinlichkeit für jede Situation. Da diese stark von den einzelnen Nutzern abhängig ist und keine statistischen Werte zur Verfügung standen, wurde hier davon ausgegangen, dass jede Situation mit der gleichen Wahrscheinlichkeit auftritt.
Die Situationen 'Keine Bewegung' und 'Flugzeug' werden nicht durch den Klassifikator ermittelt, sonder vorher abgefangen: Bei einer Geschwindigkeit von weniger als 0,1 m/s wird davon ausgegangen, dass sich das Gerät nicht bewegt, und bei einer Geschwindigkeit von über 60 m/s wird davon ausgegangen, dass sich das Gerät in einem Flugzeug befindet.

5.3.3 Geräteposition

Der grundlegende Aufbau des Gerätepositionsmoduls ist in großen Teilen identisch zum Fortbewegungsartmodul, weshalb die Implementierung an dieser Stelle nicht weiter ausgeführt wird.

6 Evaluation

In diesem Kapitel wird zunächst untersucht, inwiefern die in Kapitel 3 gestellten Anforderungen erfüllt wurden. Danach werden mögliche Gründe für Abweichungen bei der Situationserkennung gesucht, bevor versucht wird das in dieser Arbeit erschaffene Framework in einem praktischen Kontext zu verwenden.

6.1 Erfüllung der Anforderungen

Der Grad, zu dem seine Anforderungen erfüllt wurden, bestimmt den Erfolg eines Softwareprojektes. In diesem Abschnitt wird dokumentiert, wie weit die Anforderungen dieses Projektes erfüllt wurden.

6.1.1 Funktionale Anforderungen

Die Anforderungen an die Sensorauswertung wurden mit Ausnahme der Integration des Umgebungslichtsensors erfüllt, wie in Kapitel 5 erläutert wurde.
Die Identifizierung des aktuellen Ortes des Nutzers funktioniert ohne Einschränkungen, jedoch kann es durch Ungenauigkeiten des GPS zu Fehlern kommen. Die Erkennung von Orten mit Bedeutung sowie ihre Interpretation ist ebenfalls den Beobachtungen nach funktionsfähig, konnte jedoch aus Zeitmangel nicht so ausführlich wie die anderen beiden Module getestet werden.
Sowohl für die Fortbewegungsart als auch für die Geräteposition wurde für jede zu erkennende Situation 20 Mal getestet, ob das Programm in der Lage ist, diese korrekt zu identifizieren. Eine Ausnahme ist hierbei das Flugzeug, für welches wie bereits erwähnt keine Tests durchgeführt werden können.

Tabelle 6.1: Erfolgsrate der Erkennung von Kontexten im Fortbewegungsart-Modul

Fortbewegungsart	Keine	Laufen	Fahrrad	KFZ	Flugzeug
Erfolgsrate	100%	65%	40%	90%	Keine Daten

Die niedrige Erfolgsrate des Laufens liegt daran, dass bei niedrigen Geschwindigkeiten das GPS manchmal zwei sich zu nah beieinander befindliche Orte rückmeldet, was das System zu der Schlussfolgerung verleitet, dass keine Bewegung stattfindet, ohne die restlichen Sensoren überhaupt abzufragen. Die eigentliche Analyse der aller Messwerte, sofern diese stattfand, lieferte in 100% der Fälle das korrekte Ergebnis.
Das selbe gibt für die Fehlschläge bei der Erkennung der Nutzung eines Kraftfahrzeugs. Alle Fehler kamen zustande, als das Fahrzeug sich temporär nicht bewegte, zum Beispiel an Ampeln.
Das größte Problem bei der Erkennung des Fahrradfahrens war die Verwechslung mit dem Laufen. Hierbei scheint die Auswertung der Daten zu simpel zu sein um diese Situationen zuverlässig zu unterscheiden.

