Institut für Visualisierung und Interaktive Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3611

Entwicklung einer Architektur für das Betriebssystem von intelligenter Kleidung

Tobias Birmili

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr. Niels Henze
Betreuer/in:	Stefan Schneegaß M.Sc.
Beginn am:	2. Januar 2014
Beendet am:	3. Juli 2014
CR-Nummer:	D.2.6, D.2.11

Kurzfassung

Intelligente Kleidung wurde in der Vergangenheit häufig prototypisch, in niedrigen Stückzahlen und für einen bestimmten Zweck entwickelt. Diese Arbeit definiert Schnittstellen und eine Architektur um zukünftige Entwicklungen modularer zu machen und den Entwicklungsprozess zu unterstützen.

Neben der Betrachtung der kompletten Systemarchitektur liegt der Fokus auf den Schichten die zwischen dem Computer und der Anwendung für Endbenutzer liegt. Zielgruppe sind dabei Entwickler von Sensoren, Erkennungsalgorithmen und Anwendungen. Für diese Zielgruppen werden Werkzeuge zur Verfügung gestellt, die den Entwicklungsprozess verbessern sollen.

Die Architektur wurde beispielhaft implementiert. Dazu wurden durchgehend Webtechnologien wie HTML, JavaScript und WebSockets eingesetzt. Um existierende Sensoren als Eingabe zu benutzen wurde dieser über eine Brücke mit Bluetooth verbunden. In einer Benutzerstudie wurde das implementierte System evaluiert. Es wurden dadurch einige Verbesserungsvorschläge erkannt die zukünftig integriert werden können.

Abstract

Intelligent clothes were usually produced prototypically, in low amounts and only for one specific purpose. This work defines interfaces and an architecture to modularise future development and support the development process.

Aside of looking at the whole system architecture, the focus lies on the layers between the computer hardware and the end-user application. The audience are developers of sensors, recognition algorithms and software developers. For those groups, different tools are integrated to optimise the whole development process.

The architecture was implemented exemplary using web technologies as HTML, JavaScript and WebSockets. Existing sensors were supported by a bluetooth bridge. The implementation was evaluated with a practical study. Some improvement suggestions were identified which can be implemented in the future.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

> In Zukunft könnte es Computer geben, die weniger als 1½ Tonnen wiegen.
>
> *(Popular Mechanics, 1949)*

Schon seit einigen Jahren werden Computer zunehmend mobiler. ENIAC, der erste elektronische Universalrechner, beanspruchte circa 167m^2 an Stellfläche und wog 27 Tonnen. Zur Programmierung wurde die Verkabelung von einzelnen Komponenten und Drehschalter benutzt. Nach und nach wurden die Computer immer persönlicher, kleiner und hielten Einzug in private Haushalte, zunächst auf dem Schreibtisch, dann auch auf dem Schoß. Heutzutage sind sie aus unseren Hosentaschen nicht mehr wegzudenken und fangen langsam an sich auf tragbare Gegenstände wie Uhren und Brillen auszubreiten. Google Glass, ein Rechner der sich in die Brille integriert, wiegt nur noch 50 Gramm und nimmt durch die Kombination mit einem Alltagsgegenstand kaum mehr Platz ein.

Jede große Veränderung des Formfaktors brachte bisher auch neue Paradigmen zur Benutzungsschnittstelle und Programmierschnittstelle mit sich. Heutzutage ist es viel mehr Menschen möglich, Software zu schreiben und benutzen. Diese Arbeit schaut in die Zukunft und versucht eine Architektur zu definieren, die vor allem Anwendungsentwicklung für mobile Einsatzzwecke abdeckt und dabei auch intelligente Kleidung integriert; Kleidung die als Datenquelle oder Interaktionsraum von Anwendungen dient.

Es gab und gibt viele Forschungsprojekte, die sich mit der Erforschung neuer Sensoren und Anwendungsfällen von intelligenter Kleidung beschäftigen. Einige Beispiele dafür sind Aktivitätserkennung über verschiedene Sensoren an Hosen [VC00], Erkennung der Körperhaltung anhand Falten in der Kleidung [HAT10] oder in flexible in Kleidung gewebte Sensoren [PLTP06].

SimpleSkin[1] ist ein EU-Projekt, welches das Ziel hat, intelligente Kleidung zu entwickeln um mit ihr Interaktion, die Überwachung von physiologischen Daten und Aktivitätserkennung, zu realisieren. Die Sensoren sollen dabei direkt im Stoff der Kleidung integriert sein. Ziel ist es, den kompletten Prozess selbst zu realisieren, von der Herstellung des Stoffes bis zur Softwareanwendung. Im interdisziplinären SimpleSkin-Projekt kooperieren sieben Organisationen aus verschiedenen Forschungsrichtungen, um dies zu ermöglichen.

Diese Arbeit definiert Schnittstellen und Zuständigkeitsbereiche, die den Entwicklungsprozess von intelligenter Kleidung verbessern sollen. Der Fokus liegt dabei auf den Schnittstellen die zwischen den Anwendungen für Endbenutzern und den Sensoren liegen. Zielgruppe sind daher Entwickler, die in den Schichten zwischen Sensorik und Anwendung arbeiten, Sensorentwickler, Entwickler von

[1] `http://simpleskin.org/`, zuletzt besucht am 30. Juni 2014

Erkennungsalgorithmen und Softwareentwickler. Ein wichtiger Bestandteil ist zudem die Entwicklung von Werkzeugen, die als Teil der Architektur den Entwicklungsprozess optimieren.

Neben der Definition der Architektur wird diese auch exemplarisch implementiert und in einer Nutzerstudie auf ihre Benutzbarkeit und den Grad der Entwicklerunterstützung untersucht. Zum Testen des Systems wurden auf dem Markt verfügbare Sensoren herangezogen.

Gliederung

Diese Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen und verwandte Arbeiten betrachtet verwandte Arbeiten und Grundlagen dieser Arbeit.

Kapitel 3 – Anforderungen an eine Systemarchitektur für intelligente Kleidung zählt konkrete Anforderungen an eine mögliche Architektur auf.

Kapitel 4 – Architektur einer Middleware zur Entwicklungsunterstützung beschreibt eine Architektur, welche die erfassten Anforderungen befriedigt.

Kapitel 5 – Eingesetzte Technologien und Umsetzung beschäftigt sich mit einer exemplarischen Umsetzung der Architektur. Um die Machbarkeit zu demonstrieren und evaluieren wird die Architektur soweit möglich implementiert.

Kapitel 6 – Nutzerstudie Die Implementierung der Architektur wird im Praxiseinsatz evaluiert.

Kapitel 7 – Zusammenfassung und Ausblick fasst die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

2. Grundlagen und verwandte Arbeiten

> It's a little bit like that wonderful invention called the Segway. It's such a fantastic piece of technology but you just look like a complete dick when you drive around on it.
>
> *(Marc Newson über Google Glass)*

In diesem Kapitel werden für das Thema wichtige Begriffe näher beschrieben. Zudem werden Arbeiten genannt, die sich mit ähnlichen und angrenzenden Themenbereichen beschäftigen. Es werden die Begriffe des Wearable Computing und Intelligenter Kleidung erläutert und der Entwicklungsprozess der selbigen beschrieben. Zudem wird ein Überblick über die Ansätze von Architekturen und Betriebssystemen, die sich bisher auf dem Markt befinden oder von Hobbyisten genutzt werden, gegeben.

2.1. Wearable Computing

Je nach Definition von *wearable* und *computing* könnten schon der erste Rechenschieber am Halsband oder die erste Armbanduhr dieser Kategorie von Gegenständen angehören. Setzt man allerdings voraus, dass ein Computer nicht alleinig dazu da ist, die Zeit anzuzeigen oder einfache Zahlen zu berechnen, sondern auch von Benutzern programmierbar sein soll, dann wurde der erste *wearable Computer* in den späten 1970er-Jahren von Steve Mann entwickelt [Man98b]. Was als rucksackgroßes Gerät begann, ist heute in vielen Produkten gemündet, die einen wachsenden Markt bedienen.

Steve Mann beschreibt einen Wearable Computer in einer Keynote, die er 1998 auf der *International Conference on Wearable Computing* hielt, wie folgt [Man98a]: Der Computer ist immer an, immer verfügbar und immer im persönlichen Bereich des Benutzers. Im Besonderen kann der Benutzer immer damit interagieren, das heißt beispielsweise während er läuft oder andere Tätigkeiten vollbringt. Genau wie auch nicht tragbare Computer sollte der Aspekt der Allgemeingültigkeit und Konfigurierbarkeit beibehalten werden. Ein tragbarer Computer hat die volle Funktionalität eines Computersystems, aber ist zusätzlich mit dem Benutzer untrennbar verbunden. Dies setzt ihn von Armbanduhren, normalen Brillen und tragbaren Radios ab.

Heutige sich auf dem Markt befindliche Wearables entsprechen nicht unbedingt vollständig dieser Definition, da sie eher für spezifische Einsatzzwecke entworfen sind. Zusammen mit einem Smartphone werden die Anwendungen allerdings generischer. Verbindet man das System noch mit einem Heads-Up-Display in einer Brille und Sprachsteuerung kommt es der obigen Definition eines *wearable*

computers sehr nahe. Google Glass[1] ist ein Beispiel für ein solches Gerät: Es vereint Sprachsteuerung, Display in der Brille und programmierbare Anwendungen, sogenannte *Glassware*.

Die heute verfügbaren *Wearable Computing* Produkte sind vielfältig. Für nahezu jeden Körperbereich gibt es entsprechende Geräte, die dem Benutzer Unterhaltung oder Unterstützung bieten sollen. Von oben nach unten begonnen gibt es Brillen und Halsbänder, Brustgurte oder Shirts, Armbänder, Handschuhe, Accessoires für die Beine, Schuhe und Einlagen für diese. Einige Beispiele sind in Abschnitt 2.4 beispielhaft aufgeführt.

2.2. Intelligente Kleidung

Unter intelligenter Kleidung, im Englischen *Smart Clothes*, versteht man heute hauptsächlich Kleidung die digitale Komponenten enthält. Der Begriff des intelligenten Stoffs wurde das erste Mal 1989 in Japan geprägt. In diesem Fall war es ein Seidenstoff mit Formgedächtnis. Durch anhaltende Abnahme der Textilindustrie in der westlichen Welt ist auch das Interesse an intelligenten Textilien gestiegen um einen neuen lukrativen Markt zu schaffen [VH04].

Nach Van Langenhove und Hertleer erfüllen die heutigen Produkte im Bereich von intelligenter Kleidung alle Kriterien, die einen Wandel hin zum Hightech-Produkten erlauben:

- Statt ressourcenbasiert sind sie wissensbasiert
- Qualität statt Quantität
- Nach Nachfrage produzierte, mehrmals benutzbare und ausbaufähige Dienste anstelle von massengefertigten Waren zur einmaligen Verwendung
- Mehrmals benutzbar
- Integrierte Dienstleistungen und nicht greifbare Mehrwerte anstatt materieller Werte

Intelligente Materialien (das heißt nicht nur Textilien) können nach [Tao01] in drei Kategorien eingeteilt werden:

Passive intelligente Materialien können nur die Umgebung oder Reize erfassen. Sie sind einfache Sensoren.

Aktive intelligente Materialien können die Umgebung oder Reize erfassen und darauf reagieren. Im Unterschied zu passiven Materialien müssen diese auch Aktoren enthalten die je nach Sensorwerten entsprechende Aktionen ausführen.

Sehr intelligente Materialien können Umgebung oder Reize erfassen, darauf reagieren und sich entsprechend selbst anpassen. Dazu ist neben den Sensor und Aktor noch ein dritter Teil notwendig, der für eine intelligente Steuerung des Aktors zuständig ist. Solche Stoffe werden

[1] `http://www.google.com/glass/`

> durch die Kombination von traditioneller Kleidungs- und Textiltechnolgoie mit Materialwissenschaften, Sensortechnologie, Datenverarbeitungs- und Kommunikationstechnik, Künstlicher Intelligenz, Biologie.

Es wurden schon seit langem versucht, die Natur zu kopieren um unsere Kleidungsmaterialien zu verbessern. Ein Beispiel dafür sind Mikrofasern, die mit der Untersuchung von Seide begonnen haben. Bis jetzt sind die meisten Stoffe allerdings nicht intelligent sondern ohne Leben. Einen Stoff zu entwickeln, der so intelligent wie unsere menschliche Haut ist, ist ein sehr herausforderndes Ziel. Die Entwicklung von intelligenten Materialien, auch Funktionswerkstoffe genannt, ist allerdings in vollem Gange. Es gibt viele verschiedene Materialien die zu den Funktionswerkstoffen zählen. Beispielsweise Stoffe mit piezoelektrischem Effekt, welche Spannung produzieren wenn sie verformt werden, oder sich beim anlegen von Spannung verformen. Ein weiteres Beispiel sind thermoresponsive Polymere welche ihre physikalischen Eigenschaften unter Einfluss von Temperatur verändern [Wik14].

Zum aktuellen Zeitpunkt ist die Marktdurchdringung von intelligenter Kleidung noch gering, allerdings wird dieser Markt als stark wachsend angesehen [DS14]. Dabei sind viele verschiedene Interessensvertreter und Produktsegmente in diesem Markt engagiert: Persönliche Schutzausrüstung und Militär, Raumfahrt, Telemedizin, Gesundheitsprodukte, Kleidermode, Licht abstrahlende Kleidung und Audio-Unterhaltung. Die Markttreiber sind dabei vor allem professionelle Sportanwendungen und der Trend zur Telemedizin um die Kosten für das Gesundheitswesen zu senken. Forschung und Entwicklung durch die Europäische Union, die Europäische Weltraumorganisation und NASA, tragen allerdings auch stetig zur Marktvergrößerung bei.

Viel größer als der Markt für intelligente Textilien ist im Moment der Markt für tragbare intelligente Geräte, *Wearable Technology*. Diese Geräte sind klein genug, um direkt am Körper getragen zu werden. Sie erfüllen häufig einen bestimmen sensorischen Zweck und eine bestimmte Anwendung. Laut Vandrico Inc. [Van14] sind im Mai 2014 220 Geräte auf dem Markt. Im Rahmen dieser Arbeit werden aufgrund der besseren Verfügbarkeit tragbare Geräte als Datenquelle betrachtet, allerdings immer mit dem Hintergrund, dass die Ergebnisse auch auf Kleidung anwendbar sein sollen.

2.3. Der Entwicklungsprozess von Intelligenter Kleidung

Suh, Carrol und Cassill [SCC10] identifizieren im Entwicklungsprozess von intelligenter Kleidung folgende fünf Phasen, von denen Entwurf, Entwicklung und Evaluation relevant für diese Arbeit sind.

1. Ideenfindung
2. Entwurf
3. Entwicklung eines Prototypen
4. Evaluation und Verfeinerung des Designs
5. Produktionsplanung

Die beteiligten Disziplinen im Entwicklungsprozess sind in die drei großen Teilbereiche *Textile Technologien*, *Physiologie* und *Design Research*: Verschiedene Technologien im Textilbereich, die Physiologie des Trägers und die Kommunikation des Systems mit der Umwelt.

Ähnlich zu Entwicklungsprozessen wie beispielsweise in Softwareprojekten steigen die Kosten einer Neugestaltung gegen Projektende bzw. Produktionsstart an, während sie in den Entwicklungsphase noch gering sind [JL07, SCC10]. Daher ist es von hohem wirtschaftlichen Interesse, mögliche Fehler oder Anforderungen, die größere Änderungen am Ende des Projekt erforderlich machen würden, schon früh zu finden und beheben. Tom de Schutter [Sch13] beschreibt, welche Vorteile eine parallele Entwicklung von Hardware und Software im Bereich eingebetteter System haben. In dem Artikel wird beschrieben, wie virtuelle Prototypen der Hardware als Basis für die Softwareentwicklung dienen können, um die Entwicklungen nebenläufig beginnen zu lassen.

2.4. Industrielle Herstellung von Wearables

Der Marktüberblick des Unternehmens Vandrico [Van14] gibt eine gute Übersicht über die am Markt verfügbaren Geräte. Im Jahr 2014 sollen 48 neue Geräte auf den Markt kommen, das sind 27% der Gesamtmenge. Dabei enthalten Geräte für die Kategorien Unterhaltung, Lifestyle und Fitness die meisten Komponenten, durchschnittlich zwischen vier und fünf. Im medizinischen und industriellen Einsatz sind weniger Komponenten im Einsatz, dafür sind diese allerdings wesentlich teurer und präziser.

Als Zielplattformen, um mit den Geräten zu kommunizieren, wählen die meisten Hersteller im Moment iOS und Android. Die Kompatibilität mit mobilen Betriebssystemen hat Priorität über Desktopbetriebssystemen wie Windows und Mac OS X. Andere Plattformen wie Windows Mobile, Blackberry und Linux werden kaum unterstützt.

Die meisten Geräte benutzen Bluetooth als Schnittstelle zur Kommunikation (147 von 225, Stand: 27. Mai 2014). Bluetooth ist Industriestandard für die Datenübertragung über kurze Distanzen. Die erste Version von Bluetooth wurde in den 1990er-Jahren entwickelt. Seit Dezember 2009 liegt Bluetooth in Version 4.0 vor. Die ersten kompatible Geräte wurden im Juni 2011 auf den Markt gebracht.

Im Bluetooth 4.0 Standard ist auch *Bluetooth Smart* enthalten. Dies ist eine vom klassischen Bluetooth abweichende Funktechnik, die einen deutlich geringeren Stromverbrauch verspricht. Geräte, die Bluetooth Smart unterstützen, müssen nicht zwangsläufig kompatibel mit klassischen Bluetooth-Geräten sein. Bluetooth Smart bietet eine ähnliche Reichweite wie klassische Bluetooth-Technologie, ist allerdings auf eine Datenübertragungsrate von 1 MBit/s beschränkt. Die verfügbare effektive Datenrate für Anwendungen liegt allerdings bei nur 0,27 MBit/s.

Der Bluetoothstandard definiert selbst Schnittstellen für die Kommunikation in Bluetooth-Umgebungen. Diese Spezifikationen werden Bluetooth-Profile genannt. Um ein Profil zu nutzen oder anzubieten, muss es damit kompatibel sein. Dabei können die Profile sich gegenseitig erweitern oder benötigen. Beispielsweise gibt es ein generisches *Health Device Profile*, von dem zwei Spezialisierungen *Health Thermometer Profile* und *Heart Rate Profile* existieren.

Die am 2. Juni 2014 von Apple angekündigte neue Betriebssystemversion iOS 8 soll eine Anwendung namens *Healthbook* enthalten. Diese soll auf dem Smartphone anstelle der einzelnen spezifischen Anwendungen alle Gesundheitsdaten aggregieren. Dazu setzt die Anwendung auf Bluetooth und von Standard definierte Bluetooth-Profile für Messgeräte für Puls, Blutzucker, Blutsauerstoff und mehr. Wie genau die Schnittstellen für die Anwendung aussehen werden ist allerdings noch nicht bekannt.

2.5. Do-it-yourself Wearables

Neben der industriellen Herstellung von Wearables gibt es auch eine Community, die sich damit beschäftigt, ihre eigenen tragbaren Geräte und Anwendungen zu bauen. Dabei ist auch die Verknüpfung von elektronischen Bauteilen mit Kleidung ein wichtiger Teil. Entwickelte Projekte sind dabei vielfältig und reichen von reinen modischen Accessoires wie LED bestückten programmierbaren Halsbändern[2] bis zu neuartigen Eingabegeräten wie Drucksensitiven Handschuhen[3].

Häufig wird dabei Hardware, die auf der Arduino-Plattform basiert, zum Einsatz gebracht. Das Buch *Open Softwear: Fashionable prototyping and wearable computing using the Arduino* [OGOW08] beschreibt Grundlagen der Hardware und Programmierung eines Arduinos und gibt Anwendungsbeispiele und Anleitung die speziell auf Kleidung ausgelegt sind.

Die Arduino-Plattform zählt in den Bereich des *Physical Computing*, das heißt durch die Kombination von Hardware und Software sollen physische Systeme erstellt werden, die auf Ereignisse in der realen, analogen Welt reagieren oder auf diese einwirken. Die Arduino-Plattform besteht aus einem programmierbaren Mikrocontroller und digitalen und/oder analogen Schnittstellen die damit angesteuert werden könne. Die Programmierung erfolgt dabei in C oder C++. Dabei haben die Arduino-Boards normalerweise eine USB-Schnittstelle, mit der sie mit dem Computer verbunden werden können und über diese der Programmcode aktualisiert werden kann.

Es gibt zwischenzeitlich verschiedene Hardware, die der Spezifikation dieser Plattform entspricht und damit kompatibel ist. Insbesondere gibt es für Wearables spezialisierte Formfaktoren, die zum Beispiel unter dem Namen LilyPad[4] bekannt sind. Diese sind darauf ausgelegt, mit Kleidung vernäht zu werden. Die Geräte sind klein und haben die Form einer Scheibe. Die elektrischen Kontakte sind durch Löcher realisiert, so dass die Kommunikationsinfrastruktur mit elektrisch leitfähigen Fäden realisiert werden kann. Neben LilyPad gibt es auch ähnliche Produkte wie beispielsweise die *FLORA*[5]-Plattform, welche auch kompatibel zu Arduino ist.

[2] `https://learn.adafruit.com/neopixel-punk-collar`, zuletzt abgerufen am 30. Juni 2014
[3] `https://learn.adafruit.com/midi-drum-glove`, zuletzt abgerufen am 30. Juni 2014
[4] `http://lilypadarduino.org/`, zuletzt abgerufen am 30. Juni 2014
[5] `http://www.adafruit.com/categories/92`, zuletzt abgerufen am 10. Juni 2014

2.6. Prototypen in der Wissenschaft

In der Arbeit *What Shall We Teach Our Pants?* [VC00], wird ein System zur Aktivitätserkennung des Benutzers anhand von Beschleunigungssensoren an Hosen entwickelt und evaluiert. Neben den Beschleunigungsdaten wurden allerdings auch noch passive Infrarot-Sensoren, ein Kohlenstoffmonoxid-Sensor, Mikrofone, Drucksensoren, Temperatursensoren und Helligkeitssensoren benutzt.