Tabelle 6.2: Erfolgsrate der Erkennung von Kontexten im Geräteposition-Modul

Geräteposition	Benutzung	Getragen	Unbenutzt
Erfolgsrate	80%	50%	100%

Hierbei fällt auf, dass die Erkennung des am Körper Tragens des Gerätes deutlich seltener erfolgreich ist als die Erkennung der anderen beiden Möglichkeiten. Eine Erklärung hierfür wäre, dass die Messwerte für das am Körper Tragen häufig zwischen denen der anderen Optionen liegt, was dazu führt, dass bei Abweichungen der Werte von der Norm, egal welcher Art, die Situation häufig falsch interpretiert wird. Auch in diesem Fall wäre eine komplexere Auswertung der Daten möglicherweise in der Lage, die Erfolgsrate zu verbessern.
Die Rückmeldung der Zuverlässigkeit und die Kommunikation mit Anwendungsprogramm wurden nach den Anforderungen umgesetzt.

6.1.2 Nichtfunktionale Anforderungen

Die Performanz des Frameworks ist zufriedenstellend. Es kommt zu keinen spürbaren Verzögerungen bei der Interaktion mit anderen Anwendungen während der Aktivität das Frameworks.
Der Energieverbrauch des Frameworks ist angemessen, die Erfüllung der Anforderung hängt jedoch vom Zustand des jeweiligen Akkus ab.
Das Frameworks ist modular gestaltet und erweiterbar.
Der Code des Frameworks ist durchgängig kommentiert, allerdings existiert keine

externe Dokumentation. Er ist in großen Teilen gut leserlich, allerdings sind in diesem Aspekt stellenweise noch starke Verbesserungen möglich.
Die Bedienbarkeit des Frameworks entspricht dem Standard von Ionic-Plugins.
Die Unterstützung des Frameworks für Android wurde erfolgreich getestet. Es sollte auch mit iOS kompatibel sein, allerdings konnte dies in Ermangelung eines iPhone nicht getestet werden.

6.2 Mögliche Fehlerquellen

Wie in den Tabellen 6.1 und 6.2 zu sehen ist, fällt es dem Programm äußerst schwer, einige Situationen korrekt zuzuordnen. In diesem Abschnitt werden mögliche Gründe für dieses Problem diskutiert.

6.2.1 Messfehler

Während die in Smartphones eingebauten Sensoren zwar als präzise gelten, so ist es immer möglich, dass ihre Messungen nicht komplett akkurat die Realität repräsentieren. Gerade bei GPS-Koordinaten, welche während einer Bewegung des Gerätes gemessen wurden, konnte festgestellt werden, dass es mitunter zu Abweichungen von über hundert Metern kommen kann.

6.2.2 Auswahl der Sensoren

Die in dieser Arbeit verwendeten Sensoren stellen nur einen kleinen Ausschnitt der Informationsquellen dar, welche einem Smartphone zur Verfügung stehen [7]. Während einige dieser Informationsquellen zweifellos wenig Aufschluss über die untersuchten Situation geben (wie etwa der Fingerabdruckscanner), könnten andere (zum Beispiel das Mikrophon) durchaus eine präzisere Zuordnung der Situation erlauben.

6.2.3 Verarbeitung der Rohdaten

Bei einer Messung von drei Sensoren bei 10 Hertz über einen Zeitraum von 5 Sekunden, wie sie in dieser Arbeit durchgeführt wird, entstehen 150 einzelne Messwerte, welche dann noch aus verschiedenen Komponenten bestehen - zu viele, um

diese ohne weitere Verarbeitung einem Klassifikatoralgorithmus zu übergeben. In dieser Arbeit wurde aus diesem Grund der Mittelwert und die Standardabweichung gebildet, weil dies wenig Rechenleistung erfordert und bereits in früheren Projekten erfolgreich verwendet wurde [2]. Allerdings gehen in diesem Schritt auch sehr viele charakteristische Informationen verloren. Andere Verarbeitungstechniken, wie etwa eine Fourier-Analyse [31], könnten hier zu mehr Präzision führen.