Die rohen Sensordaten wurden dann in einem ersten Schritt in sogenannte Anhaltspunkte vorverarbeitet. Diese Anhaltspunkte werden dann benutzt um durch maschinelles Lernen ein Modell zu entwickeln, das die aktuelle Aktivität des Benutzers beschreibt. Für den Prototypen waren einfache Algorithmen wie Summe, Maximum, Minimum und Durchschnitt für die meisten Sensordaten ausreichend, um sinnvolle Anhaltspunkte zu generieren.

Diese Anhaltspunkte wurden später in Echtzeit verarbeitet und auch für Reproduzierbarkeit und spätere Analysen gespeichert. Im Experiment mussten die Probanden dann ihre Tätigkeiten während der Durchführung angeben um den Prozess des maschinellen Lernens zu unterstützen. Das Ergebnis dieser Arbeit ist, dass Aktivitätserkennung oder das Bewusstsein des Benutzerkontexts ohne anpassungsfähige Analysen sehr beschränkt ist.

In der Arbeit *Electronic Textiles: A Platform for Pervasive Computing* [MMZ+03] werden unter anderem Module beschrieben, aus denen sich ein Gesamtsystem zusammensetzen könnte und wie die Hardwareverbindungen zwischen den Komponenten realisiert werden könnten. Auf Softwareseite werden Vorschläge zur Organisation und Kommunikation mit einer möglichen Hardwareschnittstelle und der weiteren Signalverarbeitung gegeben.

Wear Ur World - A Wearable Gestural Interface [MMC09] ist ein System, das aus einem kleinen Beamer und einer auf dem Kopf bzw. Hut angebrachten Kamera besteht. Das System sieht was der Benutzer sieht und reichert seine Umgebung durch Projektionen auf Oberfläche an. Die Interaktion erfolgt mittels Handgesten, Armbewegungen oder Interaktion mit dem Objekt. Als Software kommt bei dem Projekt WPF, C# und OpenCV zum Einsatz.

Pinstripe [KWL+11] ist ein Projekt der RWTH Aachen, in dem das zusammendrücken und rollen von Stoff mit den Fingern als Eingabe des Benutzers interpretiert wird. In dieser Arbeit wurden die im Stoff verbauten leitfähigen Fäden mit einem LilyPad Mikrocontroller verbunden. Dieser ist mit einem Kabel an einen Computer angebunden, der die Signalverarbeitung durchführt.

2.7. Die Architektur von intelligenter Kleidung

Um intelligente Kleidung günstiger und in großen Mengen zu produzieren, muss der Produktionsprozess besser modularisiert werden. Dabei ist dies ein Henne-Ei-Problem: Erst bei erschwinglichen Preisen verbreiten sich die Produkte stark, und erst bei großer Verbreitung ist eine günstige Massenproduktion erstrebenswert.

Ein weiteres Problem ist zudem, dass es noch keine *killer application* gibt, die für viele verschiedene intelligente Kleidungen funktioniert. Als Killeranwendung wird eine konkrete Anwendung bezeichnet,

Abbildung 2.1.: Die grobe Architektur von intelligenter Kleidung nach [CLH+13]

die einer existierenden Technik zum Durchbruch verhilft. Statt einer generell einsetzbaren Anwendung für intelligente Kleidung gibt es nur einzelne Lösungen mit ihren jeweils eigenen spezifischen sensorischen Anforderungen.

Die verschiedenen Disziplinen, die am Entwicklungsprozess beteiligt sind, sollten unabhängiger voneinander arbeiten können um vielfältige aber miteinander kompatible Produkte erzeugen zu können. Ein Weg um die Kopplung von Zuständigkeiten zu verringern und dabei die Kohäsion zu steigern, ist die Schichtenarchitektur. Beteiligte Schichten dürfen nur mit den über und unter ihnen gelagerten Schichten kommunizieren und Aufrufe sollten wenn möglich nur auf untere Schichten gemacht werden.

In Abbildung 2.1 sind mögliche Schichten dargestellt, an denen einzelne Komponenten voneinander getrennt werden könnten. Die einzelnen Schichten werden, von unten nach oben, in den folgenden Abschnitten näher erläutert.

Sensorischer Stoff

Hiermit ist das Textilprodukt gemeint, welches neben der Funktion als Kleidung zu dienen, zusätzliche Daten seiner Umwelt aufnehmen kann. Im Projekt SimpleSkin wird an leitfähigen Materialien gearbeitet, die in einem zweidimensionalen Gitter in oder auf dem Stoff angebracht sind. Zwei dieser Gitter, mit einer zwischengelagerten Isolationsschicht, könnten beispielsweise dazu dienen, den Druck durch Messung des elektrischen Widerstands zu messen.

Die Verarbeitung des Stoffes sollte im besten Fall auch nicht stark von traditionellen Bearbeitungsmethoden abweichen. Es muss allerdings beachtet werden, dass ein auf dem Stoff befindliches Netzwerk von Sensorik mit der Kommunikations- und Engergieinfrastruktur verbunden sein muss.

Hardware-Schnittstelle, Signalverarbeitung, Computer-Hardware

An diese Stelle erfolgt die erste große Trennung der Komponenten. Die auf dem Stoff aufgebrauchte Sensor-Infrastruktur sollte durch ein steckbares Gerät benutzbar gemacht werden. So kann der Stoff ohne die elektronischen Bestandteile gewaschen werden. Über die Hardware-Schnittstelle sollen alle notwendigen Informationen die im Stoff anfallen übertragen werden, so dass die digitale und analoge Signalverarbeitung dort mithilfe der Computer-Hardware betrieben werden kann.

Wearable OS

Das Betriebssystem soll auf der Computer-Hardware laufen. Es kümmert sich darum, die Daten aus der Hardware auszulesen und den über ihm liegenden Komponenten zur Verfügung zu stellen. Es sollte mit verschiedenen Sensoren, die über die Hardware-Schnittstelle angeschlossen werden, kommunizieren können.

Laufzeitumgebung, Sensor API, Apps

Anwendungen, sogenannte Apps, die dem Endbenutzer Zugriff auf und Interaktion mit seiner Kleidung bieten sollen benötigen eine Laufzeitumgebung. Diese Laufzeitumgebung soll über eine einheitliche API die Sensordaten zur Verfügung stellen.

2.8. Betriebssysteme für Wearables

Es gibt auf dem Markt verschiedene Ansätze für Betriebssysteme die auf Wearables zugeschnitten sind. Folgende Produkte sind dabei verbreitet oder in Entwicklung.

Android Wear

Android Wear[6] wurde am 18. März 2014 von Google vorgestellt. Es basiert auf dem mobilen Betriebssystem Android und ist für Geräte ausgelegt die in der Größenordnung von Armbanduhren liegen.

In der im Moment verfügbaren Vorschau für Entwickler wird eine Übersicht über die Elemente und Entwurfsmuster der Benutzungsschnittstelle gegeben. Die Hauptaufgabe von tragbaren Geräten besteht darin, mit Benachrichtigungen zu interagieren. Vor allem sollen Informationen immer passend zum Kontext geliefert werden, ohne dass der Benutzer diese explizit nachfragen muss.

Zur Kommunikation mit externer Hardware oder Sensoren ist in der Vorschau für Entwickler noch nichts geschrieben. Der Plan für die Zukünftige Erweiterung enthält allerdings folgende Punkte die Android Wear unterstützen wird:

[6] `http://www.android.com/wear/`, zuletzt abgerufen am 30. Juni 2014

- Eigene Benutzungsoberflächen erstellen
- Daten zwischen Smartphones und tragbaren Computern austauschen
- Sensordaten in Echtzeit sammeln und darstellen
- Sprachinteraktion mit Anwendungen

Tizen

Tizen[7] ist ein auf Linux basierendes Betriebssystem. Es ist darauf ausgelegt auf Geräten wie Smartphones, Tablets, Smart TVs, Fotoapparaten, Autounterhaltungssystemen aber auch Geräten im Bereich Wearable-Computing eingesetzt zu werden. Die erste Version von *Tizen* wurde im Januar 2012 veröffentlicht. Das Projekt ist in stetiger Weiterentwicklung und hat viele große Partner in der Industrie wie zum Beispiel Fujitsu, Huawei und die Intel Corporation.

Tizen setzt stark auf mobile standardisierte Webtechnologien zur Anwendungsentwicklung wie JavaScript, XML, Ajax und alle weiteren unter dem Sammelbegriff HTML5 bekannte Technologien. Die Begründung liegt darin, dass HTML5 die Barriere zum Entwickeln von betriebssystemunabhängigen Anwendungen heruntersetzt und damit mehr Entwickler anzieht. Neben HTML5-Anwendungen unterstützt Tizen aber auch native Projekte in den Programmiersprachen C oder C++ [Tiz13].

Die von Samsung entwickelten Geräte der Marke *Gear 2* basieren auf dem Tizen-Betriebssystem, welches die frühere auf Android basierende Plattform ablöst. Ältere Geräte wie die Smartwatch *Galaxy Gear* können von Android aus auf die neue Tizen-Plattform aktualisiert werden.

Wearable OS

Wearable OS[8] ist ein Projekt, welches über die Crowdfunding-Plattform Kickstarter finanziert werden will. Es hat zum Ziel ein Betriebssystem und eine Entwicklungsumgebung für tragbare Computer zu entwickeln. Zum aktuellen Stand (29. Juni 2014) gibt es allerdings noch keine weiteren Informationen.

Auf der Webseite ist beschrieben, dass eine Entwicklungsumgebung geschaffen werden soll, die erlaubt Anwendungen aus vorgefertigten Blöcken zusammenzustellen und aus diesen Programmen dann den Code für verschiedenen Plattformen zu generieren. Dabei soll Wert auf interaktives und iteratives Entwickeln gelegt werden und das Feedback von Änderungen schnellstmöglich sichtbar werden. Von Anwendungen gesammelte Daten werden über einen Cloud-Dienst synchronisiert um analysiert zu werden.

[7] `https://www.tizen.org/`
[8] `http://wearable-os.com/`, zuletzt abgerufen am 10. Juni 2014

2.9. Sensornetzwerke, Verteilte Systeme und Datenmanagement

Unter dem Begriff Sensornetzwerk versteht man ein Netz von Sensorknoten, die ein einem Netzwerk zusammenarbeiten. Das Sensornetzwerk soll dadurch eine intelligente Umgebung aufspannen [Lew04]. Das Ergebnis dieser Arbeit ist ähnlich zu einem Sensornetzwerk, indem die an dem Körper der Person getragenen Sensoren zusammenarbeiten um nützliche Daten zu liefern. Dabei ist es allerdings nicht notwendig, dass die Sensoren selbst ihre Kommunikationsinfrastruktur organisieren.

Lehikoinen und Holopainen entwarfen in *MEX: a distributed soft- ware architecture for wearable computers* eine Architektur für ein verteiltes System von Wearable Computern [LHSA99]. Die Ziele dieser Architektur sind:

1. Kommunikation zwischen Anwendungen, die auf einem Wearable Computer laufen
2. Kommunikation der Anwendung mit dem ubiquitären Umfeld
3. Kommunikation von Anwendungen zwischen verschiedenen Wearable Computern

Die Arbeit geht dabei wenig auf die Kommunikation mit Sensoren ein, sondern konzentriert sich auf die Kommunikation zwischen verschiedenen Anwendungen.

Auch Kortuem, Bauer und Segall beschreiben mit NETMAN [KBS99] ein System, in dem die Eingabe, Verarbeitung und Ausgabe von Daten die von Wearable Computer erfasst werden, auf verschiedene Komponenten verteilt. Dadurch soll das Zusammenarbeiten von Personen verbessert werden. Im dort beschriebenen Szenario wird ein Monteur mit einer Kamera und Headset zu seinem Einsatzort geschickt und bekommt von einem Technikexperten der stationär ist Anweisungen oder Lösungen die zu seiner aktuellen Problemstellung passen.

WikiSensing[9] ist ein Projekt, dass es über Webschnittstellen erlaubt, Sensordaten im Internet zu veröffentlichen und abzurufen [SGG12]. Die Sensordaten können dabei über HTTP in den Formaten XML oder JSON angelegt und abgefragt werden.

Das Produkt Human API[10] hat sich auf die Aggregation von Gesundheitsdaten spezialisiert. Es unterstützt sowohl tragbare Geräte wie Armbänder, als auch stationäre Geräte wie Waagen oder Messgeräte für den Blutzuckerspiegel. Es greift auf die Webschnittstellen der einzelnen Produkte zu und führt all diese Gesundheitsdaten zusammen. Anwendungsentwickler können somit auf die durch Human API aggregierten Daten in einer einheitlichen Weise zugreifen, unabhängig von der benutzten Quelle.

Xively[11] (früher bekannt unter dem Namen COSM und Pachube) ist ein Service der online Sensordaten aggregiert und diese auch über eine API zur Verfügung stellt. Xively bietet fertige Bibliotheken für eine Vielzahl von Plattformen an. Der Quellcode dieser Bibliotheken ist öffentlich verfügbar. Die Plattform selbst ist allerdings proprietär und kann nicht auf eigenen Server ausgebracht werden. Die Zielsetzung von Xively ist, die Plattform für das Internet der Dinge zu werden.

[9] `http://wikisensing.org/`, zuletzt abgerufen am 30. Juni 2014
[10] `http://humanapi.co/`, zuletzt abgerufen am 30. Juni 2014
[11] `https://xively.com/`, zuletzt abgerufen am 30. Juni 2014

2.10. Produkte

Es gibt schon Produkte, die sich ähnlich zu zukünftiger intelligenter Kleidung verhalten, aber eben eigenständige Geräte sind, die nicht in der Kleidung integriert sind. Sie besitzen ausreichende Batteriekapazität und Speicherplatz um den ganzen Tag getragen zu werden und alle anfallenden Daten aufzuzeichnen.

2.10.1. Zephyr BioHarness 3

Das BioHarness 3 ist ein Gerät der Firma Zephyr[12]. Es besteht aus einem Brustgurt, in den das Hardware-Modul eingesetzt werden kann. Dieses hat einen Akku, der über den USB-Anschluss geladen werden kann und einen integrierten Speicher um aufgezeichnete Daten zu speichern. Alle hier aufgeführten Spezifikationen sind dem Handbuch entnommen [Zep11a].

Mit anderen Geräten kommunizieren kann der BioHarness (BH3) über den Universal Serial Bus (USB), Bluetooth und ein proprietäres Protokoll (802.15.4 ECHO). Dieses Protokoll namens ECHO ist auf eine Reichweite von 275 Metern ausgelegt und wird hauptsächlich mit anderen, von Zephyr vertriebenen Hardwarekomponenten zur Datenanalyse benutzt. Über USB und Bluetooth wird ein proprietäres serielles Protokoll gesprochen. Details zu diesem Protokoll befinden sich in Abschnitt 5.4.1 und 5.9.

Der BH3 unterstützt die folgenden grundlegenden physiologischen Daten:

- Puls
- Atemfrequenz
- EKG Daten (Elektrokardiogramm)
- Daten der Beschleunigungssensoren

Der BH3 liefert zudem schon vorverarbeitete Daten wie das Minimum und Maximum der einzelnen Beschleunigungswerte pro Achse und Werte die das Vertrauen in die Richtigkeit von gelieferten Daten wie Puls, Atemfrequenz oder auch das Gesamtsystem beschreiben.

2.10.2. Polar Loop

Der Polar Loop[13] wird als Armband getragen und ist dafür vorgesehen die Aktivitäten des Trägers dauerhaft zu erfassen. Es ist wasserdicht und kann Daten für 12 Tage aufzeichnen. Bei Dauernutzung hält die Batterie laut Hersteller bis zu fünf Tage.

Zu den erfassten Daten zählen das Aktivitätslevel (in den Stufen niedrig, mittel und hoch), die Anzahl der gelaufenen Schritte, die zurückgelegte Distanz und Schlafqualität. Diese Daten werden vom Rechner aus in ein Webportal geladen, in dem diese analysiert werden. Mit einem Webbrowser kann

[12] `http://zephyranywhere.com/`, zuletzt abgerufen am 30. Juni 2014
[13] `http://www.polarloop.com/`, zuletzt abgerufen am 22. Juni 2014

die Auswertung und Visualisierungen seiner Daten betrachtet werden. Über das Portal können die Daten auch mit anderen Benutzer geteilt werden.

2.11. Zusammenfassung

Produkte im Bereich Wearable Computing kommen mehr und mehr im Massenmarkt an. Momentan sind dies allerdings hauptsächlich externe Geräte, die temporär zu bestimmten Zwecken getragen werden. Intelligente Kleidung könnte der nächste Schritt sein, um Wearable-Computing Anwendungen eine konstante und dauerhafte Datenquelle zu liefern und den Computer noch näher an den Menschen zu bringen.

Der Entwicklungsprozess von intelligenter Kleidung ist vergleichbar zum Vorgehen von anderen High-Tech Produkten, allerdings stark interdisziplinär, da neben der reinen Technologie auch das Produktdesign und die Physiologie des Menschen eine wichtige Rolle spielen.

Industriell gefertigte Produkte aus dem Bereich Wearables sind meist abgeschlossene Systeme, welche die von ihnen Erfassten Daten über eine definierte Schnittstelle, meist über Bluetooth, zur Verfügung stellen. Im Hobbyistenbereich hat sich vor allem die Arduino-Plattform zur Entwicklung von intelligenter Kleidung und Accessoires durchgesetzt.

Wissenschaftliche Prototypen für intelligente Kleidung sind wenig standardisiert und spezifisch auf den Anwendungsfall ausgelegt. Da es sich meist um Prototypen handelt die nicht für die Massenfertigung vorgesehen sind, ist dies verständlich. Um zukünftige Produktentwicklungen massentauglich zu machen, müssen Schnittstellen zwischen den Beteiligten Parteien und Zuständigkeiten im Entwicklungsprozess entwickelt und umgesetzt werden. Existierende Betriebssysteme wie Android oder Tizen sind im Moment auch dabei, sich auf Wearable Computing auszurichten und dafür spezielle Funktionen und Entwicklungsunterstützung anzubieten.

3. Anforderungen an eine Systemarchitektur für intelligente Kleidung

> Der schwierigste Teil einer Anforderungsanalyse ist nicht, aufzuzeichnen was der Benutzer will, es ist die explorative Entwicklungstätigkeit des Unterstützen der Benutzer, herauszufinden sie wollen.
>
> *(Steve McConnell)*

In diesem Kapitel werden die Anforderungen, welche an eine Architektur für intelligente Kleidung gestellt werden, genauer beschrieben. Dabei wird als Grundlage die in Abschnitt 2.7 von Cheng vorgeschlagene Schichtenarchitektur benutzt. Bei der Auflistung der Anforderungen werden alle Schichten betrachtet, allerdings liegt der Fokus im Rest der Arbeit auf den Schichten, die über der Computer-Hardware liegen (siehe auch Abbildung 2.1), dass heißt dem Betriebssystem, der Laufzeitumgebung und der Sensor API.

Aufgrund dieses Fokuses sind die Zielgruppen der Architektur Entwickler, welche mit den genannten Schichten interagieren. Das sind Entwickler von Sensoren, Entwickler der Datenverarbeitung und schließlich Entwickler welche die Daten in Anwendungen für Endbenutzer einsetzen.

Die Anforderungen wurden zum einen aus Arbeiten im Bereich des Wearable Computings abgeleitet. Es wurde aber auch mit Entwicklern die beispielsweise im SimpleSkin-Projekt tätig sind über ihre Erfahrungen gesprochen. Vor allem die noch nicht vorhandenen technischen Grundlagen haben dazu beigetragen, die Architektur auf die Unterstützung des Entwicklungsprozess zu optimieren und zu helfen, diesen zu parallelisieren.

3.1. Anforderungen an die einzelnen Schichten

Im folgenden werden funktionale und nichtfunktionale Anforderungen an die einzelnen Schichten der Architektur definiert. Dabei wird in den Schichten aus Abbildung 2.1 von oben nach unten vorgegangen.

3.1.1. Anwendungen und Laufzeitumgebung

Die Anwendungen sind dazu da, dem Benutzer eine Interaktion mit den Daten anzubieten. Diese Anwendungen sollen in einer Laufzeitumgebung ausgeführt werden. Die Laufzeitumgebung ist

beispielsweise ein mobiles Betriebssystem wie iOS oder Android. Es soll aber auch möglich sein Anwendungen in beliebigen andere Technologien und Laufzeitumgebungen schreiben zu können. Die Architektur sollte also kein beschränkender Faktor für Anwendungen oder Laufzeitumgebungen darstellen und so portabel wie möglich sein.

Um Anwendungslogik zwischen verschiedenen Anwendungen oder Laufzeitumgebungen wiederverwenden zu können, sollen diese über eine einheitliche API auf Sensordaten zugreifen und Datenverarbeitung abstrahieren.

3.1.2. Einheitliche Schnittstelle für Anwendungen

Die API ist der einheitliche Zugriffspunkt auf alle in unteren Schichten anfallenden Werte. Mehrere Anwendungen sollen auch gleichzeitig auf mehrere Sensorwerte zugreifen können und sich dabei gegenseitig nicht blockieren.

Die Schnittstelle, mit der Anwendungen auf die Daten zugreifen, soll von den unterschiedliche Geräten, welche die Daten liefern, abstrahiert sein. Daten der selben Kategorie sollen über verschiedene Quellen hinweg in einem einheitliches Format angeboten werden.

Das ausgegebene Format soll maschinenlesbar sein aber dennoch auch ohne komplexe Werkzeuge von Entwicklern verstanden werden. Dadurch soll die Einfachheit der Entwicklung verbessert werden.

Über die Schnittstelle soll sowohl Zugriff auf historische Daten, die als Aufzeichnung vorliegen, als auch auf Echtzeitdaten die in Echtzeit von einem Sensor geliefert werden, geben. Dabei soll der Wechsel zwischen den zwei Arten von Datenquellen ohne große Anpassungen im aufrufenden Code zu bewerkstelligen sein.

3.1.3. Wearable OS

Das Wearable OS ist dafür zuständig, mit der Computer-Hardware zu kommunizieren. Es soll die Daten von angeschlossenen Sensoren aufzeichnen und diese der oberen Schicht, der Laufzeitumgebung, bereitstellen. Es soll möglichst portabel sein und auf Geräten, die den Anforderungen an die Computer-Hardware genügen, lauffähig sein.

3.1.4. Computer-Hardware

Die Aufgabe der Computer-Hardware ist es, Host für das Wearable OS zu sein und die an der Hardware-Schnittstelle anliegenden Daten auszuwerten. Da diese Hardware am Körper getragen wird, sollte sie klein, leicht und robust sein. Die digitale und analoge Signalverarbeitung soll auch auf der Computer-Hardware ausgeführt werden können.

Es gibt andere Arbeiten, die sich mit Formfaktoren und technischer Machbarkeit näher beschäftigen. Beispiele dafür sind *Design of the QBIC wearable computing platform* [ALO+04] und *A Wearable Sensor and Notification Platform* [MRSS06]. Genaue technische Details und Anforderungen der Computer-Hardware sind nicht Teil dieser Arbeit.

3.1.5. Hardware-Schnittstelle

Die hier betrachteten Anforderungen sollten die Hardware-Schnittstelle nicht einschränken. Die genaue technische Spezifikation der Hardware-Schnittstelle ist nicht Teil dieser Arbeit, allerdings werden verschiedene grobe Ausgestaltungen dieser betrachtet.

Da der Markt für intelligente Textilien erst im Wachsen ist, bietet es sich an, in der Übergangsphase auch andere im Moment auf dem Markt befindliche Geräte als Eingabe zu unterstützen. Die Architektur soll daher beliebige zusätzliche Quellen als Eingabe nutzen können.

3.2. Anforderungen aus Entwicklersicht

Der Fokus dieser Arbeit liegt in der Entwicklung von Anwendungen für intelligente Kleidung und somit im speziellen auf den folgenden drei Benutzergruppen. Ein Ziel ist es, neben der Gesamtfunktionalität der Architektur auch den Entwicklungsprozess an sich zu optimieren in dem etwaige Lücken zwischen den Benutzergruppen geschlossen werden.

3.2.1. Sensorentwickler

Die Personengruppe der Sensorentwickler ist mit der Entwicklung und der Verbesserung der Sensoren beschäftigt. Die Art der Sensoren ist dabei zweitrangig. Als Sensor sollen alle Geräte betrachtet werden, die bestimmte Werte liefern welches sich zeitlich verändern.

Während der Entwicklung eines neuen Sensors können die Prototypen noch Eigenschaften haben, die es nicht erlauben ihn mobil und am Körper zu betreiben. Dennoch sollen diese Sensordaten während der Entwicklung schon erfasst werden können. Diese Aufzeichnungen können dann anderen im Entwicklungsprozess beteiligten Gruppen zur Verfügung gestellt zu werden. Die Schnittstelle zum Erfassen der Daten soll so einfach wie möglich gestaltet sein, um die Kosten für eine Parallelisierung des Entwicklungsprozesses zu minimieren.

Als Vorteil des frühzeitigen Einpflegens der Sensordaten soll den Entwicklern eine Visualisierung ihrer Daten geboten werden. Damit sollen Tests in frühen Phasen ohne große zusätzlichen Entwicklungskosten für spezifische Analysesoftware gemacht werden können.

3.2.2. Datenverarbeitungsexperten

Die Gruppe der Datenverarbeitungsexperten beschäftigt sich damit, die von den Sensoren gelieferten Daten genauer auszuwerten. Aus den rohen Sensordaten sollen Ableitungen gemacht werden, welche die Realität des Benutzers auf einem höheren Level beschreiben.

Um frühzeitig mit dem Entwerfen von Algorithmen zur Datenverarbeitung beginnen zu können, sollen aufgezeichnete Daten gleichwertig wie Echtzeitdaten benutzt werden können. Dabei sollen auf die von den Sensorentwicklern erzeugten und eingepflegten Daten zugegriffen werden können.

Um das Erstellen von neuen Algorithmen zu erleichtern, sollen graphische Hilfsmittel zur Visualisierung der Daten und Datenverarbeitungen angeboten werden. Zudem soll es eine Bibliothek mit vordefinierten Grundbausteinen geben, um schnell einfache Algorithmen erstellen zu können. Maschinelles Lernen sollte auch eine Option sein und durch das System unterstützt werden. Dazu sollten Eingaben vom Benutzer explizit kategorisiert werden können und damit zum überwachten Lernen benutzt werden können.

3.2.3. Anwendungsentwickler

Anwendungsentwickler sind dafür zuständig, die anfallenden Daten dem Endbenutzer in einer für ihn sinnvollen Ausgabe darzustellen und ihm Interaktionsmöglichkeiten zu bieten.

Um früh im Entwicklungsprozess des Gesamtproduktes mit Prototypen der Anwendung zu beginnen, soll auf alle von Sensorentwicklern und Datenverarbeitungsexperten erstellten Zwischenprodukte zugegriffen werden können. Es sollen also beispielsweise Aufzeichnungen oder Echtzeitdaten der Sensoren verwendet werden. Auch die Ergebnisse der entwickelten Algorithmen zur Analyse dieser Daten sollen von den Anwendungsentwicklern eingesetzt werden können. Um zwischen Aufzeichnung und realem Sensor zu wechseln sollen keine großen Änderungen am Programmcode notwendig sein.

Der Zugriff auf diese Daten soll über eine einheitliche API möglich sein. Diese soll auf möglichst vielen Plattformen verfügbar sein, um Anwendungsentwicklungen nicht unnötig zu beschränken. Zudem sollen die eingesetzten Technologien mit existierenden Entwicklungswerkzeugen ausreichend gut unterstützt werden. Dadurch soll ein erneutes Henne-Ei-Problem verhindert werden, bei dem sich die neue Plattform nicht durchsetzt, da sie nicht ausreichend viele Verbesserungen bietet und nur gleichauf mit älteren aber weiterentwickelten Insellösungen liegt.

3.3. Weitere Anforderungen

Aufzeichnungen als Eingabe

Die Aufzeichnungen der Daten sollen die in den Benutzergruppen beschriebenen Anwendungsfälle abdecken, das heißt Echtzeitdaten und Aufzeichnungen gleichgestellt sein. Dies soll an jeder Stelle in der Architektur möglich sein. Ein wichtiges Element in der Entwicklung von Software ist aber auch Reproduzierbarkeit. Um Testfälle mit bekannten Erwartungswerten schreiben zu können, muss man auch konstante Eingaben haben. Das System soll es ermöglichen, aufgezeichnete Datensätze in der selben Art anzubieten, wie real verbundene Sensoren. Dadurch kann unter anderem reproduzierbar die Funktionalität der entwickelten Anwendungen getestet werden.

Arten der Datenübertragung

Es fallen Daten an, die in zwei Kategorien eingeteilt werden können. Zum einen entstehen Daten, die nicht zwingend in Echtzeit übertragen werden müssen. Beispiele dafür wären Langzeit-Durchschnittswerte des Pulses oder die Körpertemperatur. Für andere Daten ist es notwendig, dass

diese so schnell wie möglich übertragen werden. Darunter fallen je nach Anwendungsfall beispielsweise Daten von Beschleunigungssensoren. Die Architektur soll ermöglichen beide Arten von Datenübertragungen zu unterstützen.

Rückkanal

Um den Schritt von passiven intelligenten Materialien zu aktiven oder sehr intelligenten Materialien zu machen, ist es notwendig Daten an diese zurückzuliefern und sie gegebenenfalls zu steuern. Die Architektur soll diesen Anwendungsfall unterstützen. Kleidung könnte auch nicht nur als Sensor dienen sondern auch als Elemente beinhalten, die als Ausgabe fungieren, wie zum Beispiel in *Electroluminescent based Flexible Screen* [ACCL14] beschrieben. Für Anwendungsfälle ähnlich zu diesem, soll mit dem Sensor beziehungsweise der Kleidung kommuniziert werden können, falls diese es unterstützt oder erfordert.

Sicherheitsanforderungen

Die gesamte Architektur sollte die persönlichen Daten der Endbenutzer soweit möglich vor unberechtigten Dritten schützen. Unter persönliche Daten fallen sowohl die von Sensoren erfassten Daten als auch die verarbeitete Ausgabe. Nur der Benutzer, beziehungsweise von ihm zur Datenverarbeitung beauftragte Parteien sollen Zugriff auf diese Daten haben.

Aufgezeichneten Daten, wie zum Beispiel Puls und EKG, lassen unter Umständen Rückschlüsse auf den Gesundheitszustand des Trägers zu. *§203, Verletzung von Privatgeheimnissen* im Strafgesetzbuch[1] schützt patientenbezogene Gesundheitsdaten, unter welche die Aufzeichnung von physiologischen Werten durch beispielsweise Kleidung oder andere Wearable Computing Geräte auch fallen können.

3.4. Zusammenfassung

Eine Architektur für intelligente Kleidung soll vor allem die im Entwicklungsprozess beteiligten Gruppen unterstützen. Dabei sollte sie aber auch ermöglichen, verschiedene Komponenten von unbeteiligten Parteien erstellen zu lassen. Eine Schichtenarchitektur bietet sich an, um die Kopplung zwischen den Komponenten zu verringern. Die in dieser Arbeit betrachteten Benutzer der Architektur sind in drei Gruppen einzuteilen. Für jede dieser Gruppe sollte die Architektur Vorteile bieten. Neben dem Nutzen für die Gruppen sind auch Grundlegende Funktionalitäten wie bidirektionale Kommunikation und Sicherheitsanforderungen zur Umsetzung des Datenschutzes zu beachten.

[1] `http://dejure.org/gesetze/StGB/203.html`, zuletzt abgerufen am 30. Juni 2014

4. Architektur einer Middleware zur Entwicklungsunterstützung

In diesem Kapitel wird eine Architektur beschrieben, welche die in Kapitel 3 beschriebenen Anforderungen erfüllt. In Abbildung 4.1 sind die groben Komponenten davon visualisiert. Grundlage dieser Architektur sind die in Abschnitt 2.7 beschriebenen Überlegungen. Der bedeutende Unterschied zur dort vorgestellten Architektur ist eine zusätzlich eingeführte Schicht, die für die Speicherung, Verwaltung und Verarbeitung der Daten zuständig ist.

Abbildung 4.2 zeigt die selben Komponenten, allerdings mit Fokus auf die Kommunikationsgrenzen und Zuständigkeiten. Im folgenden werden die einzelnen Teile der Architektur näher beschrieben und ihre Ausgestaltung begründet.

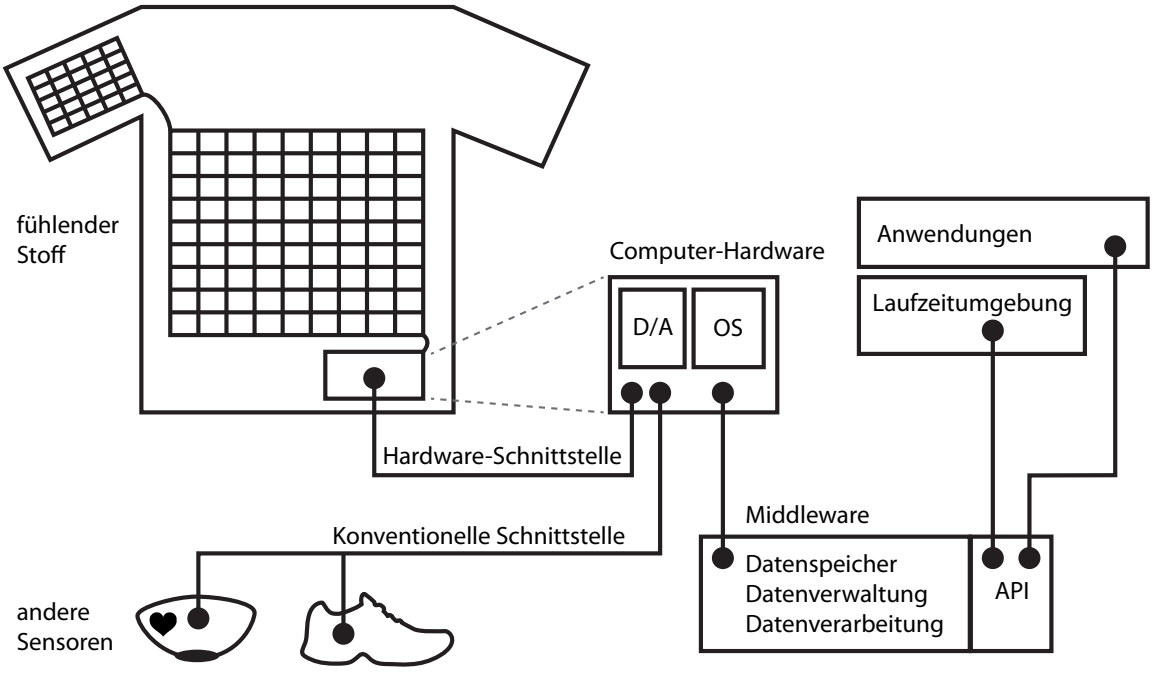


Abbildung 4.1.: Übersicht über das Zusammenspiel der einzelnen Komponenten

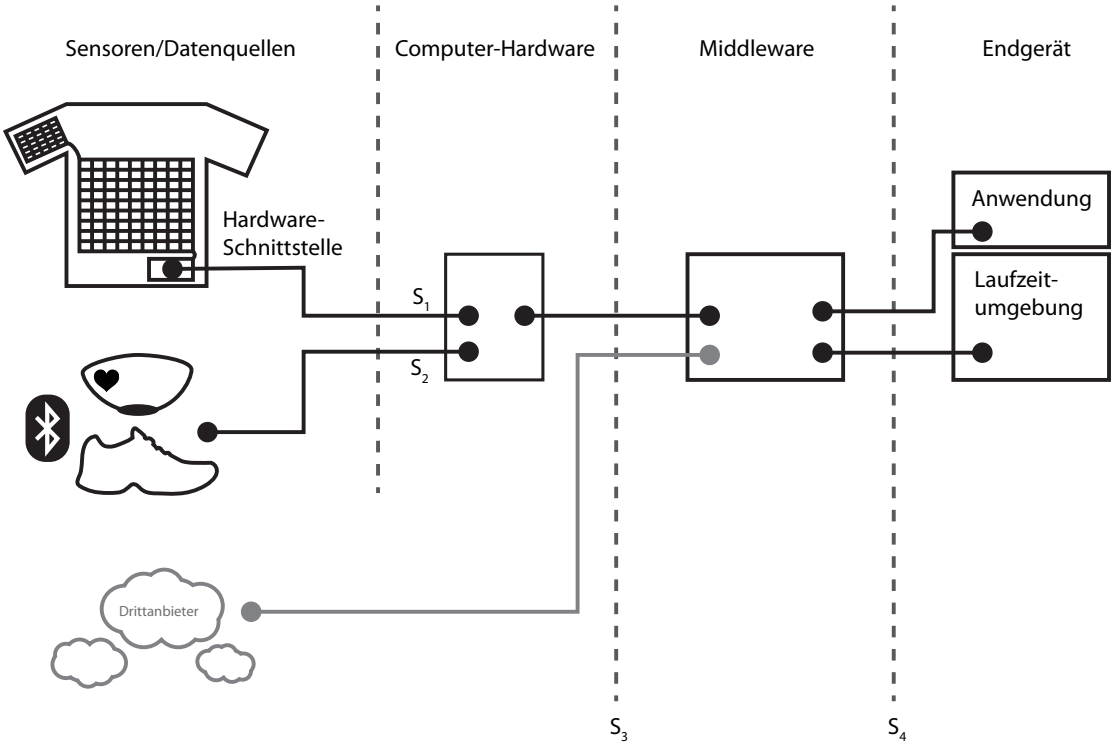


Abbildung 4.2.: Übersicht über die Architektur mit Fokus auf die verschiedenen Schnittstellen zwischen Gerätegrenzen

4.1. Hardware-Schnittstelle und Computer-Hardware

Die spezifische Schnittstelle für die Verbindung der Computer-Hardware mit intelligenter Kleidung ist hier nicht genauer spezifiziert. Im Kapitel 5 werden allerdings Vorschläge zu möglichen Implementierungen gegeben.

Um im Moment auf dem Markt verfügbare Sensoren zu benutzen, muss zusätzliche Software geschrieben werden, um die Daten von diesen Sensoren in das System zu überführen. Diese Software läuft auf der Computer-Hardware. Dies bedeutet, dass diese mindestens über eine Bluetooth-Schnittstelle verfügen muss. Alle anderen Merkmale sind offen, solange sie den Anforderungen in Abschnitt 2.7 genügen.

In Abbildung 4.3 sind drei Möglichkeiten zur Gestaltung der Hardware-Schnittstelle dargestellt, die vor allem interessant sind, wenn mehrere intelligente Kleidungsstücke gleichzeitig getragen werden. Die Varianten werden in folgenden Abschnitten erläutert.

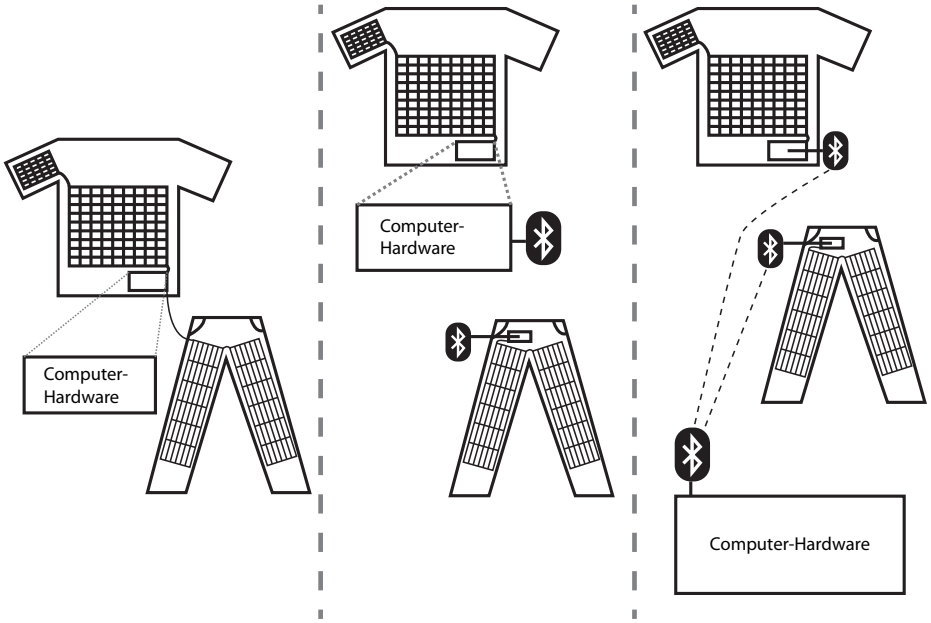


Abbildung 4.3.: Mögliche Ausgestaltungen der Hardwareschnittstelle

Verdrahtung zwischen Kleidungsstücken

In diesem Szenario ist die Computer-Hardware mit mit einer Steckverbindung nur an einem Kleidungsstück angebracht. Andere Kleidungsstücke können über flexible waschbare Verbindungen mit diesem Kleidungsstück verbunden werden und müssen keinen eigenen Sockel für die Hardware bereitstellen. In diesem Fall würde es ein Hauptkleidungsstück geben. Nachteilig sind die kabelgebundenen Verbindungen, welche je nach Kleidungsstück und Anwendungsfall störend sein könnten.

Master/Slave Kleidungsstücke

Wie im ersten Szenario gibt es ein Kleidungsstück, welches für die hauptsächliche Kommunikation zuständig ist. In dieses Kleidungsstück kann die Computer-Hardware eingesetzt werden. Die Verbindung zu weiteren intelligenten Kleidungsstücken ist dabei allerdings nicht kabelgebunden, sondern wird über eine drahtlose Verbindung wie zum Beispiel Bluetooth realisiert. Für die drahtlos verbundenen Kleidungsstücke könnte Hardware ähnlich zum *Blidget*[1] zum Einsatz kommen: Kleine (25x25x6mm) programmierbare Chips, die drahtlos über Bluetooth Smart kommunizieren können. In diesem Zuge erhöht sich allerdings die Komplexität des Systems, da aus der reinen Hardware-Schnittstelle ein neues System von zwei miteinander kommunizierenden Komponenten wird.

[1] `http://blidget.hcilab.org/`, zuletzt abgerufen am 30. Juni 2014

Vollständig drahtlose Kommunikation

Als konsequenter Schritt zwischen der Mischform in Szenario 2, wäre auch denkbar die Computer-Hardware komplett von der Kleidung zu trennen und mit dieser nur kabellos zu kommunizieren. Falls es möglich ist, robuste Chips, die klein genug sind um dauerhaft, auch während dem Waschvorgang, in der Kleidung zu verbleiben, wäre dies eine komfortable Option. Die Aufgaben der Computer-Hardware könnte dabei beispielsweise ein Smartphone übernehmen. Diese Art der Verbindung ist im Moment für Wearable Computing Geräte am meisten verbreitet, da sie keine Anforderungen an andere Geräte stellt und direkt mit einer kompatiblen Gegenstelle kommuniziert. Die Herausforderung ist, die Hardware auf den Kleidungsstücken selbst, die die drahtlose Kommunikation regelt, so robust zu machen, dass man nicht pro Kleidungsstück jeweils zusätzliche Module zum Aufladen oder Waschen abnehmen muss und diese später wieder anbringen.

4.2. Wearable OS

Durch die Einführung der Zwischenschicht, sind die Anforderungen an das Betriebssystem, das auf der Computer-Hardware läuft, recht gering. Es muss Code ausführen können, um über die spezifizierten Kommunikationskanäle mit dem zentralen Datenspeicher, der Middleware, kommunizieren. Es ist zuständig der Middleware, die an der Hardware-Schnittstelle anliegenden Werte zu senden.