6.2.4 Klassifikatoralgorithmus

Der Naive Bayes-Klassifikator ist einfach zu implementieren und benötigt wenig Rechenleistung, ist aber dennoch recht zuverlässig [40]. Ein großes Problem dieses Algorithmus ist jedoch die recht willkürliche Wahl der Klassen der Eingabedaten. Während zwar versucht wurde, diese so zu gestalten, dass möglichst große Unterschiede zwischen den verschiedenen Ausgabekategorien bestehen, ist dies möglicherweise nicht immer gelungen. Zudem geht der Naive Bayes-Klassifikator davon aus, dass alle Eingabedaten unabhängig voneinander sind - eine Annahme, die in dieser Situation offensichtlich nicht den Tatsachen entspricht. Ein komplexerer Klassifikator-Algorithmus, wie zu Beispiel ein Bayes-Netz (welches ähnlich wie der Naive Bayes-Klassifikator funktioniert, nur dass Relationen zwischen verschiedenen Eingabedaten dargestellt werden können) oder ein Neuronales Netzwerk, könnte eventuell eine höhere Zuverlässigkeit erreichen.

6.2.5 Trainingsdaten

Trotz der Versuche, Daten aus möglichst vielen Ausprägungen einer zu erkennenden Situation zu sammeln, ist definitiv nur ein kleiner Anteil dieser möglichen Ausprägungen abgedeckt. Sollte sich das Gerät also in einer Situation befinden, welche zwar zu einer der möglichen Ausgabekategorien passt, aber sich von allen Trainingsdaten unterscheidet, so ist es wahrscheinlich, dass das Programm diese falsch zuordnet.

6.3 Anwendungsbeispiel

Um die praktischen Anwendungsmöglichkeiten des Frameworks zu demonstrieren und auszutesten soll noch eine prototypische E-Health-Anwendung geplant und

umgesetzt werden. Das Ziel der Anwendung soll es sein, den Nutzer zu möglichst opportunen Zeitpunkten an das Einnehmen von Medikation zu erinnern.
Hierfür soll es möglich sein, Zeitpunkte für die Einnahme der Medikamente sowie optional die Orte, an denen diese eingenommen werden sollen, anzugeben (falls die Medikamente an einem bestimmten Ort gelagert werden und der Nutzer sie deshalb nur dort zu sich nehmen kann). Die Anwendung soll dann eine Benachrichtigung senden, sobald der angegebene Zeitpunkt verstrichen ist, der Nutzer sich am angegebenen Ort befindet (falls ein Ort angegeben wurde), er nicht Kraftfahrzeug oder Fahrrad fährt und er das Smartphone entweder am Körper trägt oder gerade benutzt (da ihn diese Zustände am Bemerken der Nachricht hindern könnten).
Für die Eingabe wird ein einfaches Formular verwendet, welches den Namen des Medikamentes, die Uhrzeit (als Stunde und Minute) sowie die Position und Größe des Ortes entgegennimmt. Zur Vereinfachung der Eingabe wird ein Knopf integriert, welcher die Koordinaten der aktuellen Position in das Formular einfügt. Das Formular kann per Knopfdruck abgeschickt werden, was es einer internen Liste hinzufügt und in der App anzeigt. Hierbei wird nur dann ein Ort vermerkt, wenn alle drei nötigen Stellen ausgefüllt sind (Längengrad, Breitengrad, Größe des Ortes).
In regelmäßigen Abständen (im Test eine Minute) werden die drei Analysemodule aufgerufen und es wird überprüft, ob sich das Smartphone am Körper oder in Benutzung befindet sowie ob der Nutzer gerade nicht oder zu Fuß unterwegs ist. Sollten diese Bedingungen erfüllt sein, wird für jedes Medikament überprüft, ob sein Einnahmezeitpunkt überschritten ist und ob sich der Nutzer an seinem Einnahmeort befindet, sollte ein solcher vermerkt sein. Falls dies zutrifft, vibriert das Gerät, zeigt eine Meldung an welche an die Einnahme des Medikamentes erinnert, und schaltet das Medikament bis zum nächsten Tag inaktiv.