4.3. Middleware (Datenspeicher, Verwaltung und Verarbeitung)

Im folgenden wird diese Schicht auch auch Middleware genannt. Diese zusätzliche Schicht dient der weiteren Abstraktion zwischen Sensoren und Endgerät und erlaubt einfachere zukünftige Erweiterungen und Unterstützung während des Entwicklungsprozesses. Als Eingabe bekommt sie die Sensordaten von der Computer-Hardware oder Sensoren selbst geliefert. Die Middleware dient auch als API für Anwendungen des Endbenutzers. Als zukünftige Erweiterung könnten auch Daten von Drittanbietern, die im Internet über Schnittstellen von Drittanbietern zur Verfügung stehen, von der Middleware importiert werden oder direkt dort angeliefert werden.

Die Middleware könnte auch neben dem Wearable OS auch auf der Computer-Hardware ausgeführt werden. Zu frühen Testzwecken im Entwicklungszyklus kann dies ein gangbarer Weg sein. Langfristig ist es allerdings realistischer, diese Middleware auszulagern um die Anforderungen an die Computer-Hardware möglichst niedrig zu halten. Die Speicherung, Verwaltung und Verarbeitung von Sensordaten wird dann auf einem anderen eigenständigen Rechner ausgeführt, der weder direkt für das Sammeln der Sensordaten zuständig ist, noch die letztendlichen Anwendungen des Benutzers ausführt.

Es ist theoretisch auch möglich, die Middleware nur während den frühen Entwicklungsphasen zu und später aus den dort angefallenen eingepflegten Daten automatisch oder halbautomatisch Schnittstellen für verschiedene Plattformen zu generieren. Diese Variante ist allerdings in dieser Arbeit nicht vorgesehen, könnte allerdings eine zukünftige Erweiterung darstellen. Langfristig geht der Trend zum Auslagern von Datenverarbeitung. Ein Beispiel dafür ist beispielsweise die mobile

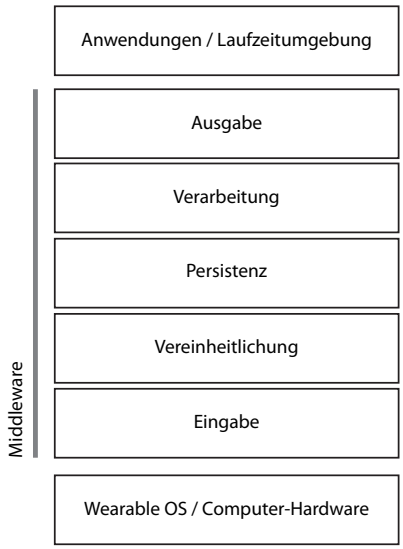


Abbildung 4.4.: Middleware mit umliegenden Schichten

Anwendung Moves[2], welche nur die Bewegungsdaten auf dem mobilen Endgerät sammelt und die Auswertung bei sich vornimmt und diese Ergebnisse dann zurücksendet.

Die einzelnen Schichten der Middleware (siehe Abbildung4.4) sind für folgende Aufgaben zuständig:

4.3.1. Eingabe

Die Eingabe ist zuständig um einkommende Daten zu empfangen. Sie stellt Endpunkte zur Verfügung mit denen sich Sensoren verbinden können um Daten anzuliefern. Zu Testzwecken, vor allem in frühen Phasen der Prototypen, die noch nicht in Echtzeit kommunizieren können, ist es von Vorteil auch Daten aus Dateien importieren zu können.

Als Eingabewerte können beliebige Daten benutzt werden, die zeitabhängig sind. Eine Datenquelle muss dabei keinem strikten Schema unterliegen, das heißt, nicht jeder Eintrag eines Sensors muss unbedingt alle Felder ausfüllen. Die einzige harte Anforderung an einen Eintrag ist der Zeitstempel. Beispielhafte Daten eines Sensors, die den Anforderungen genügen, sind in Tabelle 4.1 aufgeführt. Daten können dabei vom Datentyp `Integer`, `Float` oder `String` sein.

Aufgrund der möglichen Diversität von Daten soll es auch möglich sein, diese Felder hierarchisch zu organisieren, anstatt ähnliche Felder nur mit gleichem Präfix zu versehen. Ein Beispieldokument würde in JSON-Notation folgendermaßen aussehen: `{accel: {x: 1, y: 0, z: 0.5}, heart_rate: 67}`. Mit derartigen Datenstrukturen könnten ähnliche Daten, die aber beispielsweise von Verschiedenen Körperregionen gleichzeitig kommen, sinnvoll organisiert werden.

[2] `http://moves-app.com/`, zuletzt abgerufen am 30. Juni 2014

Zeitstempel	**Daten**			
timestamp	accel_x	accel_y	accel_z	heart_rate
2014-05-05T11:52:57.699Z	-0.6761	-0.1925	0.6749	
2014-05-05T11:52:57.901Z				67
2014-05-05T11:52:58.000Z	-0.6774	-0.1927	0.6701	71

Tabelle 4.1.: Beispieleingabe von schemalosen Daten

4.3.2. Persistenz

Aufgezeichnete Daten sollen im System gespeichert werden können. Dabei soll sowohl das Abspeichern als auch der Zugriff auf die Aufzeichnungen schnell genug erfolgen können, um sie ich Echtzeit wieder abspielen zu können. Allerdings sollten eingehende Daten keinen strikten Schema entsprechen müssen. Es sollen beliebige Schlüssel-Wert-Paare abgelegt werden können. Die einzige strikte Anforderung ist ein Zeitstempel, der jedem Paar zugeordnet ist.

4.3.3. Verarbeitung

Die Verarbeitung ist zuständig, die rohen Sensordaten nach ihrer Vereinheitlichung weiter zu verarbeiten. Dabei wird der eingehende Datenstrom durch eine Kombination von Filtern transformiert und dann an die Ausgabe weitergeleitet. Um die Entwicklung neuer Filter zu vereinfachen, soll das System den Entwicklern vorgefertigte Bausteine bereitstellen.

Der Großteil der Datenverarbeitung soll dabei auf der Middleware stattfinden, um den einzelnen Anwendungen, beziehungsweise Geräten auf denen diese Anwendungen ausgeführt werden, Last zu ersparen. Auch können die Algorithmen, die zur Verarbeitung der Daten auf Serverseite eingesetzt werden, aktualisiert und gegebenenfalls verbessert werden, ohne die Anwendung selbst aktualisieren zu müssen.

4.3.4. Ausgabe

Die Ausgabe ist zuständig um die im System angefallen Daten in einem maschinenlesbaren Format an andere Anwendungen weiterzuleiten. Dabei wird zwischen zwei Arten der Ausgabe unterschieden:

Als *Datenstrom* werden Daten angesehen, die so schnell wie möglich von der Quelle zur Anwendung gelangen sollen. Dabei soll die Verbindung bidirektional möglich sein, um einen eventuellen Rückkanal zum Sensor zu bieten.

Neben dem kontinuierlichen Datenstrom gibt es auch die Möglichkeit, nur einen *einzelnen Wert* abzufragen. Dies könnte beispielsweise der aktuelle Wert eines Sensors sein. Dieser Wert wird nach einem Frage-Antwort-Schema übermittelt. Dieser Wert kann auch Ergebnis einer Datenverarbeitung

sein. In diesem Fall hat die Antwort eventuell eine Verzögerungszeit, je nachdem wie groß der Puffer der Verarbeitung sein muss und auf wie vielen Daten er aufbaut.

Um nicht dauerhaft nach Änderungen zu fragen, können auch Rückrufaktionen registriert werden. Dabei kann ein Sensor, eine Bedingung und eine Rückrufadresse angegeben werden. Falls die Bedingung für den Sensor zutrifft, wird die Aktion an der spezifizierten Adresse aufgerufen.

Ein Bestandteil der Ausgabe ist optional auch die Entkopplung des spezifischen Sensors vom Konsument der Daten. Der Konsument kann eine Anfrage nach bestimmten Daten stellen, und die Middleware sucht daraufhin eine auf die Anforderung passende Quelle heraus. Die Anfrageparameter könnten dabei beispielsweise Anforderungen an bestimmte Felder (z.B. Puls, Atemfrequenz) sein. Die Middleware stellt aus den Anforderungen einen Datensatz zusammen und liefert diesen zurück.

4.3.5. Verwaltung

Dieses Modul ist hauptsächlich während der Entwicklung relevant. Es erlaubt die im System existenten Datensätze einzusehen. Für jeden Datensatz können die verfügbaren Felder eingesehen werden und die Daten auch visualisiert werden.

Neben einfachen Visualisierungen für einzelne Datensätze ist ein erweiterter Analysemodus angedacht, in dem mehrere Datensätze und das Ergebnis ihrer Verarbeitung (siehe Abschnitt 4.3.3) visualisiert werden können.

4.4. Schnittstelle zu Anwendungen und Laufzeitumgebung

Anwendungen können auf beliebige in der Middleware vorhandenen Sensordaten und Metadaten zugreifen. Der Zugriff auf einen bestimmten Sensoren ist von mehreren Anwendungen gleichzeitig möglich ohne diesen zu blockieren. Die Kommunikation sollte möglichst einfach auf verschiedenen Plattformen einsetzbar sein.

4.5. Authentifizierung und Autorisierung

Um unautorisiertes Abfragen von Sensordaten zu verhindern, müssen Verbindungen die im Zugriffsbereich von anderen Parteien liegen abgesichert werden. Dazu sollten Sensor und Empfänger gepaart werden und nur mit den bekannten Gegenstellen kommunizieren.

4.6. Vor- und Nachteile der Middleware

Die Einführung einer zusätzlichen Schicht bringt sowohl Vorteile als auch Nachteile mit sich. Mehr Schichten bedeutet mehr Indirektion und erhöhter Ressourcenverbrauch. Auf der Seite der Vorteile stehen allerdings Entkopplung, Flexibilität und die Funktionen zur Entwicklerunterstützung.

Es sind verschiedene Möglichkeiten denkbar, um in der Entwicklungsphase die Vorteile auszunutzen aber später im Endprodukt auf die Einbußen durch die Middleware zu verzichten, beziehungsweise diese zu verringern. Eine Möglichkeit wäre, Code für verschiedene Plattformen zu generieren, die an den Endpunkten die Arbeit übernimmt und damit die Middleware komplett ersetzt. Es wäre auch denkbar, einen Mischbetrieb zu erlauben und Daten die eine sehr geringe Latenz benötigen nicht durch die Middleware zu transportieren. Dabei könnten Endpunkte in einen speziellen direkten Modus gesetzt werden, der über die Middleware verhandelt wird, aber bei dem die Quelle und der Konsument direkt miteinander Daten austauschen.

4.7. Zusammenfassung

Eine Schichtenarchitektur ist eine gute Grundlage für die Gesamtarchitektur. Da jede Schicht eigene Zuständigkeiten hat, sind die darüber und darunter liegenden Komponenten flexibel und können je nach Bedarf auch unterschiedlich gestaltet sein.

Es wurde eine neue Schicht eingeführt, die zwischen den Anwendungen und der Computer-Hardware, die zuständig für die Kommunikation mit den Sensoren selbst ist, liegt. Diese Schicht übernimmt die Verarbeitung und Bereitstellung der Daten. Während dem Entwicklungsprozess bietet sie Unterstützung in den einzelnen Schritten der Datenerfassung, Analyse und Verarbeitung.

5. Eingesetzte Technologien und Umsetzung

> Nichts behebt Designprobleme wie eine Implementierung.
>
> *(J. D. Horton)*

Dieses Kapitel beschreibt eine mögliche Implementierung der Architektur. Die eingesetzten Technologien sind in Abbildung 5.1 veranschaulicht. Die dort grau dargestellten Teile konnten nicht implementiert werden, da die Komponenten nicht verfügbar waren. Anstatt der Kleidung wurde unter anderem der BioHarness 3 als Datenquelle implementiert und die Computer-Hardware mit Wearable OS entspricht gängiger Computer-Hardware und Betriebssystemen. In den folgenden Abschnitten werden die einzelnen Komponenten und Technologien näher erläutert.

Es sind nicht alle in der Architektur genannten Funktionen vollständig implementiert, sondern hauptsächlich Funktionen die für die Grundfunktionalität von Bedeutung sind und in der Evaluation eine Rolle spielen. Ein nicht implementierter Teile ist beispielsweise der gesamte Bereich der Authentifizierung und Autorisierung von Benutzern des Systems.

5.1. Computer-Hardware und Betriebssystem

Es wurde Wert auf plattformunabhängige Komponenten gelegt und somit sollte die umgesetzte Architektur auf allen gängigen Betriebssystemen (Windows, Linux/Unix) funktionieren. Die am stärksten plattformspezifische Komponente ist der Wrapper, der für die Kommunikation über Bluetooth mit dem BioHarness 3 zuständig ist.

Als Sprache wurde Python[1] gewählt. Python ist eine stark und dynamisch typisierte Programmiersprache, die Programmlesbarkeit in den Vordergrund stellt. Python ist auf Windows, Linux/Unix, Mac OS X und anderen Betriebssystemen verfügbar.

Python wurde dabei in der Version 3.x eingesetzt. Diese ist nicht vollständig rückwärtskompatibel mit Quellcode der für Python 2.x geschrieben wurde. Alle im Zuge dieser Arbeit angefertigten Softwareartefakte sind auf Kompatibilität mit Python 3 ausgerichtet.

Als mögliche Computer-Hardware, vor allem während der Prototypenerstellung, könnten beispielsweise Plattformen wie der Raspberry-Pi oder ein Beagleboard zum Einsatz kommen. Alle eingesetzten Technologien werden auf diesen Plattformen unterstützt.

[1] `https://www.python.org/`

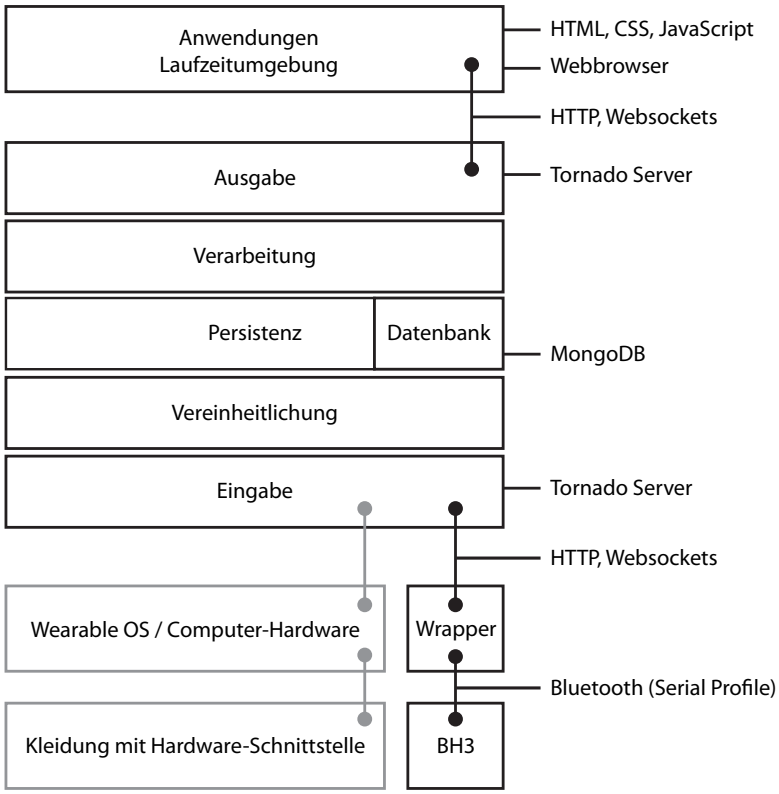


Abbildung 5.1.: Bei der Implementierung eingesetzte Technologien

5.2. Abhängigkeiten

Die folgenden Anwendungen und Bibliotheken sind notwendig, um die entwickelte Software auszuführen.

5.2.1. Tornado

Tornado[2] ist eine in Python geschriebene Bibliothek um Netzwerkanwendungen zu entwickeln. Sie wurde für das soziale Netzwerk FriendFeed[3] entwickelt. Die Funktionalität von FriendFeed liegt darin, in Echtzeit Neuigkeiten von Freunden zu aggregieren und anderen Nutzern einheitlich zu präsentieren.

[2] `http://www.tornadoweb.org`
[3] `http://friendfeed.com/`

Tornado setzt auf asynchrone Kommunikation. Konventionelle Serverimplementierungen wie zum Beispiel Django[4], benutzen ein synchrones Verarbeitungsmodel. Das bedeutet, dass eine neue Anfrage erst beantwortet wird, wenn die im Moment schon laufende Anfrage abgearbeitet wurde.

1. Der Server empfängt eine Anfrage und leitet sie an den entsprechenden Handler weiter
2. Der Handler führt eine komplexe Datenbankabfrage durch
3. Wenn die Behandlung abgeschlossen ist wird die Antwort gesendet
4. Es kann eine neue Anfrage angenommen werden

Während der Handler ausgeführt wird, blockiert der Server und es können keine neuen Anfragen angenommen werden. Beim asynchronen Modell hingegen wird dieses Blockieren aufgehoben. Dazu werden sogenannte Rückrufaktionen (Callbacks) benutzt. Dabei wird der länger laufenden Aufgabe eine Aktion mitgegeben die nach ihrem Abschluss aufgerufen wird. Um moderne Webtechnologien wie zum Beispiel WebSockets umzusetzen, ist es notwendig, die Anfragen asynchron zu beantworten.

1. Der Server empfängt eine Anfrage und leitet sie an den zuständigen Handler weiter
2. Der Handler setzt die Datenbankabfrage mit einer Rückrufaktion ab. Die Rückrufaktion enthält die Informationen um die Antwort zu erstellen.
3. Während die Datenbank arbeitet, kann der Server nun neue Anfragen bearbeiten.
4. In der Zeit in der keine neuen Anfragen ankommen, können beispielsweise die Rückrufaktionen ausgeführt werden

Die Koordination wird mittels einer Ereignisschleife umgesetzt. Diese läuft in einem Thread ab. Da der Entwickler nicht selbst mit Threads arbeiten muss, um asynchrones Verhalten zu erhalten, wird diese Klasse von Fehlern begrenzt. Um eine Tornado-Anwendung zu skalieren, können automatisch mehrere Prozesse gestartet werden, die jeweils eine eigene Ereignisschleife besitzen. Der eingehende Netzwerkverkehr wird auf die Schleifen aufgeteilt. Im Produktivbetrieb ist es gängig, eine Schleife pro CPU zu starten.

Neben der asynchronen Ein- und Ausgabe bringt Tornado noch Werkzeuge zum schreiben von Servern und Webanwendungen mit. Die Werkzeuge für asynchrone Netzwerkkommunikation enthalten die Ereignisschleife, einen asynchronen HTTP-Client und TCP-Server. Um Webanwendungen zu schreiben, wird ein asynchroner HTTP-Server und eine Templatesprache mitgeliefert. Zur Integration mit Drittdiensten werden beispielsweise OpenID- und OAuth-Logins unterstützt.

[4] `https://www.djangoproject.com/`

5.2.2. MongoDB

MongoDB ist eine Datenbank die dokumentenorientiert arbeitet. Im Gegensatz zu relationalen Datenbank wie MySQL ist sie schemafrei. In einem MongoDB-Server können mehrere Datenbanken nebeneinander existieren. Eine Datenbank enthält Dokumentensammlungen (*Collections*), in denen Dokumente abgelegt werden können. Ein Dokument ist ein nach dem BSON-Format serialisiertes Objekt.

JSON[5] ist ein Datenformat um Schlüssel-Wert-Paare in einem menschenlesbaren Format zu übertragen. Im Vergleich zu XML (Extensible Markup Language) ist die Syntax einfacher gestaltet und enthält in der Regel weniger Verwaltungsdaten, definiert dafür allerdings nur weniger Datentypen.

BSON ist eine Erweiterung von JSON und steht für Binary JSON. Es erweitert JSON in zwei Bereichen. Es speichert zum serialisierten Objekt jeweils noch den Datentyp und die Länge des Eintrags ab. Durch die gespeicherte Länge des Eintrags kann die Geschwindigkeit beim parsen des Eintrags erhöht werden.

5.2.3. Andere Python Bibliotheken

Hier folgt eine kurze Beschreibung von anderen eingesetzten Python-Bibliotheken.

motor ist zuständig für die Kommunikation zwischen Tornado und MongoDB. Der Datenbanktreiber ist asynchron für den Einsatz mit Tornado entworfen worden.

toro stellt Primitive zur Verfügung, um asynchrone Abläufe zu synchronisieren.

werkzeug ist eine Bibliothek für Webanwendungen. Sie enthält einen Debugger für Webanwendungen, der in der Middleware integriert wurde.

pyserial bietet Unterstützung für serielle Verbindungen nach dem RS-232 Standard. Die serielle Bluetoothverbindung wird über *pyserial* benutzt.

5.2.4. Sensorspezifische Abhängigkeiten Abhängigkeiten

Um mit dem BioHarness 3 in Echtzeit zu kommunizieren, wurde eine existierende Bibliothek angepasst und integriert. Die Bibliothek namens *zephyr-bt*[6] wurde entwickelt, um durch übermittelte Biowerte des Trägers in Echtzeit Musik zu erzeugen. Im Zuge dieser Arbeit mussten Anpassungen an der Bibliothek vorgenommen werden, um sie kompatibel mit Python 3 zu machen.

[5] `http://json.org/`, zuletzt abgerufen am 30. Juni 2014
[6] `https://github.com/jpaalasm/zephyr-bt`, zuletzt abgerufen am 30. Juni 2014

5.2.5. Bibliotheken zur Umsetzung der Benutzungsschnittstelle

Die Benutzungsoberfläche wurde mit Webtechnologien umgesetzt. Die Webseiten werden von der Middleware generiert. Als Grundlage für die Gestaltung wurde das CSS-Framework *Bootstrap*[7] eingesetzt. Logik die auf Klientenseite abläuft ist in JavaScript geschrieben. Für eine einfachere Benutzung des DOMs im Webbrowser wird *jQuery*[8] eingesetzt. Um Graphen zu zeichnen wurde zunächst der Einsatz *D3.js*[9] getestet. Für einfache Graphen ist die notwendige Einarbeitung und Komplexität allerdings zu hoch. Daher sind die Graphen jetzt mithilfe von *Google Charts*[10] umgesetzt.

5.3. Kommunikationsschnittstellen

Im folgenden wird die Umsetzung der Schnittstellen zwischen den verschiedenen Komponenten näher beschrieben. Dabei wird mit den Verbindungsbezeichnungen auf Abbildung 4.2, die Übersicht der Schnittstellen zwischen Gerätegrenzen, Bezug genommen.

Hardware-Schnittstelle

Diese Verbindung (S_1) ist zuständig um die Daten beispielsweise eines drucksensitiven Gitters auf dem T-Shirt mit der Computer-Hardware zu verknüpfen. Da es dafür noch keine benutzbaren Prototypen gab, wurde diese Schnittstelle nicht implementiert, sondern durch eine Schnittstelle mit anderen Geräten ersetzt.

Verbindung mit Altgeräten

Im Moment auf dem Markt befindliche Wearables kommunizieren hauptsächlich über Bluetooth. Daher bietet es sich an, diese über Bluetooth mit der Computer-Hardware koppeln zu können (S_2). Die Software auf der Computer-Hardware kümmert sich dann darum, diese Daten über die Schnittstelle S_3 an die Middleware zu senden. Exemplarisch wurde dies für den Zepyhr BioHarness 3 implementiert (siehe auch Abschnitt 5.4.1).

Kommunikation mit der Middleware

Die Middleware als zentraler Punkt, in dem Daten ein- und ausgegeben werden, ist über ein Netzwerkprotokoll erreichbar. Im Rahmen dieser Arbeit wurde für die Kommunikation zwischen Computer-Hardware und Middleware (S_3) und zwischen Middleware und Endgerät (S_4) das gleiche Protokoll

[7] http://getbootstrap.com/
[8] http://jquery.com/
[9] http://d3js.org/
[10] https://developers.google.com/chart/

benutzt. Es ist allerdings möglich dieses je nach Anforderungen, die vor allem von Seiten der noch nicht spezifizierten Computer-Hardware kommen können, zu ersetzen.

Als Protokoll um mit der Middleware zu kommunizieren, wird HTTP (Hypertext Transfer Protokoll) und WebSocket eingesetzt. HTTP wird überall da eingesetzt, wo der Nachrichtenaustausch dem Request-Response-Muster folgt: das heißt auf eine *Anfrage* des Clients folgt eine *Antwort* des Servers.

Um eine dauerhaft Verbindung zwischen zwei Endpunkten aufzubauen, wird das WebSocket-Protokoll benutzt. Das Protokoll wurde 2011 standardisiert [Int11]. Es ist darauf ausgelegt, für die Kommunikation zwischen Webbrowsern und Webservern benutzt zu werden, aber nicht darauf beschränkt. Es bietet eine Vollduplex-Verbindung zwischen beiden Parteien an. Das heißt Client und Server können gleichzeitig Nachrichten austauschen. Es ist damit im Gegensatz zum Request-Response-Muster besser geeignet um Daten in Echtzeit auszutauschen. Die Nutzdaten der über WebSockets übertragenen Nachrichten sind dabei im JSON-Format kodiert.

Der Einsatz der Webtechnologien HTTP und WebSockets erlaubt es, Anwendungen mithilfe von HTML, CSS und JavaScript zu erstellen. „Webtechnologien setzen sich vor allem in Bereichen durch in denen verschiedene Plattformen mit wenig Aufwand unterstützt werden wollen“, meint Herr Dipl.Inf Fellger, Softwareentwickler in einem Unternehmen, das sich auf die Entwicklung von Softwarelösungen für Unternehmen spezialisiert hat. Nach Charland und Leroux ist es günstiger, webbasierte Anwendungen zu entwickeln, wenn man dabei in Kauf nimmt, dass die Benutzungsschnittstelle je nach Anwendungsfall nicht die Leistungsfähigkeit einer nativen Anwendung erreichen wird [CL11].

Die Benutzung von WebSockets und HTTP-Schnittstellen ist allerdings auf allen gängigen mobilen Plattformen (iOS, Android, Windows Phone) auch mit nativen Programmcode möglich und so können auch native Anwendungen geschrieben werden, die auf die selben Daten zugreifen und eine optimierte plattformspezifische Benutzungsoberfläche anbieten.

5.4. Umsetzung der Middleware

Die Middleware wurde mit Hilfe der in Abschnitt 5.2 erwähnten Abhängigen umgesetzt. Die Middleware wird im folgenden als `SEDM` bezeichnet. Dieses Akronym steht für *Se*nsor*DataM*anagement. `SEDM` besteht aus 4 Modulen die jeweils einen Bereich der Funktionalität realisieren. Die im folgenden aufgezählten Module werden in jeweils eigenen Abschnitten detaillierter beschrieben.

`sedm.sources` Dieses Modul beinhaltet Code um mit spezifischen Sensoren zu sprechen und deren Daten zu importieren.

`sedm.processing` kümmert sich um die Verarbeitung der Sensordaten.

`sedm.storage` ist für die Persistenz der Daten zuständig.

`sedm.server` ist das Bindeglied zwischen den anderen Modulen und ist für die Gesamtkoordination und Kommunikation zuständig.

Listing 5.1 Die Definition der Rückrufaktion, die Daten des BH3 zum SEDM-Server sendet. Die Verbindung zum Websocket `ws_connection` wurde bei Programmstart initialisiert.

```
def callback(value_name, value):
   if value_name == 'ecg': # ignore ECG values
      return
   elif value_name == 'acceleration': # flatten accelerometer data
      message = {
         'timestamp': dt.now(),
         'accel_x': value[0],
         'accel_y': value[1],
         'accel_z': value[2],
      }
   else:
      message = {
         'timestamp': dt.now(),
         value_name: value,
      }
   ws_connection.write_message(json_dumps(message))
```

5.4.1. Eingabe

Um den Prototypen zu implementieren wurde auf dem Markt verfügbare Hardware eingesetzt. Im folgenden wird die benutzte Hardware und ihre Benutzung als Quelle näher beschrieben.

BioHarness 3

Die grundlegenden Funktionen des BioHarness 3 (BH3) sind in Abschnitt 2.10.1 näher erläutert. Er wurde als eines der möglichen Datenquellen implementiert und liefert Beschleunigungsdaten, Puls und Atemfrequenz des Trägers. Der BH3 spricht ein serielles Protokoll, das über USB und Bluetooth gleich aufgebaut ist. Die genaue Spezifikation der Nachrichten, die über das serielle Protokoll gesprochen werden, sind in einem Dokument des SDKs näher spezifiziert [Zep11b].

Um über Bluetooth mit dem BH3 zu kommunizieren, wurde die in 5.2.4 beschriebene Bibliothek zepyhr-bt eingesetzt. Diese liefert in Echtzeit einen Strom von physiologischen Werten. Es wurde ein kleines Wrapper-Programm geschrieben, dass diese Bibliothek benutzt und die Daten im richtigen Format an den `SEDM`-Server sendet. Listing 5.1 zeigt, wie Werte vor dem Senden noch bearbeitet werden. Beispielsweise werden die von *zephyr-bt* als Tripel gelieferten Daten des Beschleunigungssensors in eine flache Struktur verpackt. Die EKG-Werte werden nicht gesendet, sondern ignoriert. Die letztendliche Nachricht wird dann ins JSON-Format überführt und über eine WebSocket-Verbindung gesendet.

Um die auf dem BH3 gespeicherten Logdaten auszulesen, gibt es nur eine für Windows verfügbare Software die von Zephyr bereitgestellt wird. Um aufgezeichnete Daten des Geräts auf auf anderen Betriebssystemen auszulesen, wurde das in Abschnitt 5.9 beschriebene Artefakt entwickelt.

Bei ersten Testläufen ist die Verbindung zum BH3 reproduzierbar nach 10 Minuten abgebrochen. Das Problem lag in der zephyr-bt Bibliothek, welche eine Nachricht zur Aufrechterhaltung der Bluetooth-

Listing 5.2 Beispiel des Sendens von Beschleunigungsdaten über JavaScript.

```
var samplesPerSecond = 30;
var sampleDelay = 1000 / samplesPerSecond;

$('#start').click(function(evt){
      $('#start').hide();
      $('#stop').show();
      var ax, ay, az;
      window.ondevicemotion = function(event) {
            ax = event.accelerationIncludingGravity.x;
            ay = event.accelerationIncludingGravity.y;
            az = event.accelerationIncludingGravity.z;
      };
      intervalId = setInterval(function() {
            ws.send(JSON.stringify({
                  'accel_x': ax,
                  'accel_y': ay,
                  'accel_z': az
            }));
      }, sampleDelay);
});
```

Verbindung nicht sendet. Ein möglicher Weg dieses Problem zu lösen ist, die Zeit bis zum Abschalten auf 0 zu konfigurieren. Um einfache Konfigurationsnachrichten an das BH3 zu senden, wurde ein in 5.9 genauer beschriebenes Werkzeug entwickelt.

Bewegungssensoren eines Smartphones

Auf einigen mobilen Geräten ist es möglich, die Beschleunigungssensoren im Webbrowser abzufragen. Dazu muss die Hardware und der Webbrowser dies unterstützen. Sofern unterstützt, wird bei der Änderung der Geräteorientierung ein `DeviceMotionEvent`[11] geworfen. Dieses Ereignis enthält die Beschleunigung des Geräts in kartesischen Koordinaten, jeweils mit oder ohne Berücksichtigung der Gravitation.

Es wurde eine Beispielanwendung implementiert, die im Webbrowser eines Smartphones ausgeführt werden kann. Diese Anwendung sendet die Beschleunigungsdaten über einen WebSocket an die Middleware. Die Daten werden nicht bei jeder Änderung gesendet sondern es wird immer nach Ablauf einer bestimmten Zeitspanne der aktuelle Wert übertragen. Im Beispiel, siehe Listing 5.2, werden 30 Werte pro Sekunde gesendet. Abbildung 5.7b zeigt die Anwendung auf einem Smartphone.

Um die korrekte Übertragung und Leistungsfähigkeit zu testen wurde eine Anwendung erstellt, die diese Werte live oder aus einer Aufzeichnung visualisiert. Die Beschleunigung wird in eine die Rotationsbewegung eines dreidimensionalen Würfels umgesetzt.

[11] `http://w3c.github.io/deviceorientation/spec-source-orientation.html#devicemotion`, zuletzt abgerufen am 29. Mai 2014

Listing 5.3 Minimales Beispiel um Daten mit Python an die Middleware zu senden.

```
import json
from tornado.ioloop import IOLoop
from tornado import websocket

def create_connection(server, sid):
   return websocket.websocket_connect('ws://{}/input/live/{}.ws'.format(server, sid),
        connect_timeout=5)

def collect_input(connection_future):
   connection = connection_future.result()
   while True:
      message = input('>>')
      connection.write_message(json.dumps({'message': message}))

create_connection('10.0.0.46:8888', 'test-python-source-1').add_done_callback(collect_input)
IOLoop.instance().start()
```

Erstellen von Markierungen

Um Datensätze mit Annotationen zu versehen, wurde eine Anwendung entwickelt. Diese Anwendung wurde auch als Webanwendung umgesetzt. Beim öffnen der Anwendung wird ein neuer Websocket geöffnet und über den Druck von verschieden beschrifteten Schaltflächen wird der Wert mit dem aktuellen Zeitpunkt an `SEDM` übertragen. Die mit dieser Anwendung erstellten Annotationen können in der Analyseansicht eingeblendet werden und helfen, verschiedene Aktionen unterscheiden zu können, um dafür Erkennungsalgorithmen zu entwerfen.

Beispiel für beliebige Quellen

Zu Testzwecken gibt es auch eine minimale Python-Anwendung um Daten an `SEDM` zu senden. Der dafür zuständige Quellcode ist vollständig in Listing 5.3 dargestellt. Dabei wird die Websocket-Implementierung aus der Tornado-Bibliothek benutzt, um die Verbindung zum Server herzustellen. Das Programm sendet daraufhin Benutzereingaben von der Konsole an die Middleware.

5.4.2. Persistenz

Um Eingehende Daten zu speichern, wird die Datenbank MongoDB eingesetzt. Die Implementierung benutzt allerdings keine Funktionen die spezifisch diese Datenbank als Backend erfordern. Prinzipiell ist jeder Datenspeicher einsetzbar, der effizient Schlüssel-Wert-Paare abspeichern und auslesen kann und dabei kein festes Schema der Daten benötigt. MongoDB unterstützt auch geschachtelte Dokumente, um größere Dokumente besser strukturieren zu können.

Um die MongoDB-spezifische Implementierung zu abstrahieren, existieren Basisklassen welche die Schnittstelle zu kompatiblen Datenspeichern definieren. Diese sind in `sedm.storage.base` definiert.

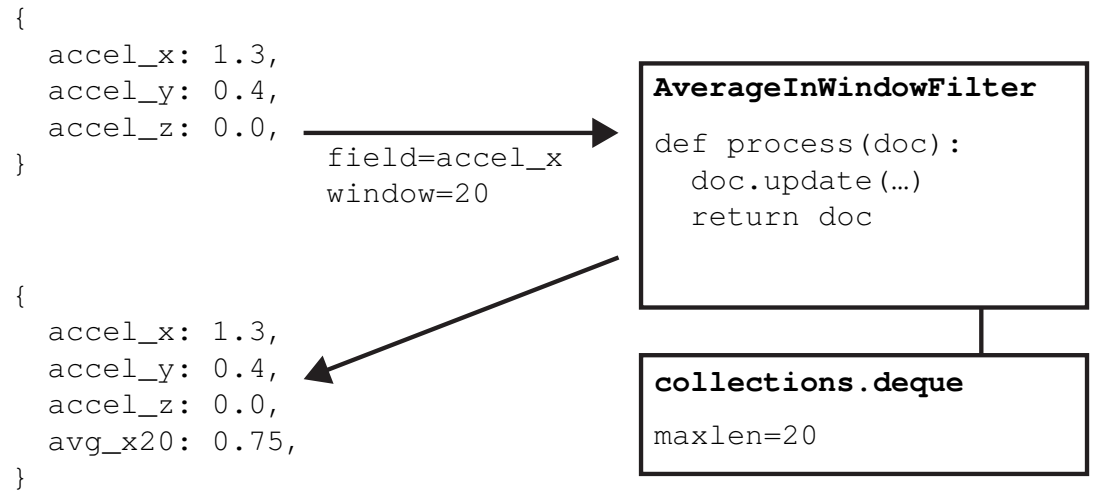


Abbildung 5.2.: Beispiel eines Filters mit ein- und ausgehenden Daten

Von den Basisklassen erbt jeweils eine spezifisch auf MongoDB ausgelegte Klasse, die in der gesamten Middleware benutzt wird. Um andere Backends zu unterstützen, müssten nur diese Klassen ausgetauscht werden.

Die Grundfunktionalität, existierende Datensätze wieder mit dem richtigen Zeitverhalten abzuspielen ist auch in dem Datensatz selbst implementiert, da dazu die Werte bei Bedarf aus der Datebank gelesen werden müssen. Es ist nicht praktikabel den kompletten Datensatz in den Arbeitsspeicher zu laden bevor er dann abgespielt wird. In der jetzigen Implementierung wird erst nach erfolgreichem ausführen der Rückrufaktion das nächste Dokument aus der Datenbank geladen.

5.4.3. Verarbeitung

Die Verarbeitung ist dafür zuständig, aus den Eingaben abhängige Werte zu berechnen. Filter sind für einzelne Aufgaben zuständig und können miteinander kombiniert werden. Die dafür zuständigen Klassen sind in `sedm.processing` enthalten. Das Modul selbst besteht aus zwei Teilen, den atomaren Filtern und Filterketten, welche diese Filter kombinieren.

Ein Filter besteht mindestens aus einer Methode, die ein eingehendes Dokument, das heißt eine Sammlung von Schlüssel-Wert-Paaren verarbeitet und eventuell modifiziert zurückgibt. Das modifizierte Dokument wird dann eventuell von weiteren Filtern verarbeitet. Dies ist in Abbildung 5.2 exemplarisch dargestellt. Beim Erzeugen des Filters wird der zu verarbeitende Feldname und die Fenstergröße initialisiert. Der Datenspeicher für die Werte innerhalb des Fensters ist hierbei mit einer *Deque* realisiert. Dies ist eine Datenstruktur ähnlich zu Stapelspeichern oder Warteschlangen, in der Daten an beiden Enden eingefügt und entfernt werden können. Hier ist Länge auf die Fenster beschränkt und alte Elemente werden automatisch beim Einfügen von neuen Elementen gelöscht.

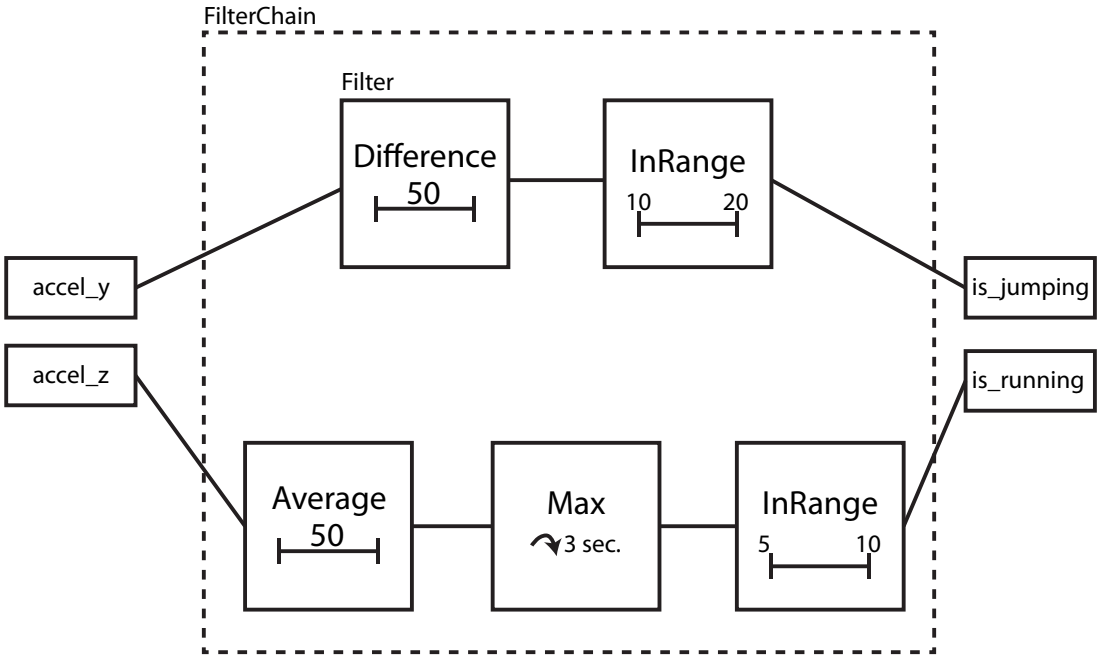


Abbildung 5.3.: Beispiel einer Filterkette. Die rohen Werte der Quelle werden durch verschiedene Filter verarbeitet. Ausgaben eines Filters können die Eingabe für einen anderen Filter sein. Die Filter sind im Code durch eine Filterkette gruppiert. Das Ergebnis sind hier zwei neue Felder, die beispielsweise in einer Anwendung anstatt den rohen Beschleunigungsdaten benutzt werden können.

Eine Filterkette definiert eine bestimmte Konfiguration von hintereinandergeschaltenen Filtern und kann diese auch einheitlich bei der Initialisierung der Kette konfigurieren. Die Filterkette kapselt damit eine bestimmte Funktionalität. Abbildung 5.3 zeigt eine Beispielhafte Filterkette die aus einzelnen Filtern besteht. Sie erwartet zwei Eingabewerte und berechnet aus diesen zwei Ausgabewerten. Der dargestellte Filter ist Abschnitt 5.7 näher beschrieben.

5.5. Maschinenlesbare Ausgabe / API

Über die API können Anwendungen auf die im System vorliegenden Daten zugreifen. Der Zugriff selbst erfolgt dabei über WebSockets, wie in Abschnitt 5.3 genauer beschrieben. Dabei wird jeweils mit einem Endpunkt kommuniziert, der von der Middleware bereitgestellt wird. Es kann entweder auf historische Daten oder Live-Daten zugegriffen werden.

Alle Aufzeichnungen von einzelnen Sensoren sind unter jeweils einem Endpunkt erreichbar. Dabei wird der entsprechende Datensatz aus der Datenbank gelesen und im richtigen Zeitverhalten gesendet. Dafür ist `sedm.server.handlers.historic.HistoricDataSocket` zuständig. Die zu benutzende Filterkette kann dabei zur Laufzeit bei der Initialisierung des Sockets angegeben werden.

Der Zugriff auf Sensoren die in Echtzeit kommunizieren ist auf der Seite des Servers mittels dem Beobachter-Muster umgesetzt. Es ist möglich, dass mehrere Anwendungen gleichzeitig die Eingaben eines Sensors abrufen wollen, dazu registrieren sie sich bei diesem Sensor. Bei neuen Werten beliefert dieser dann alle interessierten Klienten unter Berücksichtigung der gewünschten Filter mit diesen Daten. Jeder einzelne Klient kann dabei spezifizieren, ob und welche Filterkette benutzt werden soll.

Listing 5.4 zeigt eine beispielhafte Benutzung der API. Die URL des Websockets gibt an, welche Daten genutzt werden sollen. In diesem Fall werden Echtzeitdaten des Sensors des Namens `accel-demo-1` benutzt. Dabei werden die Werte durch die Filterkette `jump_and_run` verarbeitet. Diese Filterkette reichert eingehende Beschleunigungsdaten mit den zusätzlichen Feldern `is_jumping` und `is_running` an. In einer Anwendung können diese Werte dann benutzt werden, um dem Benutzer Visualisierungen zu bieten. Im Beispiel werden verschiedene Bereiche auf der Webseite ein- oder ausgeblendet.

Die URLs um direkt mit Sensoren beziehungsweise Aufzeichnungen zu kommunizieren sind wie folgt aufgebaut:

Senden von neuen Daten: `ws://host:port/input/live/<sensor_id>.ws`

Zugriff auf Aufzeichnung: `ws://host:port/output/historic/<sensor_id>.ws`

Zugriff auf aktuelle Daten: `ws://host:port/output/live/<sensor_id>.ws`

Um die Kommunikation vor Man-in-the-Middle-Angriffen und unerlaubter Ausspähung der Daten zu schützen kann auch eine sichere Verbindung mittels TLS (Transport Layer Security, auch Transportschichtsicherheit) aufgebaut werden. Dazu ist der Bezeichner für das Protokoll zu `wss` zu ändern. Um allerdings TLS benutzen zu können, muss die Anwendung korrekt mit privatem Schlüssel und Zertifikat initialisiert werden.