7 Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war es, eine Middleware zur Kontextermittlung für Smartphones zu implementieren, welche insbesondere für kontextsensitive eHealth-Anwendungen verwendet werden kann.
Hierfür wurde sich zunächst mit den Begriffen des mobilen Kontextes und der eHealth sowie verwandten Themen wie dem Mobile Computing, der Sensor Fusion und der mHealth auseinandergesetzt. Es wurden Definitionen von Kontext und Kontextsensitivität herausgearbeitet und existierende Projekte in den Feldern der Middleware für kontextsensitive Anwendungen und der kontextsensitiven eHealth-Anwendungen vorgestellt.
Danach wurden Anforderungen an diese zu entwickelnde Middleware erarbeitet, eine Architektur entwickelt und die Umsetzung der Software geplant, bevor diese auch durchgeführt wurde. Schließlich wurde die fertige Software auf die Erfüllung der Anforderungen überprüft, mögliche Fehlerquellen ermittelt und letztendlich eine prototypische eHealth-Anwendung auf der Basis der Middleware umgesetzt.

7.1 Ergebnis

Das Ziel der Bachelorarbeit wurde im Allgemeinen erreicht. Es ist gelungen, über die Nutzung von GPS, Accelerometer, Magnetometer und Rotationssensor festzustellen, an welchem Ort sich das Gerät befindet, auf welche Art, wenn überhaupt, sich der Nutzer fortbewegt, sowie inwiefern es von seinem Nutzer verwendet wird. Der Umgebungslichtsensor konnte leider nicht genutzt werden. Die Erfolgsrate ist hierbei im ersten dieser Ziele zufriedenstellend, für anderen beiden jedoch je nach Situation starken Schwankungen unterzogen. Stillstand sowie die Nutzung eines Kraftfahrzeugs konnten beinahe immer korrekt identifiziert werden, Laufen in etwa zwei Drittel und Fahrradfahren nur in etwas mehr als einem Drittel der Fälle. Hierbei wurde im Fall des Laufens die Ungenauigkeit des GPS-Systems als Fehlerquel-

le identifiziert, welche umgangen werden könnte, indem über längere Zeiträume gemessen oder den Koordinaten weniger Bedeutung beigemessen wird. Bei der Ermittlung der Position des Gerätes relativ zum Nutzer konnten die aktive Nutzung und die Abwesenheit von Nutzung meistens korrekt identifiziert werden. Die Identifikation des Tragens des Gerätes am Körper war hingegen nur in der Hälfte der Fälle erfolgreich. Der Mangel an Erfolg an dieser Stelle sowie bei der Identifikation des Fahrradfahrens scheint strukturelle Gründe zu haben, welche nur über eine Steigerung der Komplexität des Systems überwunden werden könnten.

7.2 Ausblick

Es besteht eine Vielzahl an Möglichkeiten, wie die Arbeit an diesem Projekt fortgeführt werden könnte. Die bestehenden Analysemodule könnten weitere Ausgabewerte erhalten, welche ihre jeweiligen Arten von Kontext präziser identifizieren - so könnte zum Beispiel versucht werden, zu erkennen, wo genau sich das Gerät am Körper des Nutzers befindet, oder eine Unterscheidung zwischen öffentlichem und privaten Kraftfahrzeug treffen. Die Genauigkeit der bestehenden Analysemodule könnte verbessert werden, indem die Parameter des Naiven Bayes-Klassifikators durch die Integration eine höheren Anzahl an Trainingsdaten optimiert werden. Eine weitere Verbesserungsmöglichkeit wäre es, den Naiven Bayes-Klassifikator durch einen komplexeren Algorithmus zu ersetzen, welcher die realen Zusammenhänge besser repräsentieren können, wie zum Beispiel ein Bayes-Netz oder ein Neuronales Netzwerk. Zudem könnte man weitere Sensormodule hinzufügen, um die Präzision der Analysemodule zu verbessern. Optionen wären hier die Nutzung des Barometers, Näherungssensors oder Mikrophons. Eine letzte Möglichkeit wäre es, zusätzliche Analysemodule implementieren, um weitere Arten hochleveligen Kontextes erkennen zu können. Hierbei könnten zum Beispiel die Körperhaltung des Nutzers untersucht oder Gespräche erkannt werden.

Literatur

[1] Daniel Abowd u. a. „Context-awareness in wearable and ubiquitous computing“. In: *Virtual Reality* 3.3 (1998), S. 200–211.

[2] Khaled Alanezi und Shivakant Mishra. „Design, implementation and evaluation of a smartphone position discovery service for accurate context sensing“. In: *Computers & Electrical Engineering* 44 (2015), S. 307 –323. ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2015.01.015. URL: http://www.sciencedirect.com/science/article/pii/S004579061500021X.

[3] *Anforderung (Informatik)*. URL: https://de.wikipedia.org/wiki/Anforderung_(Informatik). (Zugriff: 23.09.2020).

[4] Tim Aschermann. *Wie funktioniert ein Schrittzähler im Handy? Einfach erklärt*. URL: https://praxistipps.chip.de/wie-funktioniert-ein-schrittzaehler-im-handy-einfach-erklaert_50573. (Zugriff: 15.10.2020).

[5] *Bayes-Klassifikator*. URL: https://de.wikipedia.org/wiki/Bayes-Klassifikator. (Zugriff: 29.11.2020).

[6] Dror Ben-Zeev u. a. „Strategies for mHealth Research: Lessons from 3 Mobile Intervention Studies“. In: *Administration and Policy in Mental Health and Mental Health Services Research* 42.2 (1. März 2015), S. 157–167. ISSN: 1573-3289. DOI: 10.1007/s10488-014-0556-2. URL: https://doi.org/10.1007/s10488-014-0556-2.

[7] Olena Bochkor. *Sensoren in modernen Smartphones im Überblick*. URL: https://entwickler.de/online/mobile/sensoren-smartphones-ueberblick-579936038.html. (Zugriff: 26.09.2020).

[8] Mathias Brandt. *Hier nutzen die Deutschen ihr Smartphone*. URL: https://de.statista.com/infografik/8070/wo-die-deutschen-ihr-smartphone-nutzen/. (Zugriff: 23.07.2020).

[9] S. E. Butner und M. Ghodoussi. „A real-time system for tele-surgery“. In: *Proceedings 21st International Conference on Distributed Computing Systems*. 2001, S. 236–243.

[10] *C-Sharp*. URL: https://de.wikipedia.org/wiki/C-Sharp. (Zugriff: 30.10.2020).

[11] James L Crowley und Yves Demazeau. „Principles and Techniques for Sensor Data Fusion“. In:

[12] Pim Cuijpers u. a. „Internet and mobile interventions for depression: Opportunities and challenges“. In: *Depression and Anxiety* 34.7 (2017), S. 596–602. DOI: 10.1002/da.22641. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/da.22641. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/da.22641.

[13] B. V. Dasarathy. „Sensor fusion potential exploitation-innovative architectures and illustrative applications“. In: *Proceedings of the IEEE* 85.1 (1997), S. 24–38.

[14] Tamara Denning u. a. „Patients, Pacemakers, and Implantable Defibrillators: Human Values and Security for Wireless Implantable Medical Devices“. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: Association for Computing Machinery, 2010, 917–926. ISBN: 9781605589299. DOI: 10.1145/1753326.1753462. URL: https://doi.org/10.1145/1753326.1753462.

[15] Statista Research Department. *Schlafzeit pro Tag*. URL: https://de.statista.com/statistik/daten/studie/36749/umfrage/schlafzeit-pro-tag-in-den-oecd-laendern-im-jahr-2006/. (Zugriff: 07.10.2020).

[16] Anind K. Dey. „Understanding and Using Context“. In: *Personal and Ubiquitous Computing* 5.1 (1. Feb. 2001), S. 4–7. ISSN: 1617-4909. DOI: 10.1007/s007790170019. URL: https://doi.org/10.1007/s007790170019.

[17] Patrick Fahy und Siobhán Clarke. „CASS-Middleware for Mobile Context-Aware Applications“. In: *MobiSys* (Jan. 2004).

[18] Dr. Denzil Ferreira. *AWARE - Open-source Context Instrumentation Framework For Everyone*. URL: https://awareframework.com/. (Zugriff: 02.09.2020).

[19] *Flutter (Software)*. URL: https://de.wikipedia.org/wiki/Flutter_(Software). (Zugriff: 30.10.2020).

[20] *Flutter – der Hype aus der Sicht eines Nicht-Entwicklers*. URL: https://www.new-communication.de/neues/detail/flutter-der-hype-aus-der-sicht-eines-nicht-entwicklers/. (Zugriff: 30.10.2020).