5.6. Analysewerkzeuge

Ein wichtiger Teil der Middleware sind Werkzeuge, die bei der Entwicklung von neuen Sensoren und Erkennungsmethoden Unterstützung bieten. Implementiert sind dabei Detailansichten von einzelnen Aufzeichnungen und eine Analyseansicht, um den Effekt von Filtern zu betrachten.

Details und Graphen von Aufzeichnungen

In der Detailansicht (siehe Abb. 5.4), werden hauptsächlich die Metadaten zu einer Aufzeichnung angezeigt. Die Metadaten sind beliebige Schlüssel-Wert-Paare von denen `created` und `last_modified` automatisch angelegt werden. Im Abschnitt *Data* wird die Größe des Datensatzes und alle in diesem Datensatz vorkommenden Felder angezeigt. Aus der Gesamtheit aller enthaltenen Dokumente werden

Listing 5.4 Beispiel der API-Nutzung in JavaScript.

```
$('#jump, #run').hide();

socket = new WebSocket('ws://sedm.dev:8888/output/live/accel-demo-1.ws?chain=jump_and_run');

socket.onmessage = function(message){
      data = JSON.parse(message.data);
      if(data.error){
            alert(data.error);
      } else if (data.info) {
            console.log(data.info);
      } else if(data.is_jumping) {
            $('#jump').show();
            $('#run').hide();
      } else if(data.is_running){
            $('#run').show();
            $('#jump').hide();
      } else {
            $('#jump, #run').hide()
      }
}
```

beispielhaft die ersten fünf in der Tabelle dargestellt. Auch wird auf weitere Ansichten auf die Daten verlinkt, beispielsweise zu den Graphen.

In dieser in Abbildung 5.5 gezeigten Ansicht können beliebige numerische Felder als Liniendiagramm visualisiert werden. Dabei können beliebig viele Felder ausgewählt werden. Da diese allerdings auf einer Achse dargestellt werden, bietet sich für weitergehende Analysen die dafür vorgesehene Ansicht an.

Analyseansicht

In der in in Abbildung 5.6 gezeigten Ansicht können Eingabewerte und das Ergebnis von eingesetzten Filtern visualisiert werden. Als Quelle können dabei Felder aus verschiedenen Datensätzen benutzt werden und einzelne Felder auch beliebig auf eine der zwei Y-Achsen gelegt werden. Neben den numerischen Feldern wird auch ein Datensatz unterstützt, der Markierungen enthält. Diese Markierungen werden neben den Daten in der Eingabe und Ausgabe angezeigt, um für den Anwendungsfall relevante Merkmale besser identifizieren zu können. Die Ausgabe wird durch Anwendung eines Filters oder einer Filterkette auf die Eingabedaten erzeugt und zeigt die diese errechneten Werten an.

5.7. Implementierte Filter und Erkennungen

Folgende grundlegenden Bausteine sind im System vorhanden um eigene Filterketten zusammenzustellen:

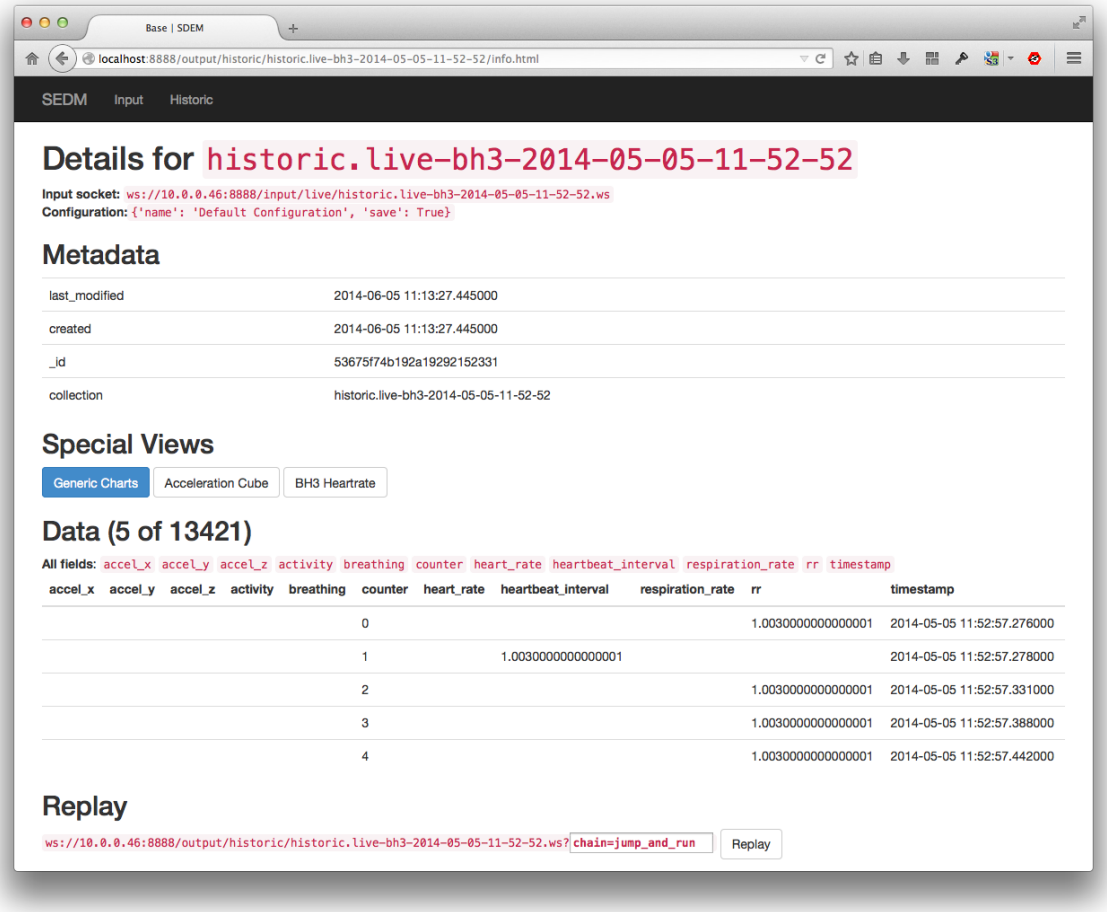


Abbildung 5.4.: Screenshot der Detailansicht

AverageFilter berechnet den Durchschnittswerte zweier Eingaben.

AverageInWindowFilter berechnet den Durchschnittswert einer Eingabe über ein spezifiziertes Fenster hinweg.

MaxFilter gibt den maximalen Wert des Felds zurück. Optional kann der Werte nach einer bestimmten verstrichenen Zeit zurückgesetzt werden.

DifferenceFilter gibt die Differenz zweier Eingabewerte zurück.

DifferenceInWindow gibt die maximale Differenz von Werten in einem Fenster zurück.

ProductFilter berechnet das Produkt zweier Eingabewerte.

InRangeFilter gibt 100 zurück, wenn die Eingabe in einem definierten Bereich liegt.

AndFilter gibt 100 zurück wenn beide Eingabewerte der Wert 100 haben.

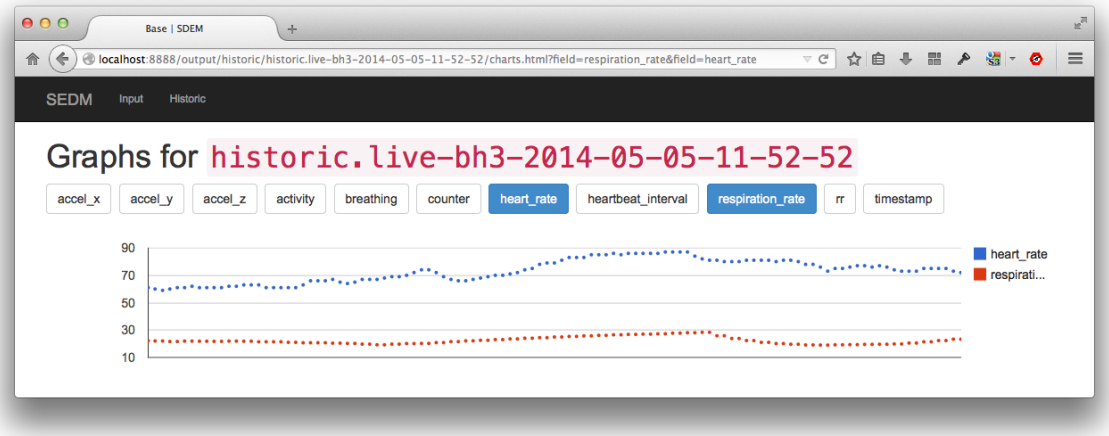


Abbildung 5.5.: Screenshot der Graphenansicht

Abbildung 5.6.: Screenshot der Analyseansicht

Hüpfen und Rennen

Als einfaches Beispiel wurde eine Filterkette definiert, die den Unterschied zwischen einer rennenden und hüpfenden Person feststellen soll. Als Eingabewerte erhält sie Beschleunigungsvektoren in hochwärts (Y-Achse) und vorwärts (Z-Achse). Als Merkmal um Hüpfen zu erkennen kann die Differenz des Y-Vektors über ein Fenster von 50 Werten benutzt werden. Liegt diese in einem hohen Bereich ist ein Hüpfen sehr wahrscheinlich.

Um Rennen zu erkennen kann der Maximalwert über die Z-Achse untersucht werden. Um kurze Ausreißer vorwärts zu dämpfen ist ein Durchschnittswert darüber notwendig. Um nicht dauerhaft Rennen zu erkennen, ist es auch notwendig den Maximalwert nach einer bestimmten Zeit zurückzusetzen.

5.8. Implementierte Beispielanwendungen

Als Beispiel um anliegende Daten zu visualisieren, wurden zwei Beispielanwendungen entwickelt. Beide dieser Anwendungen basieren auf HTML und JavaScript. Eine der Anwendungen baut auf der Filterkette für Hüpfen und Rennen auf und zeigt den aktuellen Zustand des Benutzers (Rennen, Hüpfen, Stillstand) und einen Zähler für die getanen Sprünge an. In Abbildung 5.7a ist die Anwendung dargestellt.

Die andere Anwendung nutzt WebGL um in einem Webbrowser einen dreidimensionalen Würfel anzuzeigen. Dieser Würfel dreht sich aufgrund von anliegenden Beschleunigungswerten. Die Anwendung ist gut geeignet, um die Leistungsfähigkeit der Quellen und des Netzwerks zu testen.

5.9. Weitere Software-Artefakte

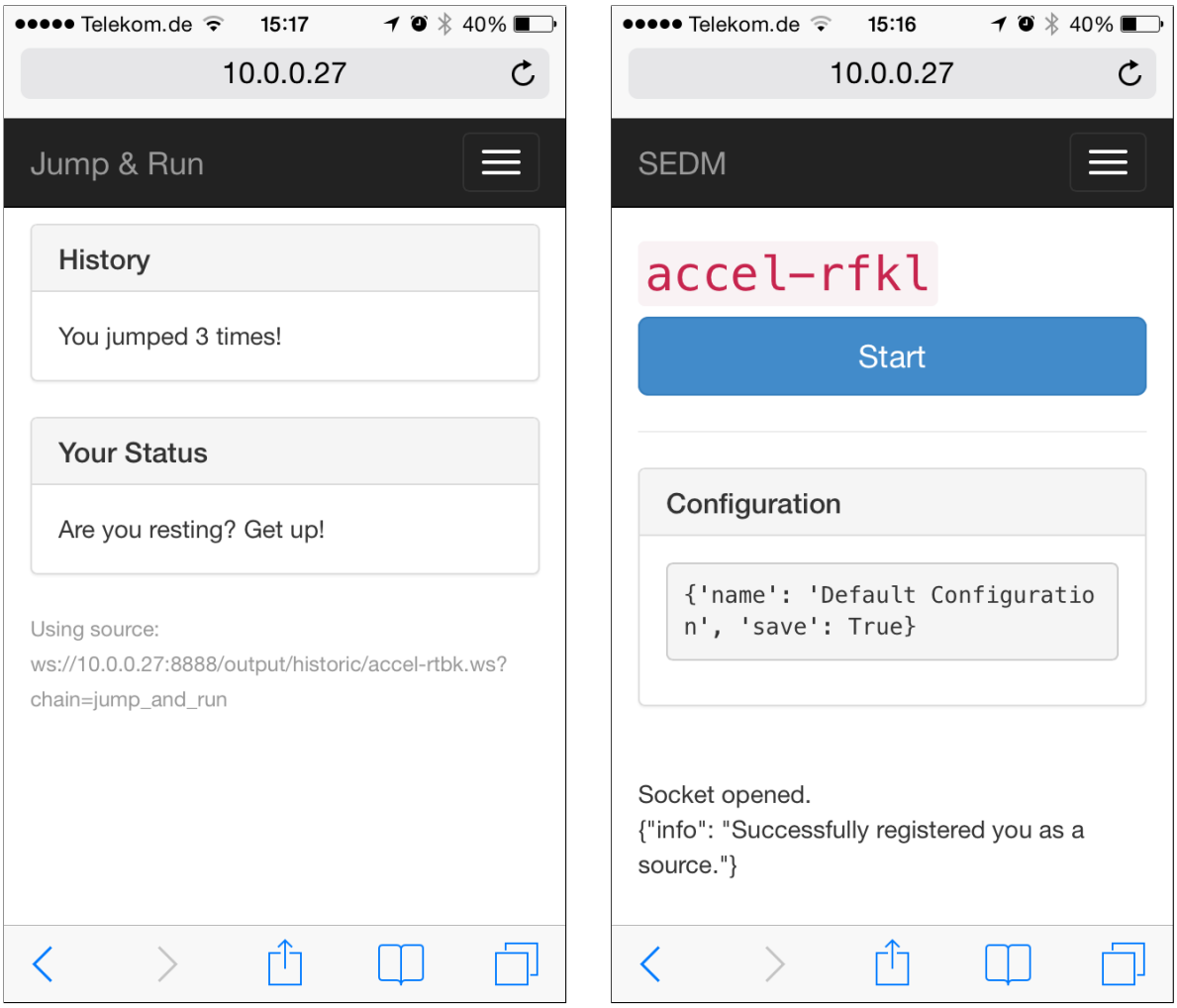
Im Zuge der Arbeit sind folgende Artefakte entstanden, um mit dem BioHarness 3 besser arbeiten zu können:

Programm zum Auslesen der Aufzeichnungen

Der BH3 ermöglicht es Daten nicht nur drahtlos zur Verfügung zu stellen, sondern diese auch im internen Speicher aufzuzeichnen. Leider stellt der Hersteller keine Software zur Verfügung, um diese Daten auf Systemen ungleich Windows auszulesen.

Basierend auf der im SDK enthaltenen Beispielanwendung die in C# geschrieben ist wurde eine Anwendung entwickelt, die es ermöglicht auch mit anderen Betriebssystemen die gespeicherten Daten auszulesen. Unterstützt werden alle Betriebssysteme, die Mono[12] als Laufzeitumgebung unterstützten.

[12] `http://mono-project.com`



(a) Darstellung der Beispielanwendung für Hüpfen und Rennen auf einem mobilen Endgerät.

(b) Anwendung zum Aufzeichnen der Beschleunigungsdaten nach erfolgreichem Registrieren am Server

Abbildung 5.7.: Screenshots von Anwendungen auf einem Smartphone.

Auf dem Gerät sind die Daten im RIFF-Format abgespeichert. Ein Parser für dieses Format und ein Konvertierer, der die Einträge ins CSV-Format konvertiert war im SDK enthalten. Es wurde allerdings die graphische Benutzungsschnittstelle und das initiieren des Kommunikationskanals neu in C# und ohne Abhängigkeiten zu windowsspezifischen Bibliotheken implementiert.

Konfiguration des BioHarness 3

Um Betriebsparameter des BH3 zum konfigurieren stellt Zepyhr eine nur mit Windows kompatible Software zur Verfügung. Um das Gerät auch von anderen Plattformen steuern zu können wurde ein

Listing 5.5 Beispiel zur Benutzung des entwickelten Frameworks zur Kommunikation mit dem BioHarness.

```
class TimeMessage(BaseMessage):
   name = 'Time'
   class Request(BaseMessage.Request):
      ID = 0x08
      DLC = 0
   class Response(BaseMessage.Response):
      padding = 3
      payload_format = '<bbHbbb'
      payload_fields = 'day month year hours minutes seconds'

class MACMessage(BaseMessage):
   name = 'MAC'
   class Request(BaseMessage.Request):
      ID = 0x12
      DLC = 0
   class Response(BaseMessage.Response):
      padding = 3
      payload_format = '17s'
      payload_fields = 'MAC'

class SetBTLinkConfgMessage(BaseMessage):
   name ='SetBTLinkConfig'
   class Request(BaseMessage.Request):
      ID = 0xA4
      DLC = 4
      payload_format = 'HH'
      LINK_TIMEOUT = 10000
      LIFESIGN_PERIOD = 3000
      @classmethod
      def get_payload(cls):
         return cls.LINK_TIMEOUT, cls.LIFESIGN_PERIOD

port = Port('/dev/tty.usbmodemfd121')
port.open()

print(send_messgage(port, TimeMessage))
print(send_messgage(port, MACMessage))
send_messgage(port, SetBTLinkConfgMessage, {'LINK_TIMEOUT': 0, 'LIFESIGN_PERIOD': 5000})
```

kleines Framework geschrieben mit dem einzelne Kommandos über die serielle Schnittstelle (USB oder Bluetooth) an das Gerät gesendet werden können.

Das Framework ist in Python 3 implementiert und ermöglicht es die in der Spezifikation [Zep11b] genannten Kommandos einfach im Quellcode zu spezifizieren und zu benutzen. Dazu müssen Nachrichten von der Basisklasse `BaseMessage` erben und die in der Spezifikation definierten Nachrichten-Ids und Längen der übertragenen Nutzdaten angeben.

Im Beispiel in Listing 5.5 sind drei Nachrichten spezifiziert. Die oberen zwei lesen die Uhrzeit und MAC-Adresse des Geräts aus. Die dritte Nachricht konfiguriert die maximale Zeitüberschreitung und Periode für Lebenszeichen der Bluetoothverbindung.

5.10. Herausforderungen und Einschränkungen

Im laufe der Implementierung sind folgende Herausforderungen aufgefallen: Das Implementieren eines binären seriellen Protokolls nach einer Spezifikation ist aufwendig. Tests müssen immer am echten Gerät durchgeführt werden. Da dieses sich wie eine Blackbox verhält, ist der genaue Fehlergrund oft schwer einzugrenzen. Anstatt zu versuchen, selbst die Kommunikation zu realisieren, ist es wahrscheinlich häufig effizienter, schon existierende Projekte gegebenenfalls für die eigenen Anwendungsfälle zu modifizieren.

Auch das Umdenken zwischen traditionellen mehrprozessgestützten und einzelprozessgestützten Anwendungen mit einer Ereignisschleife kann ein Hindernis darstellen. Das Konzept von *Futures* und die Implementierung im spezifischen Framework und der Sprache muss zunächst verstanden werden.

Nicht zuletzt ist es ohne existierende Geräte schwer, auf die spezifischen Anforderungen dieser zu kommen. Die hier implementierten Funktionen sind alle nahe an den benutzen Eingabegeräten orientiert. Dies kann natürlich zur Folge haben, dass sie einige wichtige Anforderungen von zukünftigen Quellen nicht befriedigen können. Durch eine sehr flexible und wenig strikte Architektur wurde dies allerdings zu verhindern versucht.

Die Implementierung beinhaltet nicht alle in der Architektur beschriebenen Bestandteile, sondern vor allem die für die Benutzerstudie relevanten Teile. Authentifizierung und Autorisierung sind beispielsweise nicht vorhanden. Dennoch ist es in Zukunft möglich diese Funktionen hinzuzufügen. Auch

5.11. Zusammenfassung

Bei der Implementierung wurden Technologien benutzt die plattformunabhängig sind und damit auf verschiedener Hardware benutzt werden können. Als Grundlage wurde die asynchrone Bibliothek Tornado eingesetzt und für die Persistenz von Daten MongoDB. Die Middleware wurde in der Programmiersprache Python in Version 3+ implementiert.

Die Kommunikation mit der Middleware erfolgt über HTTP und das WebSocket-Protokoll. Für Sensoren die mit Bluetooth ausgerüstet sind, ist ein Wrapper zu erstellen. Für den BioHarness 3 wurde dies exemplarisch gemacht.

Um Anwendungen zu erstellen kann prinzipiell jede Technologie zum Einsatz kommen, die über WebSockets und HTTP kommunizieren kann. Insbesondere eignen sich allerdings Webbrowser gut als Laufzeitumgebung, da diese mit HTML, CSS und JavaScript alle dafür notwendigen Anforderungen erfüllen. Webanwendungen können vor allem als Prototypen sowohl für Ausgabe als auch für die Dateneingabe benutzt werden. Als Nachrichtenformat wurde JSON gewählt, da dieses weit verbreitet ist und schnell kodiert und dekodiert werden kann. Die Struktur der Nachrichten ist sehr flexibel und gut auf zukünftige Anwendungsfälle anzupassen.

Neben der reinen Infrastruktur für Sensoren und Anwendungen enthält die entwickelte Middleware auch die Funktionalitäten der Entwicklerunterstützung. Diese sind über einen Webbrowser erreichbar und erlauben das Aufzeichnen, Verwalten und Analysieren von Daten.

Zur Verarbeitung von Daten wird eine Kette von Transformationen benutzt, die aus den Eingaben eine Ausgabe erzeugen. Es gibt einige vorgefertigte generische Filter im System um Arbeit zu ersparen. Filter können selbst im Hintergrund beliebigen Code ausführen und beschränken die Mächtigkeit der Datenanalyse nicht.