[21] N. Fröhlich u. a. „LoCa - Towards a Context-aware Infrastructure for eHealth Applications“. In: (Jan. 2009), S. 52–57.

[22] *Fused Location Provider API*. URL: https://developers.google.com/location-context/fused-location-provider. (Zugriff: 16.10.2020).

[23] Manish J. Gajjar. *Mobile Sensors and Context-Aware Computing*. Morgan Kaufmann Publishers, 2017. ISBN: 9780128016602.

[24] *Geolocation)*. URL: https://capacitorjs.com/docs/apis/geolocation. (Zugriff: 24.09.2020).

[25] *Global Positioning System*. URL: https://de.wikipedia.org/wiki/Global_Positioning_System. (Zugriff: 13.09.2020).

[26] *Google Maps*. URL: https://www.google.de/maps/. (Zugriff: 29.11.2020).

[27] Tianyi Gu. *Newzoo's Global Mobile Market Report: Insights into the World's 3.2 Billion Smartphone Users, the Devices They Use & the Mobile Games They Play*. URL: https://newzoo.com/insights/articles/newzoos-global-mobile-market-report-insights-into-the-worlds-3-2-billion-smartphone-users-the-devices-they-use-the-mobile-games-they-play/. (Zugriff: 31.08.2020).

[28] A. Holz u. a. „Datenbrillen am Arbeitsplatz“. In: *Zentralblatt für Arbeitsmedizin, Arbeitsschutz und Ergonomie* (2. Apr. 2020). ISSN: 2198-0713. DOI: 10.1007/s40664-020-00394-7. URL: https://doi.org/10.1007/s40664-020-00394-7.

[29] *Ionic Framework*. URL: https://ionicframework.com/docs. (Zugriff: 28.10.2020).

[30] *Ionic Native Community*. URL: https://ionicframework.com/docs/native/community. (Zugriff: 28.10.2020).

[31] Y. Lee u. a. „MobiCon: Mobile context monitoring platform: Incorporating context-awareness to smartphone-centric personal sensor networks“. In: *2012 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*. 2012, S. 109–111. DOI: 10.1109/SECON.2012.6275765.

[32] *Liste der Geschwindigkeitsrekorde*. URL: https://de.wikipedia.org/wiki/Liste_der_Geschwindigkeitsrekorde. (Zugriff: 15.10.2020).

[33] David Marimon u. a. „MobiAR: Tourist Experiences through Mobile Augmented Reality“. In: *Information and communication technologies in tourism* (Jan. 2010).

[34] Prof. Dr. David Matusiewicz. *Mobile Health*. URL: https://wirtschaftslexikon.gabler.de/definition/mobile-health-54125. (Zugriff: 14.10.2020).

[35] Uschi Mayer. *Aktuelle Studie: Wir verbringen mehr Zeit zuhause*. URL: https://www.ots.at/presseaussendung/OTS_20190503_OTS0048/aktuelle-studie-wir-verbringen-mehr-zeit-zuhause#:~:text=Im%20Schnitt%20verbringen%20die%202018,eigenen%20vier%20W%C3%A4nden%20anzutreffen%20ist.. (Zugriff: 07.10.2020).

[36] *Mobile Operating System Market Share Worldwide*. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide. (Zugriff: 28.10.2020).

[37] Computer History Museum. *Timeline of Computer History*. URL: https://www.computerhistory.org/timeline/computers/. (Zugriff: 15.10.2020).

[38] *Peripheriegerät*. URL: https://de.wikipedia.org/wiki/Peripherieger%C3%A4t. (Zugriff: 15.10.2020).

[39] Mehul Rajput. *Top Mobile App Development Frameworks in 2019-2020*. URL: https://www.mindinventory.com/blog/mobile-app-development-framework-2019/. (Zugriff: 29.10.2020).

[40] Sara Saeedi, Adel Moussa und Naser El-Sheimy. „Context-Aware Personal Navigation Using Embedded Sensor Fusion in Smartphones“. In: *Sensors (Basel, Switzerland)* 14 (Apr. 2014), S. 5742–67. DOI: 10.3390/s140405742.