6. Nutzerstudie

> Wenn du dir die Anwender deiner Programme als Idioten vorstellst, werden auch nur Idioten deine Programme verwenden.
>
> *(Linus Torvalds)*

Um die Funktionalität des entwickelten Systems zu testen und daraus Anforderungen an zukünftige Weiterentwicklungen abzuleiten, wurde eine Nutzerstudie durchgeführt. Im folgenden wird der Ablauf beschrieben und die daraus gewonnen Ergebnisse ausgewertet.

6.1. Aufbau und Ablauf der Studie

Die Benutzerstudie wurde mit einer Gruppe von Entwicklern durchgeführt. Alle Entwickler befanden sich im selben Raum, der auch mit einem Beamer ausgestattet war. Jeder Teilnehmer hatte seinen eigenen Computer zur Verfügung. Zunächst stellten sich die Teilnehmer vor und füllten den ersten Teil des Fragebogens aus. Dieser fragte das Alter, die berufliche Tätigkeit und verschiedene Kenntnisse über Technologien ab.

Danach wurde das Thema der Diplomarbeit und eine Einführung in das System gegeben. Es wurden die einzelnen Schritte der Eingabe, Verarbeitung und Ausgabe der Daten kurz im Überblick beschrieben und an welchen Stellen im Quellcode die jeweiligen Funktionalitäten zu finden sind. Die Folien der Präsentation sind in Anhang A aufgeführt.

Nach dem Überblick wurde die Aufgabenstellung gegeben und zu dieser nochmals eine kurze Liste von logisch aufeinander folgenden Schritten mit kleinen Ausschnitten von Beispielcode. Die Entwickler bearbeiteten daraufhin die gestellte Aufgabe an ihren Laptops. Dazu wurde ihnen der Quellcode des Programms gegeben. Die notwendigen Abhängigkeiten wurden vor Beginn der Studie auf den einzelnen Laptops der Teilnehmer installiert und konfiguriert.

Nach Ende der praktischen Phase wurde jeweils der zweite Teil des Fragebogens ausgeteilt. Dieser Teil erfasste mithilfe einer Likert-Skala die Unterstützung der Entwickler durch das System und eine grobe Einschätzung, inwiefern es zukünftig erweiterbar ist und ob die eingesetzten Technologien als hilfreich für Plattformunabhängigkeit und der Erstellung von Prototypen angesehen werden.

Nach der Erfassung dieser strukturierten Daten mussten sich die Teilnehmer in Zweiergruppen zusammenfinden und Probleme notieren, die ihnen während der Benutzung aufgefallen sind. Jedes Problem wurde dabei auf eine Karteikarte geschrieben. Danach wurden Zweiergruppen gebildet, in denen die Karten mit möglichen Lösungen und Verbesserungen die zu dem genannten Problem passen,

(a) Teilnehmer beim Implementieren der Lösung.

(b) Teilnehmer beim Aufzeichnen von Testdaten.

Abbildung 6.1.: Bilder während der Durchführung der Nutzerstudie.

annotiert wurden. Nach Abschluss dieser Phase präsentierte jede Zweiergruppe ihre Vorschläge und es konnte über diese in der gesamten Gruppe diskutiert.

6.2. Auswertung

Es wurden zum einen persönliche Daten und Kenntnisse der Probanden erfasst. Nach der Benutzung des Systems wurden mittels einer Likert-Skala verschiedene Aussagen über das System abgefragt. Im letzten Teil wurden offene Probleme identifiziert und in Gruppenarbeit mögliche Verbesserungen diskutiert.

6.2.1. Persönliche Daten und Kenntnisse

Es haben sechs Teilnehmer an der Nutzerstudie teilgenommen. Die Teilnehmer waren durchschnittlich 25 Jahre alt (mit einer Standardabweichung von 0.63). Alle Teilnehmer waren männlich. Die Teilnehmer sind alle in ihrem Haupt- oder Nebenberuf mit Softwareentwicklung beschäftigt.

Die Persönlichen Fähigkeiten und Kenntnisse wurden in einer Skala mit den Werten des Kenntnisständen *keine*, *etwas*, *gute*, *sehr gute* abgefragt. In Tabelle 6.1 sind die Teilnehmer mit den jeweiligen Daten vermerkt. Dabei stehen die Spalten K1 bis K8 für folgenden Kenntnisse:

K1 Webentwicklung allgemein

K2 HTML

K3 CSS

K4 JavaScript

K5 WebSocket (RFC 6455)

Persönliche Daten		**Kenntnisse**							
Alter	Berufliche Tätigkeit	K1	K2	K3	K4	K5	K6	K7	K8
25	IT-Berater	1	1	1	1	0	1	0	0
24	Softwareentwickler	2	3	2	2	1	2	0	1
25	Softwareentwickler	2	3	2	2	0	1	0	0
25	Softwareentwickler	2	2	2	1	1	2	0	1
25	Student (Softwaretechnik)	3	3	2	2	2	2	0	1
26	Student (Softwaretechnik)	1	1	1	1	0	0	0	0

Tabelle 6.1.: Persönliche Daten und Kenntnisse der Teilnehmer. 0 entspricht *keinen* Kenntnissen, 3 entspricht *sehr guten* Kenntnissen.

K6 Python (Programmiersprache)

K7 Tornado (Netzwerkframework)

K8 Algorithmen zur Aktivitätserkennung

Traditionelle Webtechnologien waren bei allen Teilnehmern mindestens *etwas* vorhanden. Im Durchschnitt sind Kenntnisse in HTML *gut*. Kenntnisse in CSS und JavaScript sind zwischen *etwas* und *gut* einzuordnen. Kenntnisse mit der Benutzung von Websockets hingegen wurden nur von drei Personen angegeben.

Programmiererfahrung in Python war bei fünf der sechs Teilnehmern grundlegend vorhanden, allerdings wies keiner der Teilnehmer Kenntnisse im benutzten Framework für den Server auf. Die Hälfte der Teilnehmer gab an, *etwas* Kenntnisse im Themengebiet *Algorithmen zur Aktivitätserkennung* zu besitzen.

6.2.2. Unterstützung durch das Systems

Die Funktionalität und Benutzbarkeit des Systems wurde anhand des praktischen Einsatzes getestet. Die Teilnehmer hatten als Aufgabe eine Anwendung zu entwickeln, die visualisiert ob der Benutzer still steht, rennt oder hüpft. Diese Aufgabe wurde auch als Beispiel in 5.7 umgesetzt. Die Teilnehmer hatten aber nur Zugriff auf die einzelnen Filter und mussten die Filterkette selbst entwerfen und implementieren. Auch die Beispieldaten mussten selbst aufgezeichnet werden. Als Hilfestellung hatten alle Teilnehmer die zur Einführung genutzte Präsentation zur Hand. Zum Aufzeichnen der Daten wurde der Beschleunigungssensor von Smartphones über deren Webbrowser eingesetzt. Jeder Teilnehmer hatte Zugriff auf ein Smartphone.

Für die Lösung der Aufgabe wurden 45 Minuten Zeit gegeben. In dieser Zeit konnten drei Teilnehmer die Aufgabe abschließen und eine visuelle Ausgabe zu eingehenden Daten erzeugen. Der Rest der Teilnehmer war noch mit dem Optimieren des Erkennungsalgorithmus beschäftigt. Die Bearbeitung der Aufgabe war auf Einzelpersonen ausgelegt und eine Kooperation war zunächst nicht vorgesehen.

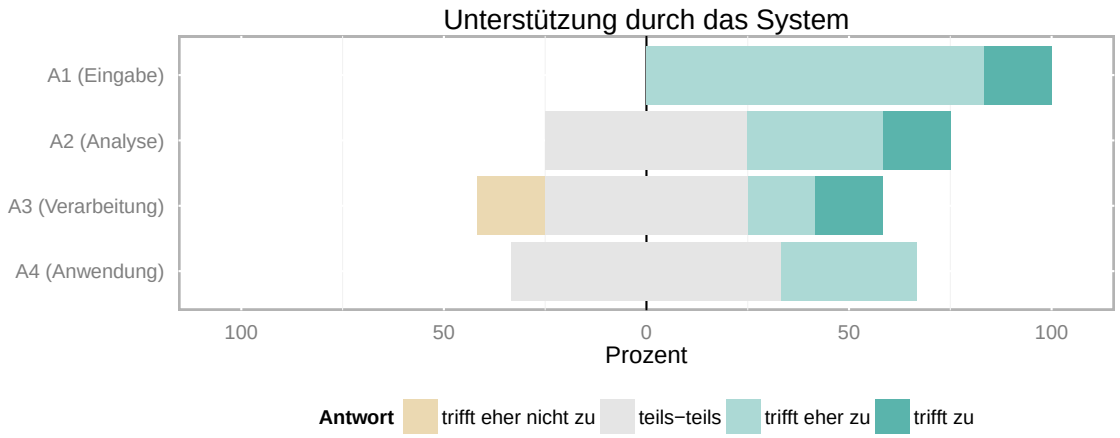


Abbildung 6.2.: Ergebnisse des Fragebogens für die Unterstützung durch das System

Gegen Ende der Aufgabe zeigten sich allerdings Versuche der Zusammenarbeit, die vom Versuchsleiter nicht unterbunden wurden. Eine Versuchsperson, die schon eine fortgeschrittenere Anwendung zur Visualisierung entwickelt hatte wollte diese mit besseren Eingabedaten versorgen. Da alle Teilnehmer im selben Netzwerk waren, konnte dies auch schnell durch ein einfaches ändern der IP-Adresse der Datenquelle realisiert werden. In einem sehr kleinen Maßstab zeigt diese selbstorganisierte Kollaboration den Gedanken der hinter dem Gesamtsystem steht. Eine verbesserte Zusammenarbeit zwischen den beteiligten Benutzergruppen.

Nach der abschließen der Aufgabe wurden den Teilnehmern auf einem Fragebogen einige Aussagen zur Unterstützung durch das System gegeben. Dabei wurde eine Likert-Skala mit fünf möglichen Antworten eingesetzt. Die zu bewertenden Aussagen lauteten dabei wie folgt (Durchschnittliche Bewertung von 1 bis 5 und die Standardabweichung in Klammern):

A1 Das System hat mich beim Aufzeichnen von Testdaten gut unterstützt. (4,2 / 0,4)

A2 Das System hat mich beim Analysieren der Testdaten gut unterstützt. (3,7 / 0,7)

A3 Das System hat mich beim Erstellen des Erkennungsalgorithmus gut unterstützt. (3,3 / 0,9)

A4 Das System hat mich beim Erstellen einer Anwendung gut unterstützt. (3,3 / 0,5)

Abbildung 6.2 zeigt die Ergebnisse der Befragung. Vor allem das Aufzeichnen von Testdaten wurde als einfach empfunden. Dieser Vorgang wurde von einigen Personen auch mehrmals durchgeführt, als sich die aufgezeichneten Daten in der Analyse als ungeeignet erwiesen. Die Unterstützung des Systems um Erkennungsalgorithmen zu entwerfen und eine Anwendung zu schreiben liegt im erwarteten Rahmen. Die Kenntnisse der Teilnehmer im Bereich Erkennungsalgorithmen (K8) war generell nicht vorhanden oder nur gering. Um Anwendungen zu entwickeln stellt das System außer die Schnittstelle keine weitergehenden Funktionalitäten wie graphische Elemente zur Anwendungsentwicklung selbst

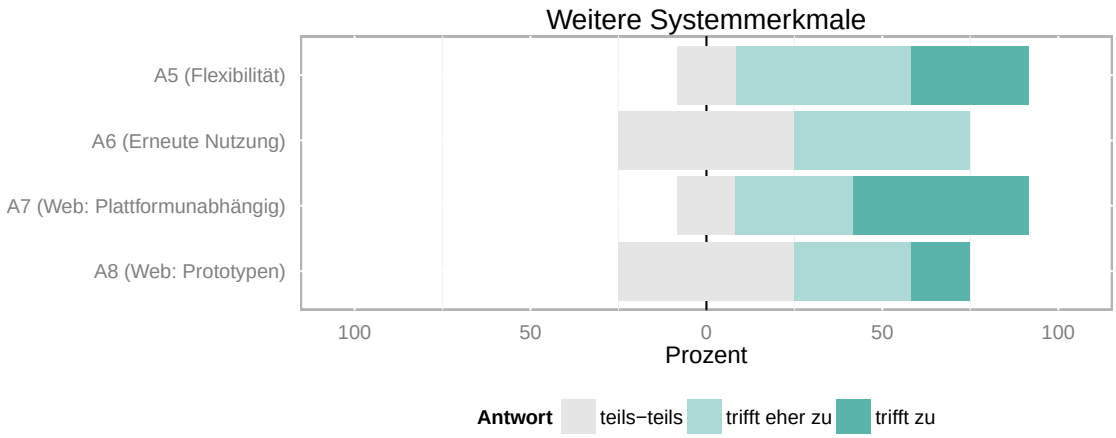


Abbildung 6.3.: Auswertung der restlichen Fragen

bereit. Es wurde lediglich ein Grundgerüst einer Webseite mit integriertem Grunddesign (Boostrap CSS Framework) und jQuery angeboten.

6.2.3. Weitere Merkmale

Neben der Unterstützung durch das System wurden vier weitere Aussagen abgefragt. Die Auswertung dafür ist in Abbildung 6.3 dargestellt. Die Aussagen A5 bis A8 mit Durchschnitt und Standardabweichung waren im Wortlaut:

A5 Ich denke, dass ich mit der Architektur vielfältige Anwendungsfälle umsetzen kann. (4,2 / 0,7)

A6 Ich würde das System auch für zukünftige Entwicklungen in diesem Bereich einsetzen. (3,5 / 0,5)

A7 Webtechnologien sind für plattformübergreifende Entwicklungen gut geeignet. (4,3 / 0,5)

A8 Webtechnologien sind gut für das Erstellen von Prototypen geeignet. (3,7 / 0,7)

Die Einschätzungen über den Einsatz von Webtechnologien für plattformübergreifende Projekte oder Prototypen decken sich mit den in *Mobile application development: web vs. native* [CL11] beschrieben. Die hohe Wertung in der Flexibilität der Architektur ist ein gutes Zeichen, auch wenn das nach einem solch kurzen Test nicht zu stark gewichtet werden darf.

6.2.4. Herausforderungen und Verbesserungsmöglichkeiten

Nach der Befragung mit dem Fragebogen wurden den Testpersonen Karteikarten gegeben auf denen sie jeweils ein Problem notieren sollten, das ihnen während der Benutzung des Systems aufgefallen ist. Nach dem Erfassen der einzelnen Probleme wurden dann Zweiergruppen gebildet und mögliche

Lösungsvorschläge auf den Karten notiert. Am Ende präsentierte jede Zweiergruppe ihre Karten und stellte kurz die Vorschläge vor.

Insgesamt wurden 19 Karten angelegt. Eine Kategorisierung der Karten in jeweils die Bereiche des Systems und Problemtypen ergibt sich die in Tabelle 6.2 dargestellte Verteilung. Eine Visualisierung dieser Daten ist in Abbildung 6.4 sichtbar.

Der Problemtyp *Bug* steht für Programmfehler die während des Tests entdeckt wurden. Keiner der gefundenen Bugs hat verhindert, dass die Aufgabe lösbar war. Der Typ *Dokumentation* steht für Probleme, die durch verbesserte Dokumentation ausgeglichen werden könnten. Es wurden unter anderem mehrfach bessere Beschreibungen von Argumenten von Filtern gefordert. Auch dass die Kenntnisse in eingesetzten Technologien nicht ausreichend vorhanden war, ist in diesem Typ einsortiert, da dies durch bessere Dokumentation verhindert oder mindestens verringert werden könnte.

Probleme des Typs *Funktionalität* bemängeln fehlende Funktionen, die nicht hauptsächlich mit einer Umgestaltung der Benutzungsschnittstelle zu tun haben. Ein Beispiel dafür ist, in der Detailansicht von Aufzeichnungen, zusätzliche statistische Werte wie Durchschnitt, Minimum und Maximum einzelner Felder anzuzeigen.

Unter dem Problemtyp *UI* wurden Karten sortiert, die sich mit der Benutzungsschnittstelle befassten. Diese befassten sich häufig teilweise auf das gesamte System. Aber vor allem auch im Bereich der Verarbeitung wurde häufiger eine bessere graphische Unterstützung gewünscht. Beispielsweise wurde sich mehrmals eine grafische Oberfläche gewünscht, um die einzelnen Elemente einer Filterkette visualisiert zu sehen. Auch die Möglichkeit bestimmte Zeiträume in der Analyseansicht zu vergrößern und die Skalierung dynamisch anzupassen fallen in diesen Problemtyp.

Bei der Präsentation der Karten vor der Gruppe wurden vor allem Elemente der Benutzungsschnittstelle diskutiert. Dort liegt laut Aussagen das größte Verbesserungspotential. Eine Lösung, welche die Mächtigkeit des Schreiben der Filter in Python mit einer einfachen Übersicht und Konfiguration verbinden würde wäre, die Filter ähnlich wie in Abbildung 5.2 zu visualisieren und dabei die Argumente interaktiv modifizierbar zu machen. Mit dem grundlegenden Konzept der Filter fanden sich alle Teilnehmer auch ohne Vorkenntnisse gut zurecht.

Folgende Zitate der Teilnehmer, die in der Diskussion und während der Studie fielen, repräsentieren den aktuellen Stand des Systems gut:

- „Mit Hilfe des Systems konnte ich innerhalb kürzester Zeit eine funktionierende Applikation erstellen."
- „Die Komplexität der Datenverarbeitung der Bewegungsdaten konnte durch den Einsatz der Filterketten stark reduziert werden."
- „Der Prototyp bietet nicht nur einen Mehrwert für Entwickler, sondern hat auch das Potential Nicht-Entwicklern einen Zugang zu der Thematik zu geben."
- „Für einen produktiven Einsatz ist vor allem eine Verbesserung der GUI notwendig."
- „Die GUI war verbesserungswürdig, der Rest ganz gut."
- „Nach Startschwierigkeiten war das System gut zu bedienen."

Bereich	**Problemtyp**				Summe
	Bug	Dokumentation	Funktionalität	UI	
Allgemein	0	2	0	2	4
Analyse	0	1	3	0	4
Anwendung	3	0	0	0	3
Eingabe	1	0	0	1	2
Verarbeitung	0	3	1	2	6
Summe	4	6	4	5	19

Tabelle 6.2.: Auswertung der Karteikarten der Nutzerstudie

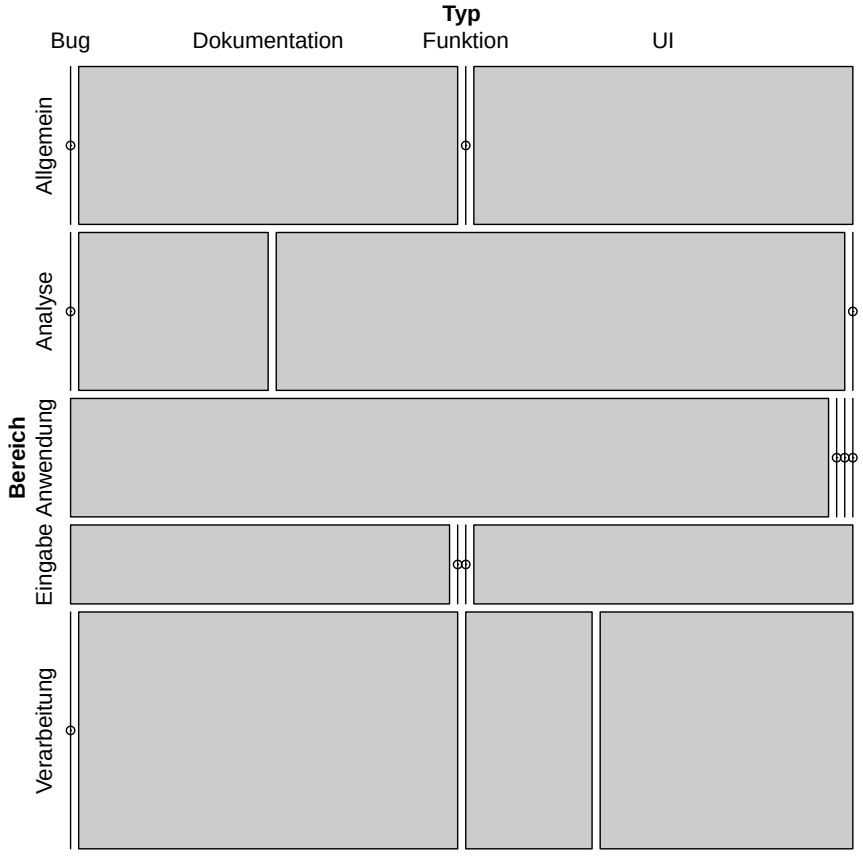


Abbildung 6.4.: Darstellung der Verteilung der Problemtypen und Problembereiche als Mosaikplot.

6.2.5. Zusammenfassung

Die Benutzerstudie hat gezeigt, dass das System grundlegend funktioniert und es auch Nutzern die zuvor nicht damit gearbeitet haben, ein komplettes kleines Projekt umsetzen können. Ohne dass explizit eine Kooperation während dem Entwicklungsprozess vorgesehen war, ist dieser durch einige Teilnehmer selbstorganisiert zustande gekommen. Die Unterstützung durch das System wurde in den Bereichen des Datenerfassens und der Analyse als gut angesehen. Die Datenverarbeitung und Anwendungsentwicklung wurde als weniger gut unterstützt angesehen.

Mögliche Verbesserungspotential ist vor allem im Bereich der Benutzungsschnittstelle und Funktionalität zu finden. Für das Erstellen von Erkennungsalgorithmen wäre eine ausführliche graphische Oberfläche hilfreich, in der die Filter angeordnet und konfiguriert werden können. Da der Fokus der Implementierung nicht auf dem direkten Unterstützung beim Entwerfen von Algorithmen lag, sondern auf der Funktionalität des Gesamtsystems und dem Zusammenspiel der einzelnen Bereiche, sind die Ergebnisse positiv.