[41] B. N. Schilit und M. M. Theimer. „Disseminating active map information to mobile hosts“. In: *IEEE Network* 8.5 (1994), S. 22–32.

[42] W. Schwinger u. a. „Context-awareness in Mobile Tourism Guides - A Comprehensive Survey“. In: (Jan. 2005).

[43] A. Solanas u. a. „Smart health: A context-aware health paradigm within smart cities“. In: *IEEE Communications Magazine* 52.8 (2014), S. 74–81.

[44] Navigation National Coordination Office for Space-Based Positioning und Timing.

[45] Department for Media Tate Modern und Audiences. *Tate Modern Multimedia Tour*. URL: https://www.tate.org.uk/about-us/projects/tate-modern-multimedia-tour. (Zugriff: 24.07.2020).

[46] *The Good and The Bad of Xamarin Mobile Development*. URL: https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/. (Zugriff: 30.10.2020).

[47] World Health Organization eHealth Unit. *eHealth at WHO*. URL: https://www.who.int/ehealth/about/en/. (Zugriff: 29.06.2020).

[48] Department for Economic United Nations und Social Affairs. *World Population Prospects 2019*. United Nations, Department for Economic und Social Affairs, 2019. ISBN: 978-92-1-148317-8.

[49] *Vibrating structure gyroscope*.

[50] Till Hänisch Volker P. Andelfinger. *eHealth - Wie Smartphones, Apps und Wearables die Gesundheitsversorgung verändern werden*. Springer Gabler, 2016. ISBN: 9783658122386.

[51] VuMA. *Anzahl der Smartphone-Nutzer in Deutschland in den Jahren 2009 bis 2019*. URL: https://de.statista.com/statistik/daten/studie/198959/umfrage/anzahl-der-smartphonenutzer-in-deutschland-seit-2010/#:~:text=Die%20Anzahl%20der%20Smartphone%2DNutzer,Teil%20des%20allt%C3%A4glichen%20Lebens%20geworden.. (Zugriff: 31.08.2020).

[52] Jason Yang. *Cordova Gyroscope Plugin*. URL: https://github.com/NeoLSN/cordova-plugin-gyroscope. (Zugriff: 13.11.2020).

[53] Ji Soo Yi u. a. „Context awareness via a single device-attached accelerometer during mobile computing“. In: Jan. 2005, S. 303–306. DOI: 10.1145/1085777.1085839.

[54] Griffioen-Both et al. „Testing for Mobile E-Health Interventions“. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. SAC '16. Pisa, Italy: Association for Computing Machinery, 2016, 137–142. ISBN: 9781450337397. DOI: 10.1145/2851613.2851686. URL: https://doi.org/10.1145/2851613.2851686.

[55] Özgür Yürür et al. „Context-Awareness for Mobile Sensing: A Survey and Future Directions“. In: *IEEE COMMUNICATION SURVEYS AND TUTORIALS* 18.1 (2016), S. 68–93.

[56] denzilferreira. *AWARE: Google Activity Recognition*. URL: https://github.com/denzilferreira/com.aware.plugin.google.activity_recognition. (Zugriff: 15.10.2020).

[57] deshanktd. *cordova-plugin-lightsensor*. URL: https://www.npmjs.com/package/cordova-plugin-lightsensor. (Zugriff: 10.11.2020).

[58] erisu. *cordova-plugin-geolocation*. URL: https://github.com/apache/cordova-plugin-geolocation. (Zugriff: 10.11.2020).

[59] fabiorogeriosj. *Cordova Sensors Plugin*. URL: https://github.com/fabiorogeriosj/cordova-plugin-sensors. (Zugriff: 10.11.2020).

[60] sdesalas. *cordova-plugin-magnetometer*. URL: https://github.com/sdesalas/cordova-plugin-magnetometer. (Zugriff: 10.11.2020).

[61] timbru31. *cordova-plugin-device-motion*. URL: https://github.com/apache/cordova-plugin-device-motion. (Zugriff: 10.11.2020).

Name: Peter Hösch Matrikelnummer: 856162

Erklärung

Ich erkläre, dass ich die Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den .....

Peter Hösch