7. Zusammenfassung und Ausblick

Der Bereich der *Wearables* ist ein Zukunftsmarkt, der stark am wachsen ist. In naher Zukunft werden wohl auch mehr Sensoren den Sprung von Geräten, die wir an uns tragen in unsere Kleidung machen. In dieser Arbeit wurde der aktuelle Stand der Technik bei der Entwicklung von solch intelligenten Kleidungsstücken beschrieben. Auf dem Markt befindliche Geräte sind noch nicht mit unserer Kleidung verbunden, sondern einzelne Geräte die eine Vielzahl von Sensoren enthalten. Hauptsächlich findet die Kommunikation über Bluetooth mit genau einem anderen Gerät statt, oder die Daten werden auf dem Gerät selbst für längere Zeit aufgezeichnet und später ausgewertet.

Es wurden Anforderungen an eine einheitlichere Architektur erfasst, die neben der reinen Funktionalität der Datenverarbeitung zwischen Sensor und Anwendung, auch darauf ausgelegt ist, den Entwicklungsprozess zu verbessern und vereinheitlichen. Aus diesen Anforderungen wurde eine Architektur entwickelt und beschrieben.

Diese Architektur wurde teilweise implementiert, um zum einen die Machbarkeit mit bestimmten Technologien zu demonstrieren und auch eine qualitative Evaluation der Funktionalität und Benutzbarkeit durchzuführen. Als Datenquellen wurde dabei der BioHarness 3 und Beschleunigungssensoren des Smartphones benutzt. Die Implementierung setzt für die Schnittstellen und Entwicklerunterstützung auf Python und Webtechnologien.

Bei der Evaluation der Implementierung im Rahmen einer Nutzerstudie zeigte sich, dass die Grundfunktionalität komplett verfügbar und benutzbar ist. Eine Beispielanwendung konnte komplett von Erstbenutzern des Systems implementiert werden. Dennoch wäre eine verbesserte graphische Unterstützung in den Bereichen der Datenanalyse und Entwicklung des Erkennungsalgorithmus sehr sinnvoll, auch wenn das System auf Entwickler ausgelegt ist.

Ausblick

Die Implementierung der Architektur ist eher konzeptuell und auf die Demonstration der Praktikabilität der eingesetzten Technologien ausgelegt. Es kann noch viel Energie in die Benutzbarkeit der Werkzeuge investiert werden. Hier sind einige Punkte aufgelistet, bei denen Raum für Verbesserungen besteht:

Echtzeitdaten in der Analyseansicht Neben Aufzeichnungen könnten auch Echtzeitdaten als Eingabe akzeptiert werden und mit diesen die Graphen in parallel zu ihrer Aufzeichnung gezeichnet werden.

Visualisierung mehrdimensionale Werte Auch werden alle betrachteten Werte im Moment als unabhängige eindimensionale Werte visualisiert. Für komplexere Eingaben können noch weitere Visualisierungen entwickelt und implementiert werden.

Visualisierung der Filter & Filterkette Die Filterkette könnte grafisch visualisiert werden. Gegebenenfalls könnte auch Filterargumente und Kombination der Filter über die Oberfläche konfiguriert werden.

Verbesserte Filter Die im Moment enthaltenen Filter sind ausreichend um einfach Anwendungsfälle abzudecken. Es könnten weitere Filter entwickelt werden und diese auch flexibler konfigurierbar gemacht werden.

Integration von maschinellem Lernen Mit den aufgezeichneten Beschriftungen könnten Aktivitäten durch maschinelles Lernen erkannt werden.

Vorgefertigte Visualisierungskomponenten Um Anwendungsentwicklung zu vereinfachen könnten einfache Diagramme als Komponente angeboten werden, welche den aktuellen Wert von bestimmten Parametern darstellen.

Diese Arbeit ging davon aus, ein immer verfügbares Netzwerk zu Kommunikation zwischen den Geräten zu haben, in dem sich intelligente Kleidung ähnlich eines Sensornetzes um den Benutzer aufspannt. Diese Datenwolke um den Benutzer soll Anwendungen in seiner Umgebung mehr Möglichkeiten bieten, auf den aktuellen Kontext des Benutzers reagieren. Auch wenn ein immer verfügbares Netzwerk heutzutage noch nicht die Realität ist, ist es langfristig ein realistisches Szenario.

Die hier vorgestellte Middleware erlaubt neben dem aufspannen einer persönlichen Datenwolke auch das Teilen dieser Daten mit anderen Menschen. Denkbar wäre beispielsweise, einige erfasste Daten zu medizinischen Zwecken seinem Arzt oder einem automatisierten Frühwarnsystem zur Verfügung zu stellen. Neben den reinen Daten könnten auch die Algorithmen zur Datenverarbeitung geteilt werden.

Wenn mehr Daten anfallen, wächst allerdings auch die Nachfrage nach diesen Daten. Um sich zu einem gläsernen Bürger zu machen und seine Privatsphäre zu schützen, ist darauf zu achten, seine Daten nicht leichtfertig Dritten zu übergeben. Um die Datenhoheit zu bewahren, ist auch vorstellbar die Middleware bei sich zu Hause zu betreiben. Beispielsweise könnte sie auf dem Rechner, der sowieso schon für die Verwaltung des elektronischen Gedächtnisses zuständig ist, ausgeführt werden.

All die Aspekte der Sicherheit und Privatsphäre gehen über diese Arbeit, die hauptsächlich ein Werkzeug zur Entwicklungsunterstützung beschreibt, weit hinaus. Dennoch sind es gerade diese Aspekte, die den späteren Endbenutzer eines Systems stark betreffen. Durch eine bessere Unterstützung der Entwickler bleibt diesen hoffentlich genug Zeit, sich mit den wichtigen Aspekten der Software zu beschäftigen und intelligente Kleidung auf eine solide Plattform zu stellen.

A. Präsentation der Nutzerstudie

Die Präsentation wurde in HTML und JavaScript mit Hilfe der Bibliothek `reveal.js`[1] umgesetzt. Die Teilnehmer hatten in ihrem Webbrowser, während des Lösens der Aufgabe, Zugriff auf diese Präsentation.

Im folgenden sind die einzelnen Folien aufgeführt, die Leserichtung der Folien ist (nach Drehen der Seite) von oben nach unten und links nach rechts.

[1] `http://lab.hakim.se/reveal-js/`

Benutzerstudie

Entwicklung einer Architektur
für das Betriebssystem intelligenter
Kleidung

Diplomarbeit – Tobias Birmili

Überblick

Entwicklungsunterstützung für *Wearables*

Vorstellungsrunde

Ich bin *Tobias* und schreibe gerade meine Diplomarbeit im Studiengang *Softwaretechnik*.

Zielgruppen

- Sensorentwickler
- Datenverarbeitungsexperten
- Anwendungsentwickler

Funktionen

- Aufzeichnung von Daten
- Verarbeitung von Daten
- Schnittstelle zu Anwendungen

BioHarness

- Puls
- Atemfrequenz
- Beschleunigungssensoren

Eingabe

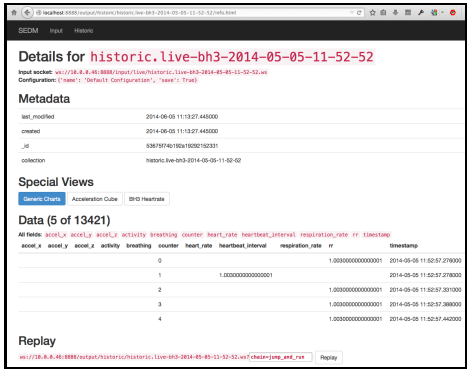
- Zephyr BioHarness 3
- Bewegungssensoren
- Beschriftungen
- ...

Bewegungssensor

Beschriftungen erstellen

Datensätze

Aufzeichnungen (`/historic/`) und Detailansichten



Beliebige Quellen

WebSocket API

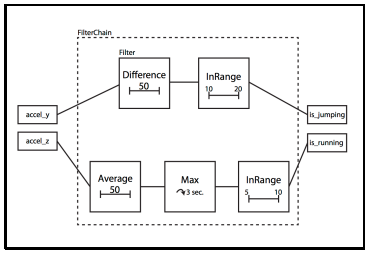
```
var ws = new WebSocket("ws://server.dev:8888/input/live/my-sens.ws");
// wait for successful connection
ws.send(JSON.stringify({
        'my_field': 'some_value',
        'other_field': 42
})
```

Analyse

Visualisierung von Eingabe und Filtern.

Verarbeitung

Filter + Filterketten
Vorgefertigte einfache Filter



Implementierung

Beispiel einer Filterkette

```
class SomeChain(BaseFilterChain):

    name = 'some_chain'

    def init_filters(self):

        y_diff_50 = DifferenceInWindowFilter('accel_y', window_size=50)
        self.add_filter(y_diff_50, show_in_graph=True)

        is_cool = InRangeFilter(y_diff_50.out_name, 0, 35, out_name='is_cool'
        self.add_filter(is_jumping, show_in_graph=True)
```

Implementierung

Beispiel: `InRangeFilter`

```
def process(self, document, *args, **kwargs):
    try:
        val = document[self.field]
    except KeyError:
        return document

    in_range = self.from_val <= val <= self.to_val
    document.update({
        self.out_name: int(in_range)*100  # int for plotting purposes
    })
    return document
```

Ausgabe

WebSockets

```
var socket = new WebSocket('ws://server.dev:8888/output/historic/accel-demo.w

socket.onmessage = function(message){
    data = JSON.parse(message.data);
    if(data.error){
        alert(data.error);
    } else if (data.info) {
        console.log(data.info);
    } else if(data.is_cool) {
                            // He's cool!
    } else {
                            // He is not cool!
    }
}
```

Die Aufgabe

Schreibe eine kleine Anwendung die anzeigt ob der Benutzer hüpft oder rennt.

Smartphone als Sensor ist ausreichend.

Analyseansicht benutzen

Den Analyse-View bearbeiten:
`sedm.server.handlers.analysis.AnalysisHandler`

- `MAIN_DATASET = 'accel-bla'`
- `FILTER_CHAIN = get_chain('my_chain')`

Daten Aufzeichnen

Mit dem Smartphone ein bisschen hüpfen und rennen.

Optional: Jemanden bitten währenddessen Labels zu erstellen.

Filterkette bearbeiten

Siehe `sedm.processing.chains.MyChain`.
Beispiel:

```
def init_filters(self):
    # calculates difference of 50 values using field 'accel_y'
    y_diff_50 = DifferenceInWindowFilter('accel_y', window_size=50)
    self.add_filter(y_diff_50, show_in_graph=True)
```

Anwendung bauen

Siehe `/templates/apps/my_app.html`.
Erreichbar unter `localhost:8888/app/my`
jQuery ist verfügbar. Beispiel für Datennutzung siehe hier.

Lösung der Aufgabe

```
make run_dev
```

oder:

```
python server.py --logging=debug --debug
```

Tipps

1. Hüpfen erzeugt eine starke Differenz auf einer Achse
2. Rennen beansprucht eine andere Achse
3. Analyseview einrichten und damit die Filterkette *debuggen*.

Fragebogen Teil 2

Bitte ausfüllen.

Schwierigkeiten auf Karten notieren

Was war schwierig? Bitte eine *Schwierigkeit* pro Karte.

Drei Zweiergruppen bilden

Die eigenen Karten durchgehen und darauf notieren:
Was könnte man tun um die Schwierigkeiten zu beheben/verbessern/…

Vorstellen der Karten und Diskussion

Ende

Vielen Dank für die Teilnhame

B. Fragebogen zur Auswertung der Nutzerstudie

Nutzerstudie

Fragebogen

Weiter mit ⇓

1. Wie alt sind Sie?

   _____

2. Welche berufliche Tätigkeit üben Sie in ihrem Hauptberuf aus?

   _____

3. Bewerten Sie ihre persönlichen Fähigkeiten/Kenntntisse in den folgenden Bereichen der Softwareentwicklung.

	keine Kenntnisse	**etwas**	**gute**	**sehr gute**
Webentwicklung allgemein	☐	☐	☐	☐
HTML	☐	☐	☐	☐
CSS	☐	☐	☐	☐
JavaScript	☐	☐	☐	☐
WebSocket (RFC 6455)	☐	☐	☐	☐
Python (Programmiersprache)	☐	☐	☐	☐
Tornado (Netzwerkframework)	☐	☐	☐	☐
Algorithmen zu Aktivitätserkennung	☐	☐	☐	☐

4. Benutzen Sie selbst Geräte aus dem Bereich *Wearable Computing*? Wenn ja, welche?

   _____

Gruppe 1

Weiter mit ⇓

5. Sie haben nun die Software für eine kleine Aufgabe eingesetzt. Bewerten Sie die folgenden Aussagen in Bezug auf das benutzte System:

	trifft nicht zu	**trifft eher nicht zu**	**teils-teils**	**trifft eher zu**	**trifft zu**
Das System hat mich beim Aufzeichnen von Testdaten gut unterstützt.	☐	☐	☐	☐	☐
Das System hat mich beim Analysieren der Testdaten gut unterstützt.	☐	☐	☐	☐	☐
Das System hat mich beim Erstellen des Erkennungsalgorithmus gut unterstützt.	☐	☐	☐	☐	☐
Das System hat mich beim Erstellen einer Anwendung gut unterstützt.	☐	☐	☐	☐	☐
Ich denke, dass ich mit der Architektur vielfältige Anwendungsfälle umsetzen kann.	☐	☐	☐	☐	☐
Ich würde das System auch für zukünftige Entwicklungen in diesem Bereich einsetzen.	☐	☐	☐	☐	☐
Webtechnologien sind für plattformübergreifende Entwicklungen gut geeignet.	☐	☐	☐	☐	☐
Webtechnologien sind gut für das erstellen von Prototypen geeignet.	☐	☐	☐	☐	☐

Literaturverzeichnis

[ACCL14] O. Amiraslanov, J. Cheng, P. Chabrecek, P. Lukowicz. Electroluminescent based Flexible Screen for Interaction with Smart Objects and Environment. Technischer Bericht, German Research Center for artifical Intelligence, 2014. (Zitiert auf Seite 27)

[ALO+04] O. Amft, M. Lauffer, S. Ossevoort, F. Macaluso, P. Lukowicz, G. Troster. Design of the QBIC wearable computing platform. In *Proceedings. 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, 2004.*, S. 398–410. IEEE, 2004. doi:10.1109/ASAP.2004.1342488. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1342488`. (Zitiert auf Seite 24)

[CL11] A. Charland, B. Leroux. Mobile application development. *Communications of the ACM*, 54(5):49, 2011. doi:10.1145/1941487.1941504. URL `http://portal.acm.org/citation.cfm?doid=1941487.1941504`. (Zitiert auf den Seiten 42 und 61)

[CLH+13] J. Cheng, P. Lukowicz, N. Henze, A. Schmidt, O. Amft, G. A. Salvatore, G. Troster. Smart Textiles: From Niche to Mainstream. *IEEE Pervasive Computing*, 12(3):81–84, 2013. doi:10.1109/MPRV.2013.55. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6562715`. (Zitiert auf den Seiten 7 und 17)

[DS14] C. Dalsgaard, R. Sterrett. Market Opportunities for Smart Textiles 2014 White paper on smart textile garments and devices : a market overview of smart textile wearable technologies . Technischer Bericht, Ohmatex ApS, 2014. URL `http://www.ohmatex.dk/pdfer/whitepaper_smart_textiles.pdf`. (Zitiert auf Seite 13)

[HAT10] H. Harms, O. Amft, G. Tröster. Estimating posture-recognition performance in sensing garments using geometric wrinkle modeling. *IEEE Transactions on Information Technology in Biomedicine*, 14(6):1436–1445, 2010. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5585763`. (Zitiert auf Seite 9)

[Int11] Internet Engineering Task Force. RFC 6455: The WebSocket Protocol, 2011. URL `http://tools.ietf.org/html/rfc6455`. (Zitiert auf Seite 42)

[JL07] J. Judewig, H. Lichter. *Software Engineering*. dpunkt.verlag GmbH, 2007. (Zitiert auf Seite 14)

[KBS99] G. Kortuem, M. Bauer, Z. Segall. NETMAN : The design of a collaborative wearable computer system. *Mobile Networks and Applications 4*, 4:49–58, 1999. doi:10.1023/A:1019122125996. (Zitiert auf Seite 20)

[KWL+11] T. Karrer, M. Wittenhagen, L. Lichtschlag, F. Heller, J. Borchers. Pinstripe: eyes-free continuous input on interactive clothing. In *Proceedings of the 2011 annual conference on Human factors in computing systems - CHI '11*, S. 1313. ACM Press, New York, New York, USA, 2011. doi:10.1145/1978942.1979137. URL `http://dl.acm.org/citation.cfm?doid=1978942.1979137`. (Zitiert auf Seite 16)

[Lew04] F. L. Lewis. Wireless Sensor Networks. *Smart Environments: Technologies, Protocols, and Applications*, S. 1–18, 2004. doi:10.1007/b117506. URL `http://www.springerlink.com/index/10.1007/b117506`. (Zitiert auf Seite 20)

[LHSA99] J. Lehikoinen, J. Holopainen, M. Salmimaa, A. Aldrovandi. MEX: a distributed software architecture for wearable computers. In *Digest of Papers. Third International Symposium on Wearable Computers*, S. 52–57. IEEE Comput. Soc, 1999. doi:10.1109/ISWC.1999.806650. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=806650`. (Zitiert auf Seite 20)

[Man98a] S. Mann. Definition of Wearable Computer, 1998. URL `http://wearcam.org/wearcompdef.html`. (Zitiert auf Seite 11)

[Man98b] S. Mann. Wearable Computing as means for Personal Empowerment. In *Proc. 3rd Int. Conf. on Wearable Computing (ICWC)*, S. 1–8. 1998. URL `http://wearcam.org/icwckeynote.html`. (Zitiert auf Seite 11)

[MMC09] P. Mistry, P. Maes, L. Chang. WUW - Wear Ur World: A Wearable Gestural Interface. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, S. 4111–4116. 2009. doi:10.1145/1520340.1520626. URL `http://portal.acm.org/citation.cfm?doid=1520340.1520626`. (Zitiert auf Seite 16)

[MMZ+03] D. Marculescu, R. Marculescu, N. H. Zamora, P. Stanley-Marbell, P. K. Khosla, S. Park, S. Jayaraman, S. Jung, C. Lauterbach, W. Weber, T. Kirstein, D. Cottet, J. Grzyb, G. Tröster, M. Jones, T. Martin, Z. Nakad. Electronic Textiles : A Platform for Pervasive Computing. *Proceedings of the IEEE*, 91(12):1995–2018, 2003. (Zitiert auf Seite 16)

[MRSS06] U. Maurer, A. Rowe, A. Smailagic, D. P. Siewiorek. eWatch: A Wearable Sensor and Notification Platform. *International Workshop on Wearable and Implantable Body Sensor Networks (BSN'06)*, S. 142–145, 2006. doi:10.1109/BSN.2006.24. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1612916`. (Zitiert auf Seite 24)

[OGOW08] T. Olsson, D. Gaetano, J. Odhner, S. Wiklund. *Open Softwear: Fashionable prototyping and wearable computing using the Arduino.* Online, 2008. (Zitiert auf Seite 15)

[PLTP06] M. Pacelli, G. Loriga, N. Taccini, R. Paradiso. Sensing Fabrics for Monitoring Physiological and Biomechanical Variables: E-textile solutions. In *2006 3rd IEEE/EMBS International Summer School on Medical Devices and Biosensors*, S. 1–4. IEEE, 2006. doi:10.1109/ISSMDBS.2006.360082. URL `ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4201251`. (Zitiert auf Seite 9)

[SCC10] M. Suh, K. Carroll, N. Cassill. Critical Review on Smart Clothing Product Development. *Journal of Textile and Apparel, Technology and Management*, 6(4):1–18, 2010. URL http://ojs.cnr.ncsu.edu/index.php/JTATM/article/view/702. (Zitiert auf den Seiten 13 und 14)

[Sch13] T. de Schutter. The Power of Developing Hardware and Software in Parallel, 2013. URL http://www.design-reuse.com/articles/31951/the-power-of-developing-hardware-and-software-in-parallel.html. (Zitiert auf Seite 14)

[SGG12] D. Silva, M. Ghanem, Y. Guo. WikiSensing: an online collaborative approach for sensor data management. *Sensors (Basel, Switzerland)*, 12(10):13295–332, 2012. doi:10.3390/s121013295. URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=3545568&tool=pmcentrez&rendertype=abstract. (Zitiert auf Seite 20)

[Tao01] X. Tao. *Smart fibres, fabrics and clothing*. Woodhead Publishing Limited, 2001. doi:10.1533/9781855737600. URL http://woodhead.metapress.com/openurl.asp?genre=issue&id=doi:10.1533/9781855737600. (Zitiert auf Seite 12)

[Tiz13] Tizen Association. Tizen, Public Q&A, 2013. URL https://www.tizenassociation.org/PDF/Tizen_FAQ_02.24.13.pdf. (Zitiert auf Seite 19)

[Van14] Vandrico Solutions Inc. Wearables Market Insight Q1 2014. Technischer Bericht, Vandrico Solutions Inc., 2014. URL http://vandrico.com/database. (Zitiert auf den Seiten 13 und 14)

[VC00] K. Van Laerhoven, O. Cakmakci. What shall we teach our pants? In *Digest of Papers. Fourth International Symposium on Wearable Computers*, c, S. 77–83. IEEE Comput. Soc, 2000. doi:10.1109/ISWC.2000.888468. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=888468. (Zitiert auf den Seiten 9 und 16)

[VH04] L. Van Langenhove, C. Hertleer. Smart clothing: a new life. *International Journal of Clothing Science and Technology*, 16(1/2):63–72, 2004. doi:10.1108/09556220410520360. URL http://www.emeraldinsight.com/10.1108/09556220410520360. (Zitiert auf Seite 12)

[Wik14] Wikipedia. Smart Material - Wikipedia, The Free Encyclopedia, 2014. URL http://en.wikipedia.org/wiki/Smart_material. (Zitiert auf Seite 13)

[Zep11a] Zephyr Technology. BioHarness 3.0 User Manual, 2011. (Zitiert auf Seite 21)

[Zep11b] Zephyr Technology. BioHarness Bluetooth Comms Link Specification, 2011. (Zitiert auf den Seiten 43 und 54)

Alle URLs wurden zuletzt am 01. 07. 2014 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift