Technische Universität München

Fakultät für Informatik

Master's Thesis in Informatik

Nutzung von Wearables in IoT Szenarien

Christian Lübben

Technische Universität München

Fakultät für Informatik

Master's Thesis in Informatik

Nutzung von Wearables in IoT Szenarien

Using Wearables in IoT scenarios

Autor	Christian Lübben
Aufgabensteller	Prof. Dr.-Ing. Georg Carle
Betreuer	Dr. Marc-Oliver Pahl, Stefan Liebald M.Sc.
Datum	16. April 2018

Informatik VIII
Lehrstuhl für Netzarchitekturen und Netzdienste

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Garching b. München, 16. April 2018

Unterschrift

Zusammenfassung

Die manuelle Steuerung und Überwachung von Smart Spaces wird durch die steigende Anzahl von Sensoren und Aktoren immer aufwändiger und unübersichtlicher. Für heutige und zukünftige Szenarien sind automatisierte Lösungen gefordert, die die Steuerung des Smart Spaces vereinfachen. Nutzer sollen schließlich den Raum verwenden und möglichst wenig Aufwand in dessen Verwaltung investieren. In dieser Arbeit wird ein voll automatisierter Ansatz verfolgt, bei dem der Nutzer nur bei Bedarf einschreitet, um Aktionen explizit durchzuführen oder zu verhindern. Dies erlaubt dem Nutzer, sich völlig auf die Ausübung seiner Aufgaben zu konzentrieren und schafft gleichbleibend optimale Arbeitsumstände, da bei Erfassung von Abweichungen automatisch gezielte Gegenschritte eingeleitet werden. Ein kritischer Punkt bei solchen Ansätzen sind die Eingabewerte. Um bestmögliche Arbeitsumstände zu gewährleisten, müssen sie so genau wie möglich sein. Um genaue Nutzerparameter zu erfassen, können diese mit Sensoren ausgestattet werden. Durch die weite Verbreitung von Wearables tragen viele Nutzer jedoch bereits eine Reihe von Sensoren am Körper, die für die Steuerung des Smart Spaces nutzbar sein könnten. Das Ziel dieser Arbeit ist die Untersuchung der durch Wearables erfassten Sensorwerte. Während der Untersuchung werden Schwächen repräsentativer Smart Space Szenarien mithilfe von Wearables gelöst. Im Rahmen der Evaluation werden die Vorteile, die sich aus der Nutzung von Wearables ergeben, aufgezeigt.

Abstract

With increasing numbers of sensors and actuators, the challenge of controlling a smart spaces raises. It is infeasible for users to maintain new smart spaces manually. Therefore new strategies are required. The user should be able to focus on his tasks and must not spend time with managing room parameters. While automated solutions are already available it is the data they rely on which needs to be more accurate. To obtain those values users can be equipped with sensors. Since smart watches and wearables are widespread and offer lots of sensors it might be possible to use the values of these sensors to control the smart space. An assumption in this work is that users wear such devices for own purposes except controlling smart spaces and the aim is to use this devices in a paracitic manner. With this assumption it is possible to gain the values along with the planned use. Within this work representative smart space scenarios are identified and investigated to find weaknesses. These weaknesses are then solved with afore identified wearable sensors. In the evaluation part the benefits of using wearables in smart space scenarios are reviewed.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die Digitalisierung unseres Alltags schreitet immer weiter voran. Waren es zunächst Personal Computer, gefolgt von Smartphones, die Einzug in unseren Alltag gehalten haben, bewegen wir uns immer weiter zu einem Internet of Things (IoT), in dem sämtliche Geräte des alltäglichen Lebens miteinander verbunden sind. Vor allem die Entwicklung intelligenter Räume durch Verbindung von Haushaltsgeräten und Gebäudesteuerung sind spätestens seit der Einführung von Amazons *Alexa* oder *Google Home* in aller Munde. Bei genauerer Betrachtung würden wohl wenige vermuten, dass die Idee eines Smart Spaces aus dem letzten Jahrtausend stammt.

Bereits 1988 prägte der Amerikaner Mark Weiser den Begriff des *Ubiquitous computing*, der die Allgegenwart rechnergestützter Informationsverarbeitung bezeichnet. Sind diese verarbeitenden Rechner zusätzlich miteinander verbunden, so wird auch von Rechnerdurchdringung (eng. *Pervasive computing*) gesprochen.

Durch immer kleiner werdende Bauteile für Sensorik und Aktorik, sowie immer zuverlässiger werdenden Funkverbindungen, rückt das Internet der Dinge, das Weiser bereits 1991 in seinem Aufsatz „The Computer for the 21st Century“ [2] beschreibt, in greifbare Nähe.

Die Auswahl der Eingangsparameter zur Steuerung der einzelnen Entitäten ist eine zentrale Fragestellung innerhalb dieser Smart Spaces.

Die Grundlage dieser Arbeit basiert auf der Betrachtung dieser Frage. Die Idee ist es, eine intuitive Benutzerschnittstelle zur Steuerung des Smart Spaces zu finden. Es wird auf Basis aktueller technischer Möglichkeiten betrachtet, welche Verfahren so intuitiv zu bedienen sind, dass Nutzer in der Regel wenig bis keine Einweisung zur korrekten Anwendung benötigen. Zusätzlich soll die Steuerung möglichst komfortabel sein. Durch Komfort wird ausgedrückt, dass sich der Nutzer so wenig wie möglich aktiv um die Raumsteuerung kümmern soll. Er soll seine Aufmerksamkeit auf die Aktivitäten innerhalb des Smart Spaces fokussieren und ihnen ungestört nachkommen können.

Währenddessen soll die Raumsteuerung eine optimale Arbeitsumgebung bereitstellen und Raumparameter selbstständig setzen, wofür präzise Eingabewerte benötigt werden.

Um diese Ziele zu erreichen, werden als neue Quelle zur Erfassung von Eingabewerten aktuelle Wearables betrachtet. Mit einer Vielzahl von Sensoren lassen sich Nutzerparameter erfassen, ohne das der Anwender gezielt mit dem System interagieren muss. Unter der Annahme, dass die intuitivste Steuerung jene ist, um die sich ein Nutzer nicht kümmern muss, sollen die Eingabewerte der Wearables in bestehende Smart Space Szenarien integriert werden. So wird keine Einweisung benötigt und maximaler Komfort erreicht. Zusätzlich sollen durch die Erfassung von Eingabewerten am Nutzer möglichst präzise Steuerungen ermöglicht werden.

1.1 Ziel

In diesem Abschnitt soll zunächst das zugrunde liegende Problem dargestellt werden, das zur Leitfrage dieser Arbeit führt. Nach Formulierung der Leitfrage werden die zur Beantwortung dafür benötigten Schritte vorgestellt und erläutert. Schließlich wird die Gliederung der Arbeit verfasst.

1.1.1 Grundproblem

Wie zuvor bereits angedeutet wurde, besteht eine Herausforderung bei der Steuerung von Smart Spaces in der Interaktion mit dem Nutzer. Diesem soll zur Verwaltung der einzelnen Entitäten so wenig Aufwand wie möglich entstehen. Zusätzlich sollen die automatisch vom System vorgenommenen Steuerbefehle möglichst präzise den aktuellen Anforderungen des Nutzers entsprechen. Diese Präferenzen können im Voraus durch den Nutzer spezifiziert werden und zur Laufzeit durch Anpassungen des Nutzers erfasst werden. Die Wege, über die ein Nutzer mit dem System interagiert, reichen von traditionell im Raum verteilten Schaltern über Smartphone Apps bis hin zur Sprachsteuerung. Den Nachteil, den diese Verfahren gemeinsam haben, ist eine notwendige Interaktion des Nutzers. Gesucht wird demnach eine innovative Nutzerschnittstelle, um diese Interaktionen minimieren zu können. Zu diesem Zweck können Sensoren am Nutzer angebracht werden. So wird dessen Zustand in Form von Nutzerparametern zusätzlich zu den stationär erfassten Raumparametern erfasst. Durch die weite Verbreitung von Wearables tragen bereits viele Nutzer ein Gerät am Körper, dass durch die Entwicklungen der letzten Jahre eine Reihe von Sensoren zur Erfassung dieser Nutzerparameter bietet. Aus der Idee, diese wearablegenerierten Nutzerparameter zur Smart Space Steuerung zu verwenden, ist die im folgenden Abschnitt vorgestellte Leitfrage entstanden.

1.1.2 Leitfrage

Die Leitfrage der Arbeit lautet:

Welche Vorteile bietet die Integration von Wearables in IoT Szenarien?

Für die Beantwortung der Leitfrage sollen Wearables in bestehende Smart Space Szenarien integriert werden. Der Wearableeinsatz soll dabei gezielt Schwächen der Szenarien beheben. So sind nach der Implementierung eines ausgewählten Szenarios eine Version mit und eine Version ohne Wearable verfügbar. Durch den direkten Vergleich dieser beiden Versionen kann der Nutzen des Wearables festgestellt und anhand zuvor festgelegter Kriterien quantifiziert werden. Die Auswahl der betrachteten Szenarien soll repräsentativ sein, um allgemeine Aussagen zum Nutzen zu erlauben.

1.1.3 Anforderungen an erweiterte Szenarien

Um den Nutzen der Wearables evaluieren zu können, müssen durch die entstehenden Wearableszenarien alle folgenden Anforderungen erfüllt werden. Diese sollen sicherstellen, dass die verwendeten Szenarien zur Beantwortung der Leitfrage zulässig sind und ergeben sich aus dem vorhergehenden Abschnitt.

R1: Nutzung eines Wearables—Bei der Erstellung erweiterter Szenarien muss mindestens ein genutzter Sensor ein tragbarer Wearablesensor sein.

Da mithilfe der erweiterten Szenarien der Nutzen eines Wearables evaluiert werden soll, besteht die erste Anforderung an die erweiterten Szenarien darin, dass Eingangswerte eines tragbaren Sensors genutzt werden.

R2: Verbindung zwischen stationären und tragbaren Entitäten—Im erweiterten Szenario muss eine Fusion zwischen tragbaren und stationären Entitäten stattfinden. Das bedeutet, dass das Szenario nicht ausschließlich auf Wearablesensorik basieren darf, sondern Bezug zu einem Smart Space Szenario mit stationären Sensoren und Aktoren herstellen muss.

Da überprüft werden soll, ob IoT Szenarien durch die Nutzung eines Wearables verbessert werden können, muss Bezug zu einem zugrunde liegenden Szenario bestehen, das verbessert werden soll. Die stationären Entitäten dieses Szenarios sollen durch tragbare Entitäten erweitert werden, deren Messwerte in den Entscheidungsprozess zur Steuerung der Aktoren beteiligt sind.

R3: Evaluation des Nutzens—Für das entstehende Szenario müssen Performanzindikatoren festgelegt werden können, die nach der Implementierung ausgewertet werden.

Da der Wearablenutzen bewertet werden soll, muss zur Quantifizierung eine Evaluation durchgeführt werden können.

R4: Wearable muss ein COTS Produkt sein—Das Genutzte Wearable muss ein frei verfügbares kommerzielles Produkt sein.

Da in dieser Arbeit die Annahme besteht, dass ein Nutzer bereits das benötigte Wearable trägt und keine Einschränkung der Usability dadurch entstehen soll, dass der Nutzer extra für die Anwendung im Smart Space ein Gerät anlegen muss, sollen die genutzten Wearables frei verfügbare (commercial off-the-shelf) Geräte sein.

1.1.4 Methodologie

Der erste Schritt zur Beantwortung der Leitfrage ist die Analyse aktueller Wearables, um die **verfügbaren Sensoren zu identifizieren.** Im Rahmen dieses Analyseschritts müssen die aus den Sensoren ableitbaren Nutzerparameter bestimmt werden, die für die Smart Space Steuerung genutzt werden können.

Im nächsten Schritt folgt die **Bestimmung relevanter Smart Space Szenarien**, anhand derer der Wearablenutzen ermittelt wird. Diese Szenarien werden nicht willkürlich gewählt, sondern sollen wissenschaftliche Relevanz besitzen.

Nach Festlegung der zu betrachtenden Szenarien erfolgt die **Analyse des Wearblenutzens**. In diesem Schritt werden die relevanten Szenarien auf Schwächen untersucht. Mithilfe der zuvor identifizierten Eingabewerte der Wearablesensoren wird versucht, diese Schwächen zu beheben. Resultat dieser Optimierungen sind die neuen **Erweiterungen relevanter Szenarien**. Um den erwarteten Nutzen des Wearables quantifizierbar zu erfassen, müssen für jedes Szenario in dieser Phase **Performance Indikatoren festgelegt** werden.

Da der Nutzen des Wearables innerhalb des Smart Space Szenarios nicht nur prognostiziert, sondern explizit evaluiert werden soll, folgt die **Implementierung der relevanten Szenarien, sowie ausgewählter erweiterter Szenarien**. Die Auswahl der zu implementierenden erweiterten Szenarien erfolgt anhand der zu Beginn definierten Anforderungen. Um ein Rapid Prototyping zu ermöglichen, wird das Distributed Smart Space Orchestration System (DS2OS) verwendet.

Wie stark die Nutzung von DS2OS die Implementierung erleichtert, soll während der Implementierung der repräsentativen Smart Space Szenarien evaluiert werden. Das Hauptkriterium während der **Evaluation des DS2OS** wird die Usability sein. So kann neben der Beantwortung der Leitfrage am Ende der Arbeit nicht nur eine Aussage über den Nutzen von Wearables und den möglichen Usability Gewinn für den Nutzer getroffen werden, sondern zusätzlich auch, mit welchem Aufwand die betrachteten

Szenarien implementiert werden konnten. Mit diesen Ergebnissen lassen sich Aussagen zum Verhältnis des zusätzlichen Nutzens eines Wearables im Vergleich zum Implementierungsaufwand treffen. Die Evaluation bei der Implementierung repräsentativer Szenarien lässt zudem allgemeine Aussagen zur Nutzbarkeit des DS2OS zu.

Da es sich bei der Evaluation des DS2OS um einen Nebenaspekt der Arbeit handelt, soll bei Festlegung der Evaluationskriterien auf bereits vorhandenes Wissen zurückgegriffen werden. Ziel dieser Arbeit ist nicht die Analyse von Qualitätskriterien zur Softwareevaluation.

Im Rahmen dieser Masterarbeit werden zusätzlich zu den ursprünglichen Szenarien schließlich zwei **Wearableszenarien exemplarisch implementiert**. Dadurch können **ursprüngliches und erweitertes Szenario miteinander verglichen** werden. Der Mehrwert des Wearables kann so nicht nur analytisch, sondern real am vorhandenen Objekt ausgewertet werden. Für die Evaluation der Performance Indikatoren werden **Testfälle entwickelt**, die für jeden Indikator ein quantifizierbares Ergebnis liefern.

Die durch die Auswertung der Testfälle sollen die zuvor aufgestellten **Hypothesen getestet** werden. Mit den Ergebnissen soll sich der Mehrwert des Wearables am Ende dieser Arbeit beziffern lassen und eine **Antwort auf die Leitfrage** getroffen werden.

1.2 Gliederung

Nach der Einleitung werden in Kapitel 2 zunächst Wearables und Smart Space Szenarien untersucht und daraufhin miteinander kombiniert. Kapitel 3 stellt verwandte Arbeiten vor, die ebenfalls Wearables in IoT Szenarien anwenden. In Kapitel 4 erfolgt das Design der Testfälle für die Evaluation des Wearablenutzens sowie die Vorbereitung für die Implementierung. Das Kapitel 5 beinhaltet Details zur Implementierung. Die Auswertung der Testergebnisse erfolgt in Kapitel 6, woraufhin im Kapitel 7 die Antwort auf die Leitfrage gegeben wird.

Die Arbeit ist im Folgenden so aufgebaut, dass zu Beginn eines Abschnittes zunächst auf dessen Relevanz in der Gesamtarbeit eingegangen wird. Es wird kurz zusammengefasst, was in dem Abschnitt passiert und zu welchem Zweck diese Schritte ausgeführt werden. Am Ende eines jeden Abschnitts werden die erhaltenen Ergebnisse betrachtet und deren Beitrag zur Beantwortung der Leitfrage aufgezeigt.

1.3 Meilensteine

Aus der Methodologie lässt sich eine Liste von Meilensteinen bilden, die zur Beantwortung der Leitfrage erfüllt werden müssen. Mithilfe der Gliederung lassen sich diese wie folgt gruppieren.

Analyse:

- Sensoren aktueller Wearables identifizieren
- Relevante Szenarien identifizieren
- Erweiterte Wearable Szenarien erstellen
- Hypothesen über Wearablenutzen aufstellen
- Festlegen von Performance Indikatoren
- Auswahl zu implementierender Szenarien
- Vorstellung des DS2OS

Design:

- Festlegen der DS2OS Evaluationskriterien
- Vorbereitung Softwareevaluation
- Erheben der Daten: Design
- VSL Design der Szenarien
- Erstellung der Testfälle

Implementierung:

- Erheben der Daten: Implementierung
- Implementierung der allgemeinen Szenarien
- Implementierung der Wearable Szenarien

Evaluation:

- Auswertung der Testergebnisse
- Testen der Hypothesen

Fazit:

- Beantworten der Leitfrage

Kapitel 2

Analyse

Die Analyse der Arbeit beginnt mit der Definition grundlegender Begriffe in Abschnitt 2.1.

Kapitel 2.2 beinhaltet die geschichtliche Entwicklung von Smart Spaces und den aktuellen Stand der Technik.

Es folgt die Analyse aktueller Wearables in Abschnitt 2.3. Diese werden auf ihre verbauten Sensoren hin untersucht. Aus dieser Analyse wird eine Liste genereller Sensorik in Wearables gebildet, die in den folgenden Schritten benötigt wird.

In Kapitel 2.4 werden relevante Smart Space Szenarien durch die Betrachtung vorhergehender Smart Space Projekte identifiziert. Es erfolgt die Auswahl der zu implementierenden Szenarien und den bei der Umsetzung benötigten Sensoren und Aktoren.

Nach der Bestimmung relevanter Szenarien werden diese in Kapitel 2.5 auf Schwächen und Verbesserungsmöglichkeiten mittels Wearables untersucht. Für jeden Sensor auf der in Abschnitt 2.3 erstellten Liste wird geprüft, wie er sich im jeweiligen Szenario einsetzen lässt. In diesem Schritt werden die Hypothesen über den erwarteten Nutzen des Wearableeinsatzes aufgestellt. Daraufhin werden Wearableszenarien für die Implementierung ausgewählt und Performaceindikatoren festgelegt.

Da die Implementierung mithilfe des DS2OS und dessen Virtual State Layer Middleware durchgeführt und dieses ebenfalls evaluiert wird, folgt dessen Vorstellung in Kapitel 2.6.

2.1 Definitionen

Um Unklarheiten, die aus Definitionen anderer Quellen resultieren könnten, auszuschließen, werden die zentralen Begriffe dieser Arbeit zunächst definiert.

Smart Spaces—Der im Folgenden verwendete Begriff eines Smart Spaces umfasst einen physischen, räumlich begrenzten Bereich, der verschiedene Parameter besitzt, die durch Sensoren erfasst und durch Aktoren beeinflusst werden können.

Wearables—Als Wearables werden im Folgenden tragbare Geräte mit Sensoren und Aktoren bezeichnet, die der Nutzer am Körper trägt. Die von den Sensoren gemessenen Parameter des Trägers können als Input für die Smart Space Steuerung genutzt werden. In dieser Arbeit wird der Fokus auf Wearables gelegt, die am Handgelenk getragen werden (engl. wrist worns).

Beispiele für Wrist worns sind neben dedizierten Sensoren oder Aktoren Fitnessbänder, Smart Watches und intelligente Armbänder.

Entität—Als Entitäten werden Elemente der Kategorien Aktoren und Sensoren zusammengefasst. Eine Entität ist demnach ein Objekt, das einen Eingabewert liefert oder einen Ausgabewert verarbeitet. Ein Smart Space besteht aus einer Reihe von Entitäten.

2.2 Smart Spaces

Dieser Absatz bietet eine kurze Einleitung in den Bereich der Smart Spaces. Es wird auf vergangene Meilensteine, sowie den aktuellen Stand eingegangen. So wird der Rahmen, in den sich diese Arbeit einfügt, zunächst vorgestellt.

2.2.1 Geschichte

Die Idee des allgegenwärtigen Rechnens (Eng: Ubiquitous computing) existiert bereits seit Ende der 1980er Jahre und wurde durch Mark Weiser geprägt. In [2] beschreibt er, wie Computer immer mehr Einzug in das tägliche Leben der Menschen halten. Erste Ansätze, einen intelligenten Raum durch die Verwendung und Verknüpfung von Sensoren und Aktoren zu schaffen, bestehen seit den 1990er Jahren. Es handelt sich demnach beim Begriff der Smart Spaces keinesfalls um eine neue Idee. Jedoch bieten die technischen Entwicklungen der letzten Jahre eine Vielzahl neuer Möglichkeiten. Vor allem die Entwicklung kostengünstiger, frei programmierbarer Rechner wie des Arduino oder des Raspberry PI erlauben die Erstellung eines Smart Spaces mit geringem Kosten- und Energieaufwand. Es erlaubt den Nutzern unabhängig von herstellerspezifischen Lösungen eigene Smart Spaces zu implementieren. Zudem ist es durch die Einfachheit der Programmierung auch für Anfänger und Nichtinformatiker möglich, mit diesen Geräten umzugehen. Dies ermöglicht es, Smart Spaces nach eigenen Vorstellungen einfach selbst zu implementieren. Zudem erlaubt die Entwicklung immer verlässlicherer und stromsparender Drahtlosverbindungen die Herstellung kleinerer

und einfach einzubindender Sensoren. Die Vision vom Internet der Dinge (engl. Internet of Things / IoT) rückt durch diese Fortschritte zunehmend in greifbare Nähe.

Bei Betrachtung der Anzahl wissenschaftlicher Veröffentlichungen zeigt sich ebenfalls ein Bild der steigenden Popularität im Bereich der Smart Space Forschung. [3]

2.2.2 Stand der Technik

Aktuell steht der Begriff des Smart Spaces vor allem bei der Vermarktung von Smart Home Lösungen im öffentlichen Fokus. Eine Reihe von Herstellern, wie zum Beispiel der Telekom, Bosch oder AVM, bieten eigene Smart Home Systeme an. Diese Systeme sind jedoch zumeist auf die Nutzung eigener Komponenten beschränkt und bieten oft nur eingeschränkte Interoperabilität mit fremden Komponenten. Diesem entgegen steht eine Do-It Yourself Bewegung, die mittels freier Hard- und Software Smart Space Entwicklung betreibt.

Smart Spaces sind jedoch keinesfalls auf Heimanwender beschränkt. Im Rahmen der industriellen Digitalisierung und der Industrie 4.0 sind Smart Spaces beispielsweise in Form von intelligenten Fabriken (engl. Smart Factories) zu finden. Dies sind Produktionsstätten, die weitreichend ohne die Interaktion mit dem Menschen arbeiten.

Moderne Autos können ebenfalls als Smart Space angesehen werden. Immer mehr Assistenzsysteme nehmen dem Nutzer Aufgaben ab oder unterstützen diesen.

2.2.3 Zusammenfassung

Es gibt bereits eine Vielzahl von Projekten im Bereich der Smart Spaces. Diese Arbeit soll das Vorhandene um den Nutzen von Wearables erweitern. Der Fokus liegt darauf, Sensorik zu benutzen, die ein Nutzer ohnehin schon bei sich trägt.

2.3 Wearables

Seit Veröffentlichung des ersten Fitbit Activity Trackers im Jahr 2009 oder mit der Pebble die erste kommerziell erfolgreiche Smartwatch im Jahr 2013 vorgestellt wurde, haben sich diese Wearables stetig weiterentwickelt. Heutzutage bieten sie auf der Fläche einer herkömmlichen Armbanduhr eine Reihe von Sensoren zur Erfassung von Nutzerparametern.

Um eine Antwort auf die Leitfrage finden zu können, welche Vorteile Wearables in Smart Space Szenarien bieten, müssen zunächst die verfügbaren Messwerte analysiert werden. Diese sollen schließlich später für die Smart Space Steuerung genutzt werden und einen Mehrwert gegenüber der stationären Sensorik generieren.

Da aktuelle Wearables zahlreiche Sensoren bieten, deren Messwerte Aufschluss über den Zustand des Nutzers oder dessen Umgebung geben, wird in diesem Abschnitt zunächst ein Überblick über aktuelle Wearables und deren verbaute Sensoren gegeben. Daraufhin werden diese im Detail vorgestellt. Es wird auf den Aufbau, die Messwerte und deren Verwendbarkeit eingegangen.

Ziel dieses Abschnitts ist es, die Sensoren aktueller Wearables zu identifizieren und in einer Liste zusammenzufassen.

Es wird folgender Meilenstein erreicht:

- Sensoren aktueller Wearables identifizieren

2.3.1 Überblick

Für die Aufzeichnung der Nutzerparameter eignen sich aktuelle Wearables wie Smartwatches, Fitnessbänder oder Bracelets(Armbänder), da diese über ein breites Spektrum an Sensoren verfügen, frei verfügbar sind und zunehmend in den Alltagsgebrauch übergehen. Es gibt bereits eine Vielzahl von Geräten in unterschiedlichen Preiskategorien.

2.3.1.1 Wrist Worns

Neben den Wrist Worn Wearables existieren eine Vielzahl weiterer Wearables, wie beispielsweise Smart Glasses oder Smart Garnments.

Der Fokus auf Wrist Worns in dieser Arbeit ist darin begründet, dass gerade diese Kategorie in den letzten Jahren ein rasantes Wachstum erfahren hat. Die weite Verbreitung dieser Geräte lässt die Annahme zu, dass ein Nutzer bereits ein Gerät dieser Kategorie aus eigenem Interesse trägt und dieses nicht extra zur Smart Space Steuerung anlegen muss. Dies führt zu der Hypothese, dass das Risiko einer mangelnden **Nutzerakzeptanz** zur Verwendung dieser Geräte minimiert wird, da der Nutzer sich bereits entschieden hat, ein solches Gerät zu tragen. Zusätzlich ermöglicht die Annahme das Argument, dass lediglich bereits **vorhandene Sensorik** mitbenutzt wird. Dem Nutzer entsteht demnach kein zusätzlicher Aufwand für die Smart Space Steuerung.

Die zuvor genannten Kriterien würden auch von Smartphones erfüllt werden. Sie sind sogar noch weiter verbreitet als Wearables und besitzen ebenfalls ein breites Spektrum an Sensoren. Der entscheidende Vorteil, den Wrist Worns gegenüber Smartphones aufweisen, ist die feste Lokalisierung und Orientierung am Körper des Nutzers. Während Wrist Worns ausschließlich am Handgelenk getragen werden, können Smartphones an den verschiedensten Stellen am Körper oder in separaten Taschen getragen werden. Die korrekte Auswertung der Sensorwerte wird somit erheblich erschwert. Bei der

Erkennung von Gesten ist beispielsweise die Ausgangsposition des Geräts nicht bekannt oder variiert.

2.3.1.2 COTS Produkte

Um eine Liste von verbauten Sensoren auf dem aktuellen Stand der Technik zu erstellen, werden aktuell erhältliche Wearables untersucht. Gestützt durch die Annahme, dass Nutzer bereits ein Wearable zur Erfüllung eigener Ziele besitzen, das für die Smart Space Steuerung mitbenutzt wird, werden zur Erstellung der Sensorenliste bereits verfügbare kommerzielle (Commercial off-the-shelf / COTS) Produkte untersucht.

Die Nutzung von COTS Produkten bietet den Vorteil, dass es sich um Massenware handelt, was sich positiv auf die Verbreitung und den Preis auswirkt. So sind bereits günstige Wearables verfügbar (Tab. 2.1, Kategorie Preis). Zusätzlich ermöglicht die Auswahl eines weit verbreiteten Wearables für die Implementierung die Anbindung vieler einzelner Geräte des gleichen Typs. Durch das Erreichen der Serienreife ist außerdem davon auszugehen, dass sich die Geräte auf einem technisch ausgereiften Level befinden, was im Folgenden vor allem für die Genauigkeit der Messwerte von Bedeutung ist.

Vorteile von COTS Produkten:

- Bereits verfügbar
- Nutzerakzeptanz
- Weit verbreitet
- (relativ) Günstig
- Technik (relativ) ausgereift

2.3.1.3 Aktuelle Wearables

Die Tabelle 2.1 enthält die analysierten Wearables. Es wird aufgelistet, welche Sensoren jeweils verbaut sind. Zusätzlich werden der Preis sowie das verwendete Betriebssystem angegeben, das bei der Betrachtung der APIs in Abschnitt 2.3.4.2 von Bedeutung ist. Aus Gründen der Übersichtlichkeit wurden für einige Sensoren verkürzte Namen verwendet.

Pro Sensor gibt es drei mögliche Einträge:

- ✓: Sensor vorhanden (* = deaktiviert)
- —: Keine Information verfügbar
- X: Sensor nicht vorhanden

Tabelle 2.1: Sensoren aktueller Wearables

Gerät	Accelero	Gyroskop	Schritte	H-Freq	Oxy	Skin	Temp	Helligkeit	Mic	Betriebssystem	Preis
Fitnessbänder											
Fitbit Charge 2	✓	X	✓	✓	X	X	X	X	X	Fitbit OS	160
Fitbit Flex 2	✓	X	✓	X	X	X	X	X	X	Fitbit OS	80
Xiaomi Mi Band 2	✓	X	✓	✓	X	X	X	X	X	Proprietär	30
Garmin Vivosmart 3	✓	X	✓	✓	X	X	X	✓	X	Proprietär	150
Jawbone UP 3	✓	X	✓	✓	X	✓	✓	X	X	Proprietär	40
Mio Fuse	✓	X	✓	✓	X	X	X	X	X	Proprietär	130
Smartwatches											
Apple Watch 3	✓	✓	✓	✓	✓*	X	X	✓	✓	WatchOS	370
Fitbit Surge	✓	✓	✓	✓	X	X	X	✓	X	Proprietär	250
Fitbit Blaze	✓	X	✓	✓	X	X	X	✓	X	Proprietär	200
Garmin Vivoactive HR	✓	X	✓	✓	X	X	X	✓	X	Proprietär	270
Samsung Gear S3	✓	✓	✓	✓	X	X	X	✓	✓	Tizen	300
Android Wear											
Polar M600	✓	✓	✓	✓	X	X	X	✓	✓	Android Wear 2.0	230
Huawei Watch 2	✓	✓	✓	✓	X	X	X	✓	✓	Android Wear 2.0	280
Alcatel OneTouch	✓	✓	✓	✓	X	X	X	X	✓	Android	150
Sony Smartwatch 3	✓	✓	✓	X	X	X	X	✓	✓	Android Wear < 2	160

2.3.2 Sensoren

Die in Abschnitt 2.3.1.3 gefundenen Sensoren werden im Folgenden detailliert vorgestellt. Es werden der jeweils erhobene Messwert sowie Anwendungsmöglichkeiten betrachtet.

Die Reihenfolge erfolgt gruppiert nach bewegungserfassender Sensorik (Accelerometer, Gyroskop, Schritte), gefolgt von vitalwerterfassenden Sensoren (Herzfrequenz, Oxymeter, Hautwiderstand, Temperatur) und zum Schluss umgebungserfassender Sensorik (Helligkeit, Mikrofon).

Accelerometer (Accelero)

Das Accelerometer ist der am häufigsten verbaute Sensor. Alle in Tabelle 2.1 betrachteten Wearables verfügen über diesen Sensor.

Mit dem Accelerometer, auch Beschleunigungssensor oder G-Sensor, werden lineare Beschleunigungen entlang einer oder mehrerer Achsen gemessen. Die Angabe erfolgt in der Regel in den Einheiten $\frac{m}{s^2}$ oder g.

Mithilfe der aufgezeichneten Werte können Nutzerbewegungen erkannt werden. Durch die Aufzeichnung mehrerer Achsen können auch komplexere Aktivitäten erkannt werden. So wurden in [4] Accelerometer und Gyroskope genutzt, um Aktivitäten von Arbeitern in einem Baustellenumfeld zu erkennen. Die Aufzeichnung mehrerer Personen innerhalb der Testumgebung erlaubt zudem die Erkennung komplexer Aktivitäten mit mehreren Teilnehmern. [5]

Durch die Analyse von Messwerten über einen längeren Zeitraum können ebenfalls Änderungen im allgemeinen Verhalten, beispielsweise durch gesundheitliche Änderungen, erkannt und überwacht werden. [6]

Im Rahmen der Smart Space Steuerung kann die Erkennung bestimmter Bewegungen durch das Accelerometer zur Steuerung der Aktoren verwendet werden. Dieses kann automatisiert auf Basis einer Aktivitätenerkennung geschehen oder spezielle Gesten zur manuellen Auslösung einzelner Aktionen ermöglichen.

Gyroskop

Das Gyroskop, auch Kreiselinstrument, dient zur Lagebestimmung. Es können Änderungen durch Neigung und Bewegung erkannt werden.

Das Gyroskop kann, wie auch das Accelerometer, zur Erkennung von Nutzerbewegungen und Aktivitäten genutzt werden. Die Kombination dieser beiden Sensoren trägt zur

genaueren Bestimmung der Bewegungen bei und lässt es zu, noch spezifischere Gesten oder Aktivitäten zu erkennen.

Schrittzähler

Da viele Wearables mit einer Sensorik zur Schrittzählung werben, wurde dieser als extra Sensor aufgeführt. Ein dedizierter Sensor existiert jedoch in der Regel nicht. Es werden Sensoren des Mikroelektromechanischen Systems (MEMS), in der Regel Gyroskope und Accelerometer, zur Schrittzählung verwendet. Da jedes Gerät aus Tabelle 2.1 mindestens über ein Accelerometer verfügt, besitzen auch alle einen Schrittzähler.

Die Anzahl der Schritte kann Aufschluss über das generelle Aktivitätslevel des Nutzers geben. Schrittmessungen wurden bereits erfolgreich zur Bestimmung der Aktivität von Probanden verwendet. So wurde die Schrittzahl von Patienten, die sich in einer Rehabilitationsphase befanden, überwacht, um deren Fortschritte zu erfassen. [7]

Der Schrittzähler kann demnach für die Bestimmung eines Aktivitätslevels sowie der Geste „Gehen“ genutzt werden.

Herzfrequenz (H-Freq)

Der Herzfrequenzsensor misst die Anzahl der Herzschläge des Nutzers. Der Puls wird in der Regel in Schlägen pro Minute (beats per minute, bpm) angegeben.

Dieser Wert ist ein wichtiger Vitalparameter zur Nutzerüberwachung. Er lässt eine Auswertung des physischen Zustands der Person zu. Es kann Anstrengung oder Aufregung gemessen werden. Die Werte können mit dem Soll im Ruhebereich oder zuvor gesammelten Referenzwerten verglichen werden. Bei Abweichungen können mithilfe weiterer Sensoren Gründe für die Abweichung ermittelt werden. In [7] wurden beispielsweise die Werte von Patienten, die sich in einer Rehabilitationsphase befinden, über etwa 14 Tage beobachtet, um Aufschluss über die physische Aktivität der Patienten während der Genesung zu erhalten. Treten akute Änderungen auf, könnten innerhalb der Smart Space Steuerung Gründe mithilfe anderer Sensoren gesucht werden. So können beispielsweise Temperatur und Lautstärke überprüft und angepasst oder bei extremen Abweichungen ein Alarm gegeben werden.

Oxymeter (Oxy)

Das Oxymeter dient zur Bestimmung des Blutsauerstoffwertes. Dieser lässt Schlüsse auf die Atemfunktion des Nutzers zu und kann neben gesundheitlich bedingten Störungen auch durch körperliche Aktivität oder Arbeiten in sauerstoffarmen Umgebungen wie beispielsweise großer Höhe beeinflusst werden. Gemessen wird der Wert mithilfe

von Leuchtdioden, die Licht einer bestimmten Wellenlänge aussenden. Eine Fotodiode erfasst, welche Wellenlängen des ausgesendeten Lichts von Hämoglobin im Blut absorbiert werden und bestimmt so den Sauerstoffwert.

Wie die Herzfrequenz gibt auch der Blutsauerstoffwert Aufschluss über den Vitalzustand des Nutzers. Bei Abweichungen können ebenfalls Maßnahmen zur Korrektur oder ein Alarm eingeleitet werden.

Hautwiderstandssensor (Skin)

Der Hautwiderstandssensor, auch Skin Conductance Sensor, misst den Hautleitwiderstand. Dieser ändert sich bei Schweißproduktion.

Der Wert kann Aufschluss über den Stresspegel oder Überhitzung des Nutzers geben. Es kann somit nach Gründen für ein erhöhtes Stresslevel gesucht werden. Zusätzlich kann Überhitzung durch Kontrolle der Raum- und Körpertemperatur ausgeschlossen werden und gegebenenfalls mit den zuständigen Aktoren korrigiert werden.

Körpertemperatur (Temp)

Der Körpertemperatursensor misst die Hauttemperatur des Nutzers. Dieser Wert kann für die Erkennung von Anstrengung oder Überhitzung genutzt werden. Das Ergebnis kann zur Steuerung der Heizung und Klimatisierung verwendet werden.

Helligkeit

Die Messung der Helligkeit ist ein wichtiges Kriterium, um im Zusammenspiel zwischen natürlicher und künstlicher Beleuchtung eine ausgewogene Ausleuchtung des Raumes gewährleisten zu können. Zu helle Räume können der Konzentrationsfähigkeit des Nutzers schaden und verursachen vermeidbare Energiekosten. Zu dunkle Räume erschweren die Arbeit und fördern Müdigkeit. Die Helligkeit kann als Eingabewert für die Beleuchtungssteuerung sowie die Verdunkelung mittels Jalousien genutzt werden. Da ein Großteil der Wearables aus Tabelle 2.1 einen Sensor zur Bestimmung der Helligkeit besitzen, muss die Messung der Helligkeit im Smart Space nicht mehr zwingend stationär erfolgen.

Mikrofon (Mic)

Die Lautstärke eines Raums beeinflusst in entscheidendem Maße die Konzentrationsfähigkeit des Nutzers. Bei Feststellen erhöhter Lautstärke sollten geräuscherzeugende

Aktoren des Smart Spaces, wie die Belüftung, als Ursache ausgeschlossen werden. Im weiteren Verlauf kann als Aktion ein Alarm an den Nutzer ausgelöst werden.

Zusätzlich bietet ein Mikrofon die Möglichkeit Sprachbefehle aufzuzeichnen.

2.3.3 Sensorenliste

Die Sensoren aus 2.3.2 werden in diesem Abschnitt zu einer Liste zusammengefasst. Diese Liste bildet die Basis für die Bildung erweiterter Szenarien und die spätere Evaluation des Wearablenutzens in IoT Szenarien.

In Tabelle 2.2 sind die Sensoren mit den jeweiligen Anwendungsbereichen aufgeführt.

Tabelle 2.2: Sensorenliste

Sensor	Anwendung
Accelerometer	Bewegung, Gesten, Aktivitäten
Gyroskop	Bewegung, Gesten, Aktivitäten
Schrittzähler	Bewegung, Gehen
Herzfrequenz	Anstrengung, Aufregung
Oxymeter	Anstrengung
Skin Conductance	Stresspegel, Überhitzung
Körpertemperatur	Anstrengung, Überhitzung
Helligkeit	Wohlbefinden, Leistungsfähigkeit
Mikrofon	Lärmmessung, Wohlbefinden

2.3.4 Zugriff auf Rohdaten

In diesem Abschnitt wird der Zugriff auf die Rohdaten der Wearables genauer betrachtet. Dieser stellt ein wichtiges Kriterium für die Auswahl eines Wearables für die Implementierung dar. Es werden zunächst verschiedene Zugriffsmodelle vorgestellt. Abschließend werden APIs bekannter Hersteller untersucht.

2.3.4.1 Zugriffsmodelle

Hersteller bieten in der Regel verschiedene Zugriffsmöglichkeiten auf die Daten ihrer Wearables. Diese variieren im Level der Daten sowie der Zugriffszeit.

Datenlevel—Um die Sensorik eines Wearables zu nutzen, müssen die Werte der Sensoren ausgelesen werden. Dies kann auf unterschiedlichen Abstraktionsebenen geschehen. Zum einen können die direkten *low-level* Sensorrohdaten ausgelesen werden und zum anderen *high-level* Daten wie beispielsweise berechnete Schritte oder Aktivitätslevel,

die aus den zugrunde liegenden Vitalparametern berechnet wurden. Bei *higher-level* Daten werden jedoch in der Regel die zugrunde liegenden Werte nach der Berechnung verworfen und sind nicht mehr zugreifbar.

Um das volle Spektrum der Wearablesensorik nutzen zu können, wird Zugriff auf die Rohdaten benötigt, also auf einem niedrigen Level des Wearables. Der Zugriff, den Hersteller über ihre APIs zulassen, variiert in diesem Punkt sehr stark zwischen vollem Zugriff auf die Sensorrohdaten bis zum bloßen Erstellen neuer Watchfaces.

Zugriffszeit—Bei der Erfassung von Sensordaten besteht die Möglichkeit, diese in Echtzeit oder verzögert zu verarbeiten. Echtzeitwerte werden in Systemen benötigt, in denen direkt auf einen Input des Wearables reagiert werden soll. Ein Beispiel ist eine Gestensteuerung. Zur Verifikation und zum Training von lernbasierten Verfahren können Sensordaten auch verzögert verarbeitet werden. Bei einer lernbasierten Heizungssteuerung kann beispielsweise mithilfe der verzögerten Auswertung der Raumtemperatur analysiert werden, wie sich diese im betrachteten Zeitraum verhalten hat, um den eingesetzten Algorithmus zu trainieren.

2.3.4.2 Programmierschnittstelle (API)

Neben der Betrachtung der Sensorik ist ebenfalls das Betriebssystem (OS) der Wearables von Bedeutung. Dieses ist zwar für das Szenario unerheblich, jedoch für die spätere Implementierung wichtig. Die Nutzung eines weit verbreiteten OS mit einer einheitlichen API erlaubt Kompatibilität mehrerer Geräte zur Steuerung. Zusätzlich muss es in einigen Fällen möglich sein, Sensordaten in Echtzeit oder mit nur geringer Verzögerung zu erfassen.

Viele Hersteller bieten APIs für den Zugriff auf ihre Wearables an. Diese bieten unterschiedliche Möglichkeiten und Level, um auf die Daten des Wearables zuzugreifen. Im Folgenden wird eine Auswahl von APIs bekannter Hersteller vorgestellt und auf deren Funktionsumfang eingegangen.

Laut [8] sind die aktuell und zukünftig (Stand 2016/Prognose 2020) am weitest verbreiteten Smartwatch Betriebssysteme:

- 1: watchOS (52.3%/43.8%)
- 2: Android/Android Wear (22.9%/41.8%)

Zusätzlich werden die APIs der im Segment der Fitnesstracker verbreiteten Hersteller Xiaomi und Fitbit untersucht (Stand 2017). [9]

- 1: Xiaomi 14.7%
- 2: Fitbit 12.3%

watchOS—Apple bietet den Zugriff auf Bewegungssensorik wie Accelerometer, Gyroskop und Schritte in Echtzeit. Ein Weg, um auf weitere Sensoren der Apple Watch zuzugreifen, konnte nicht gefunden werden. Zusätzlich wird zur Erstellung von Apps ein Mac mit OS/X benötigt. [10]

Android Wear—Android Wear bietet vollen Zugriff auf sämtliche verbauten Sensoren. Eine ausführliche Dokumentation hierzu befindet sich unter [11]. Der Zugriff ist in Echtzeit möglich.

Xiaomi—Xiaomi bietet keine öffentliche API für den Zugriff auf Rohdaten der Sensoren. [12] Von Xiaomi werden keinerlei Informationen bereitgestellt, um auf Sensoren zuzugreifen. Auch weitere Recherchen weisen darauf hin, dass auf keine Sensoren zugegriffen werden kann.

Fitbit—Fitbit bietet keinen Zugriff auf die Rohdaten der Sensoren [13]

In Tabelle 2.3 werden die Ergebnisse der API Analyse zusammengefasst.

Tabelle 2.3: Wearable Apis

Api	RAW-Zugriff	Latenz
WatchOS	Ja, Bewegungssensorik	Echtzeit
Android/Wear	Ja, voller Sensorzugriff	Echtzeit
Xiaomi	Nein	—
Fitbit	Nein	—

2.3.5 Zusammenfassung

In diesem Abschnitt wurde der Meilenstein

- Sensoren aktueller Wearables identifizieren ✓

erreicht, indem aktuelle Wearables und deren vorhandene Sensorik vorgestellt wurde. Es wurde eine Liste von Sensoren gebildet, mithilfe derer nun die relevanten Szenarien erweitert werden können.

Zusätzlich wurden APIs verschiedener Hersteller im Punkt Sensorzugriff untersucht. Diese Ergebnisse sind relevant für die spätere Auswahl eines Geräts für die Implementierung.

2.4 Smart Space Szenarien

Laut 2.1 beschreibt ein Smart Space einen Raum mit der Gesamtheit der vorhandenen Sensoren und Aktoren. Die Implementierung einer bestimmten Funktionalität wird im Folgenden als Szenario bezeichnet. Zu jedem Smart Space gehört demnach mindestens ein Szenario, das dessen Grundfunktion implementiert. Innerhalb eines Smart Spaces können jedoch auch mehrere Szenarien implementiert sein. Abhängig sind die implementierbaren Szenarien in erster Linie von den zur Verfügung stehenden Sensoren und Aktoren.

Im Folgenden sollen zunächst relevante Smart Space Szenarien identifiziert werden. Dazu werden vorhergehende Projekte im Bereich Smart Spaces auf deren betrachtete Szenarien untersucht. Durch diese Analyse soll einerseits die Grundlage zur Auswahl der zu betrachtenden Szenarien geschaffen werden, andererseits könnten vorhergehende Projekte eine Vergleichsreferenz bei der Evaluation bilden und bieten Key Performance Indikatoren, an denen die in dieser Arbeit implementierten Szenarien ebenfalls bemessen werden können. Es folgt die Auswahl der zu implementierenden Szenarien und den bei der Umsetzung benötigten Sensoren und Aktoren.

Im Gesamtkontext der Arbeit soll in diesem Abschnitt der Meilenstein

- Relevante Szenarien identifizieren

erreicht werden.

Für die Bestimmung relevanter Szenarien wird untersucht, welche Szenarien bei früheren Forschungsprojekten behandelt wurden. Diese wissenschaftliche Grundlage für die Auswahl soll garantieren, dass repräsentative Szenarien ausgewählt werden. Den gefundenen Szenarien wurde demnach in der Vergangenheit solche Priorität eingeräumt, dass Forschungsprojekte zu deren Optimierung durchgeführt wurden.

In [3] bietet Härtinger bereits eine Auflistung und Auswertung allgemeiner Smart Space Szenarien. Diese soll als Grundlagen für diese Analyse dienen. Zunächst werden dort Kriterien für die Auswahl von Szenarien vorgestellt, die auch in dieser Arbeit Anwendung finden sollen.

Laut dieser Kriterien muss ein Szenario einen Smart Space wie in Definition 2.1 beinhalten und entweder real implementiert oder zumindest mittels realer Sensordaten simuliert werden.

Da Härtinger in seiner Analyse bereits Smart Space Szenarien bis einschließlich 2014 betrachtet hat, werden diese Ergebnisse zunächst zusammenfassend vorgestellt und für diesen Zeitraum als vollständig angenommen. Es werden in dieser Arbeit somit primär neue Szenarien ab 2014 gesucht und im zweiten Abschnitt vorgestellt.

2.4.1 Szenarien bis 2014

In seiner Arbeit gruppiert Härtinger die Szenarien in sechs Kategorien, um eine gemeinsame Vergleichsgrundlage zu schaffen. Darauffolgend werden für jede Kategorie die implementierten Szenarien vorgestellt.

In diesem Abschnitt folgt eine Zusammenfassung der Ergebnisse aus [3].

Living Labs—Die Gruppe der Living Labs beinhaltet Projekte, in denen Daten aus Smart Spaces gesammelt werden, die alltäglichen Nutzerumgebungen nachempfunden sind. In diesen Umgebungen werden in der Regel Testpersonen überwacht und Daten aufgezeichnet.

Ein betrachtetes Szenario ist die Lokalisierung von Personen. Zu diesem Zweck wurden Daten von im Boden platzierten Drucksensoren (Georgia Tech Aware Home) oder mithilfe von Ultraschall (Matilda's Smart House) erfasst. Ein weiteres Szenario ist die Identifikation von Personen und Gegenständen durch die Nutzung von RFID-Tags (Innovative Retail Laboratory), um daraufhin Aktionen, wie beispielsweise das Öffnen der Haustür, einzuleiten (Gator Tech Smart House).

AAL and Health Care—Im Bereich Gesundheit und unterstütztem Wohnen (Ambient Assisted Living) befassen sich viele Projekte mit der Überwachung der Nutzer und deren Aktivitäten, ohne eine Aktion durchzuführen. Ziel ist es Anomalitäten festzustellen und gegebenenfalls eingreifen zu können. Zur Lokalisierung der Nutzer wurden Infrarot-, Tür- und Bewegungssensoren eingesetzt. Zur Identifikation wurden Ultraschallsensoren im Türrahmen genutzt, um anhand der Größe der eintretenden Person diese zu identifizieren (Using Height Sensors for Biometric Identification in Multi-resident Homes).

Convenience—Im Bereich des Nutzerkomforts wurden ebenfalls RFID-Tags zur Ortung von Objekten verwendet (Find My Stuff). Des Weiteren wurden Kameras in einer Küchenumgebung genutzt, um spezifische Küchenaktivitäten und Objekte zu erkennen (Fine-Grained Kitchen Activity Recognition using RGB-D) sowie Gassensoren, um den Zubereitungsgrad von Speisen zu erkennen (Detecting Cooking State with Gas Sensors During Dry Cooking). Zusätzlich wurde eine 3D-Lokalisierung von Geräuschen mittels im Raum verteilter Mikrofone vorgestellt (Audio Location).

Energy and Lighting—Im Bereich Energie und Beleuchtung wurden zunächst Verfahren vorgestellt, um den Energieverbrauch zu messen (VirdiScope/ GasSense/ AgentSwitch). Darüber hinaus Verfahren, um Energie optimal zu nutzen. So wurde die Helligkeit gemessen und auf Basis dieser Daten die Verdunkelung und Beleuchtung des Smart

Space gesteuert (SunCast) sowie die Beleuchtung in Abhängigkeit von der Nutzerposition gesteuert (Treasure Hunt with Intelligent Luminaires).

HVAC—Für Szenarien in der Kategorie Heizung, Belüftung und Klimatisierung (Heating, Ventilation und AirCondition) ist das Ziel, eine möglichst hohe Energieeffizienz in Verbindung mit hohem Nutzerkomfort zu erreichen (Sensor-based Management of Energy and Thermal Confort). Grundlage der meisten Systeme ist die Betrachtung der Nutzeranwesenheit. Ist kein Nutzer anwesend, können Kühlung und Heizung heruntergefahren werden. Im Gegensatz zu bloßen reaktiven Systemen wurden Systeme mit Vorhersagen über die Nutzeranwesenheit angewendet (GPS-Thermostat/ PreHeat/ SmartThermostat/ TherML), da diese höheres Energiesparpotential bieten und HVAC-Maßnahmen in der Regel eine Verzögerung bis zum Erreichen des Soll-Zustands aufweisen.

Office—Szenarien in Büroumgebungen haben in der Regel das Ziel, die Zusammenarbeit zu verbessern. Dazu wurden verschiedene Verfahren zur Nutzerlokalisierung mittels Bluetooth , WLAN und Kamera (ReticularSpaces/ Learning to Detect User Activity and Avaliability from a Variety of Sensor Data/ Virtual Secretary) angewendet. Zusätzlich wurde Sprachsteuerung verwendet (EasyMeeting).

2.4.2 Szenarien ab 2014

Ausgehend von [3] werden im Folgenden primär Szenarien ab 2014 betrachtet. Ausgangspunkt für die Recherchen sind Aufzeichnungen der im Bereich Ubiquous Computing relevanten Konferenzen UbiComp, CHI und PerCom. Diese wurden auf wissenschaftlichen Arbeiten im Bereich Smart Spaces untersucht.

Um die Vergleichbarkeit fortzuführen, wird die Kategorisierung aus [3] beibehalten.

AAL and Health Care—Gesundheit und unterstütztes Wohnen

A Smart Kitchen for Ambient Assisted Living—In [14] stellen Blasco et al. eine Küchenumgebung vor, die Personen mit Beeinträchtigungen die Nutzung erleichtert. Neben der erleichterten Bedienung der Küchengeräte, werden deren Zustände überwacht, um nützliche Informationen sowie Warnungen zu liefern und in Notsituationen einzugreifen.

Kernstück des Systems ist der *e-servant*, eine zentrale Recheneinheit, mit der sämtliche Sensoren und Aktoren des Systems verbunden sind. Zur Erfassung von Daten dienen eine Reihe von Sensoren zur Überwachung von Gas, Feuer, Rauch, Überflutungen,

Helligkeit und Bewegungen, Türzuständen sowie RFID-Leser. Befehle können zudem auch per Spracheingabe gegeben werden.

Die Ausgabe der Informationen und Warnungen kann über angeschlossene Fernseher, Touchscreens oder Smartphones erfolgen. Zusätzlich kann das System automatisch Notrufe absetzen.

Das System wurde in zwei Living Labs mit realen Nutzern und Betreuern evaluiert und von beiden Gruppen überwiegend positiv bewertet.

Design and Implementation of a Smart Home for the Elderly and Disabled—Das et al. stellen in [15] ein Smart Home System vor, mit dem die Grundlegenden Aktoren im Haushalt gesteuert werden können. Dazu zählen Beleuchtung, Klimatisierung und Verdunkelung.

Das System basiert auf einem Arduino Mega, mit dem die jeweiligen Sensoren verbunden sind. Als nutzbare Sensoren werden Gas-, Ultraschall- und Temperatursensoren sowie RFID-Leser vorgeschlagen. Als Aktoren werden die Klimatisierung, ein Alarm, die Verdunkelung sowie die Beleuchtung angegeben.

Als Beispielszenario wird eine Beleuchtungssteuerung genannt, die abhängig von der Position des Nutzers in der Wohnung die Beleuchtung der einzelnen Räume ein- oder ausschaltet.

Energy and Lighting—Energie und Beleuchtung

Fuzzy logic controller for energy savings in a smart LED lightingsystem considering lighting comfort and daylight—In [16] stellen Liu et al. eine automatisierte Beleuchtungssteuerung vor. Das System überwacht die Anwesenheit eines Nutzers sowie die Helligkeit und regelt auf Basis dieser Werte die Stärke der künstlichen Beleuchtung.

Das System besteht aus Bewegungs- und Helligkeitssensoren, die mit einer zentralen Steuereinheit verbunden sind. In dieser wird anhand der Messwerte mithilfe einer Fuzzy-Logik für jeden überwachten Arbeitsplatz das optimale Verhältnis zwischen natürlicher und künstlicher Beleuchtung berechnet. Daraufhin werden die LEDs separat für jeden Arbeitsplatz angesteuert und eingestellt. Zusätzlich bietet das System die Möglichkeit, die Farbtemperatur den Nutzerwünschen entsprechend anzupassen.

Zur Evaluierung des Systems wurden verschiedene Nutzerszenarien innerhalb einer Testumgebung real implementiert. Die Messungen zeigen teils beträchtliches Einsparpotential (>50%) im Vergleich zur ungeregelten Beleuchtung.

Design and Implementation of Smart Home Control Systems Based on Wireless Sensor Networks and Power Line Communications—Mit [17] stellen Li und Lin ein Smart Home

System vor, das zum Datenaustausch eine Kombination aus kabellosen Verbindungen mitteln ZigBee sowie Datenübertragung mittels Powerline Communication (PLC) verwendet. Mithilfe diese Systems wurde eine intelligente Beleuchtungssteuerung implementiert.

Als Input für die Steuerung dienen Helligkeitssensoren. Die erfassten Daten werden genutzt, um die dimmbaren Lampen sowie die Verdunkelung in Abhängigkeit zur natürlichen Beleuchtung zu steuern. Der zugrunde liegende Algorithmus wird detailliert vorgestellt.

Das System wurde in einer Testumgebung implementiert und evaluiert. Die Einsparungen sind abhängig von den Witterungsverhältnissen (höhere Einsparung an sonnigen Tagen) und bewegen sich zwischen 40% und 80%.

HVAC—Heizung, Belüftung und Klimatisierung

Smart HVAC Control in IoT: Energy Consumption Minimization with User Comfort Constraints—Serra et al. stellen mit [18] eine HVAC-Steuerung vor, die basierend auf Temperatur, Energieverbrauch und Energiepreis arbeitet. Auf das System kann zudem über eine Weboberfläche zugegriffen werden.

Zur Erfassung der Eingangswerte dienen Temperatursensoren, aktuelle Verbrauchswerte der Klimatisierung, aktueller Energiepreis und ein individueller Soll-Temperaturbereich des Nutzers. Während der Berechnung wird aus den aktuellen Werten eine Vorhersage über die zukünftige Temperaturentwicklung getroffen und basierend auf dieser die Leistung des HVAC-Systems gesteuert.

Das System wurde in einem Testraum eingerichtet und beschränkt sich auf ein Heizungssystem. Zur Simulation der Heizung wurden drei Heizlüfter im Raum verteilt, die über steuerbare Steckdosen geregelt werden. Es wurde die Steuerung durch ein vorhersagenbasiertes Verfahren mit der eines einfachen Thermostats verglichen, das bei Unter- oder Überschreiten des Soll-Bereichs einschreitet. Es konnte eine Senkung des Energieverbrauchs durch den vorhersagenbasierten Ansatz gemessen werden.

Design and Implementation of a Cloud Enabled Random Neural Network-Based Decentralized Smart Controller With Intelligent Sensor Nodes for HVAC—In [19] bieten Javed et al. einen Ansatz zur genauen Bestimmung der Personenzahl innerhalb eines geschlossenen Raumes. Dieser basiert auf zwei Annahmen. Zum einen, dass zum Betreten und Verlassen des Raumes Bewegung vor der Tür messbar ist und diese geöffnet und geschlossen wird. Zum anderen, dass sich die CO2 Konzentration des Raumes in Abhängigkeit der Anzahl der anwesenden Personen ändert. Abhängig von der Belegung des Raumes wird die Klimaanlage gesteuert.

Für das System werden Reedschalter an der Zugangstür installiert. Zudem wurde ein Bewegungssensor vor der Innenseite der Tür angebracht. Ein Öffnen der Tür mit darauffolgender Bewegung wird als eintretende Person interpretiert, eine Bewegung mit folgender Türaktion als verlassende Person. Probleme treten auf, wenn mehrere Personen gleichzeitig eintreten oder den Raum verlassen. Zur Verbesserung der Genauigkeit werden diese Werte mit den Messungen des CO2 Sensors sowie einer Vorhersage auf Basis vergangener Aktionen verbunden.

Das System wurde in einer Testumgebung installiert. Die Kombination der Messwerte ermöglicht eine zu 88% korrekte Aussage, ob der Raum belegt ist. Im Vergleich zu einer thermostatbasierten Steuerung ließen sich Einsparungen von 27% messen.

Indoor Air Quality Monitoring System for Smart Buildings—Chen et al. stellen in [20] ein System zur Überwachung der Klimatisierung auf Basis der Luftqualität vor. Die Luftqualität wird anhand der in der Luft befindlichen Feinstaubpartikel bewertet.

Zur Erfassung der Luftqualität werden Feinstaubmessgeräte eingesetzt.

Im Rahmen der Evaluation wurde das System in vier Bürogebäuden eingesetzt. Neben der reinen Überwachung der Luftqualität wurde, in Abhängigkeit der Feinstaubdichte außerhalb der Gebäude sowie Temperatur, Windgeschwindigkeit, Luftfeuchtigkeit und Luftdruck, die Zeit zur Luftreinigung festgestellt. Aus diesen Werten kann die Effektivität der HVAC-Anlage bestimmt und Anomalien erkannt werden.

AirSense: An Intelligent Home-based Sensing System for Indoor Air Quality Analytics—Mit [21] stellen Fang et al. ein System zur Überwachung der Luftqualität vor. Das System bietet eine Smartphone App zur Visualisierung der Messwerte und gibt Hinweise, um die Luftverschmutzung zu reduzieren. Zusätzlich wird versucht, den zugrunde liegenden Auslöser für erhöhte Messwerte zu bestimmen.

Zur Messung der Luftverschmutzung werden ein Feinstaubmessgerät und ein Messgerät für flüchtige organische Verbindungen (Volatile organic compounds/VOC) eingesetzt. Es werden zusätzlich Temperatur und Luftfeuchtigkeit überwacht.

Zur Evaluierung wurde das System in zwei Stufen in insgesamt fünf realen Haushalten eingesetzt. Es konnte mit hoher Sicherheit (durchschnittlich 99%) das zugrunde liegende Ereignis den Kategorien Kochen, Rauchen und Pestizideinsatz zugeordnet werden.

Convenience—Nutzerkomfort

CapCouch: Home Control With a Posture-Sensing Couch—Mit [22] stellen Pohl et al. eine Möglichkeit vor, Positionsänderungen auf einem Sofa als Eingabewerte für die

Steuerung der in einem Smart Space befindlichen Aktoren zu verwenden.

Bei der Umsetzung wurde ein Sofa mit 6 Drucksensoren ausgestattet. Ausgehend von den betätigten Sensoren wurden 9 Zustände definiert, denen Aktionen zugeordnet werden können.

Das System wurde in einer Smart Home Umgebung eingesetzt, um Beleuchtung und Unterhaltungselektronik zu steuern. Die Sitzposition des Nutzers wurde mit hoher Wahrscheinlichkeit (>90%) erkannt.

Building Change: Constructive Design of Smart Domestic Environments for Goal Achievement—Mit [23] stellen Brotman et al. einen Ansatz vor, der Nutzern beim Erreichen selbst gesetzter Ziele unterstützen soll. Ziel ist es, Sensoren und Aktoren des Smart Spaces zur Motivation des Nutzers einzusetzen.

Eins von drei gegebenen Beispielen behandelt das Lernen eines Musikstücks. Zu diesem Zweck wird unter anderem das zu lernende Stück als Weckton wiedergegeben oder bei der Heimkehr die Beleuchtung gezielt auf die Gitarre gelenkt.

Bei den stark abweichenden Szenarien weichen einhergehend auch die verwendeten Aktoren und Sensoren ab. Wichtiger als die einzelnen Szenarien ist jedoch die Herangehensweise an die Konstruktion von Szenarien, nicht nur bloße Unterstützung, sondern gezielte Motivation des Nutzers in den Vordergrund zu stellen.

Integrating the Smart Home into the Digital Calendar—Mit Casalendar [24] bieten Mennicken et al. einen digitalen zentralen Kalender, in dem jedes Familienmitglied Termine eintragen kann. Zusätzlich können Ereignisse im Smart Home in diesem Kalender angezeigt werden.

Durch den Einsatz in zwei Haushalten konnte festgestellt werden, dass durch die Übersicht im Kalender Verhaltensweisen der Anwohner sowie des Smart Homes identifiziert und optimiert werden können.

2.4.3 Zusammenfassung

Die Szenarien ab 2014 werden in Tabelle 2.4 noch einmal zusammengefasst. Neben Titel, Autor und Veröffentlichungsjahr werden explizit die in den jeweiligen Arbeiten identifizierten Smart Space Szenarien genannt. Aus diesen werden im Design die relevanten Szenarien identifiziert. Zusätzlich werden die benötigten Sensoren und Aktoren aufgelistet, die für die Umsetzung benötigt werden. Schließlich wird die von den Autoren genutzte Evaluationsmethode und deren Ergebnisse aufgelistet. Aus dieser lassen sich die Performazindikatoren extrahieren.

Tabelle 2.4: Szenarien ab 2014

Titel	Autor	Jahr	Szenario	Sensoren	Aktoren	Evaluation	Ergebnisse
A Smart Kitchen for Ambient Assisted Living [14]	Blasco et al.	2014	Vernetzung von Haushalt und Geräten	Feuer, Rauch, Gas, Überflutungen, Helligkeit, Bewegung, Reedschalter, RFID-Leser, Mikrofon	Bildschirm, Telefon	Living Lab, 63 Nutzer 31 Betreuer	Bewertung überwiegend positiv
Design and Implementation of a Smart Home for the Elderly and Disabled [15]	Das et al.	2015	Implementierung einer Smart Home Umgebung	Gas, Ultraschall, Temperatur, RFID-Leser	Beleuchtung, Verdunkelung, Klima	Prototyp in Modellhaus	—
Fuzzy logic controller for energy savings in a smart LED lightingsystem considering lighting comfort and daylight [16]	Liu et al.	2016	Beleuchtungssteuerung auf Basis von Helligkeit und Anwesenheit	Helligkeit, Bewegung	RGB-Led	Reale Implementierung in Testumgebung	Gegenüber ungeregelter Beleuchtung Energieeinsparung >50% möglich
Design and Implementation of Smart Home Control Systems Based on Wireless Sensor Networks and Power Line Communications [17]	Li, Lin	2015	Implementierung einer Beleuchtungssteuerung auf Basis von Helligkeit	Helligkeit	Dimmbare Beleuchtung	Reale Implementierung in Testumgebung	Energieeinsparung zwischen 40% und 80%

Tabelle 2.4: Szenarien ab 2014

Titel	Autor	Jahr	Szenario	Sensoren	Aktoren	Evaluation	Ergebnisse
Smart HVAC Control in IoT: Energy Consumption Minimization with User Comfort Constraints [18]	Serra et al.	2014	HVAC-Steuerung, basierend auf Temperatur, Energieverbrauch und Energiepreis	Temperatur	HVAC	Reale Implementierung in Testumgebung, beschränkt auf Heizung	Senkung des Energieverbrauchs
Design and Implementation of a Cloud Enabled Random Neural Network-Based Decentralized Smart Controller With Intelligent Sensor Nodes for HVAC [19]	Javed et al.	2017	Bestimmung der Personenzahl innerhalb eines geschlossenen Raumes zur Steuerung der Klimatisierung	Bewegung, Reedschalter, CO2	Klima	Reale Implementierung in Testumgebung	Belegung zu 88% korrekt, Energieeinsparung von 27%
Indoor Air Quality Monitoring System for Smart Buildings [20]	Chen et al.	2014	Überwachung der Klimatisierung auf Basis der Luftqualität	Feinstaubmessgerät	—	Realer Einsatz in 4 Bürogebäuden	Effektivität der Klimatisierung kann gemessen und optimiert werden

Tabelle 2.4: Szenarien ab 2014

Titel	Autor	Jahr	Szenario	Sensoren	Aktoren	Evaluation	Ergebnisse
AirSense: An Intelligent Home-based Sensing System for Indoor Air Quality Analytics [21]	Fang et al.	2016	Überwachung der Luftqualität	Feinstaubmessgerät, VOC-Sensor, Temperatur, Luftfeuchtigkeit	—	Realer Einsatz in insgesamt 5 Haushalten	Im Durchschnitt konnten Quellen für die Luftverschmutzung zu 99% der korrekten Kategorie zugeordnet werden
CapCouch: Home Control With a Posture-Sensing Couch [22]	Pohl et al.	2015	Positionsänderung auf Sofa zur Steuerung von Smart Space Aktoren	Drucksensoren	Beleuchtung, Unterhaltungselektronik	Realer Einsatz in Testumgebung	Korrekte Erkennung der Sitzposition mit über 90%
Building Change: Constructive Design of Smart Domestic Environments for Goal Achievement [23]	Brotman et al.	2015	Nutzer beim Erreichen von Zielen unterstützen	z.B.:Bewegung, Reedschalter	z.B.: Beleuchtung, Lautsprecher	Tests in 3 Szenarien	Smart Home zur Nutzermotivation nutzbar
Integrating the Smart Home into the Digital Calenda [24]	Mennicken et al.	2016	Zentraler Kalender mit Smart Home Ereignissen	—	—	Realer Einsatz in 2 Haushalten	Verhaltensweisen von Nutzern und Smart Home können optimiert werden

2.4.4 Relevante Szenarien

In diesem Abschnitt werden die für die Implementierung ausgewählten Szenarien vorgestellt.

Bei den Recherchen im Bereich der relevanten Szenarien sind die Gewählten wiederholt aufgetreten oder beinhalten relevante Aufgaben, die mehrfach umgesetzt wurden. Tabelle 2.5 enthält die vier Szenarien, die bei Untersuchung der vorhergehenden Arbeiten am häufigsten vorkamen.

Tabelle 2.5: Relevante Szenarien

Szenario	Vorkommen in vorhergehenden Arbeiten
Beleuchtung	[16], [17], [25]
Heizung	[18], [26], [27], [28], [29]
Luftqualität	[20], [21]
Identifikation	[30], [31], [32]

Für jedes Szenario wird im Folgenden die Funktionsweise mit explizitem Algorithmus sowie die verwendeten Sensoren und Aktoren beschrieben.

2.4.4.1 Szenario 1 - Beleuchtung

Aufbau/Funktionalität—In diesem Szenario wird eine automatische Beleuchtungssteuerung implementiert. Es wird laufend die Helligkeit innerhalb des Raumes gemessen, um in Abhängigkeit von diesem Parameter eine optimale Ausleuchtung durch Kombination von natürlicher und künstlicher Beleuchtung zu erzielen. Um eine hohe Energieeffizienz zu erreichen, soll primär natürliche Beleuchtung in Form von Sonnenlicht genutzt werden. Um die Intensität der natürlichen Beleuchtung zu regeln, wird die Verdunkelung in Form einer steuerbaren Jalousie genutzt. Als künstliche Beleuchtung dienen mehrere steuerbare Lampen innerhalb des Raumes. Um die Energieeffizienz weiter zu steigern, wird die Beleuchtung zusätzlich in Abhängigkeit von der Anwesenheit des Nutzers gesteuert, sodass in Abwesenheit des Nutzers die künstliche Beleuchtung heruntergefahren wird.

In Abbildung 2.1 ist der Algorithmus der Beleuchtungssteuerung schematisch dargestellt. Als Input dienen Helligkeits- und Bewegungssensor. Für die Helligkeit wird geprüft, ob sich diese innerhalb des Soll-Bereichs befindet. Bei Abweichung unter den minimalen Wert wird zunächst geprüft, ob die Verdunkelung geschlossen ist. Falls ja, wird diese schrittweise geöffnet, falls nein, wird begonnen, die Intensität der künstlichen Beleuchtung zu erhöhen. Weicht der Helligkeitswert über den als Maximum definierten Wert ab, wird zunächst geprüft, ob die künstliche Beleuchtung eingeschaltet ist. In diesem Fall wird die Beleuchtung heruntergeregelt. Sollte die Beleuchtung bereits ausgeschaltet sein, wird begonnen, die Verdunkelung schrittweise zu schließen. Parallel zu diesen Auf-

gaben wird ständig die Belegung des Raumes überwacht. Wurde innerhalb der letzten fünf Minuten keine Belegung festgestellt, wird die Beleuchtung ausgeschaltet.

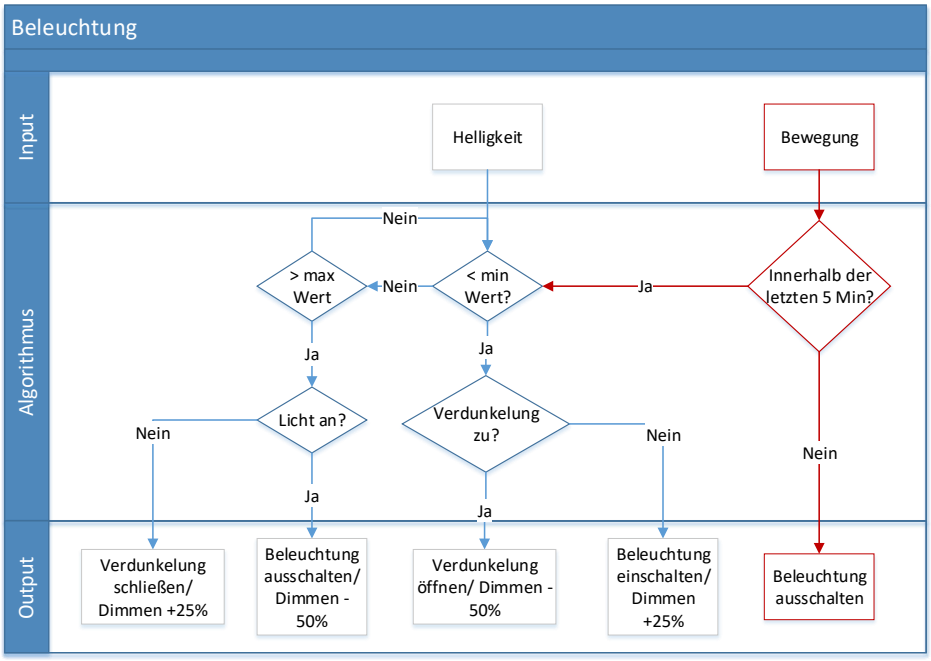


Abbildung 2.1: Beleuchtung Algorithmus

Auswahlgrund—Die Auswahl dieses Szenarios ist durch mehrfaches Vorkommen innerhalb der relevanten Szenarien begründet. So beinhalten [16] und [17] aus Abschnitt 2.4.2 eine Beleuchtungssteuerung mit Anwesenheitserkennung und Helligkeitserkennung. Bei SunCast [25] (2.4.1 Energy and Lighting) wird zusätzlich die Zufuhr natürlicher Beleuchtung über eine Verdunkelung reguliert.

Benötigte Entitäten—Um das Beleuchtungsszenario umzusetzen werden die folgenden Entitäten im Smart Space benötigt:

Sensoren:

- Helligkeit - Umgebungshelligkeit in Lux
- Bewegung - Binärwert, ob Bewegung erkannt wurde

Aktoren:

- Beleuchtung - Prozentwert, auf den die Intensität der Beleuchtung gedimmt wird
- Verdunkelung - Prozentwert, zu dem die Jalousie geschlossen wird

2.4.4.2 Szenario 2 - Heizung

Aufbau/Funktionalität—Das Szenario Heizung umfasst die automatische Regelung der Raumtemperatur. Als Grundlage für die Steuerung dient der aktuelle Temperaturwert. Zur Steigerung der Energieeffizienz wird zusätzlich die Anwesenheit des Nutzers überwacht, da durch die Erhaltung der Wunschtemperatur bei Abwesenheit Energie verschwendet wird. Zur Beeinflussung der Temperatur werden ein Thermostat zur Steuerung der Heizung, sowie ein Lüfter zur Steuerung der Kühlung verwendet.

Der Algorithmus für dieses Szenario ist in Abbildung 2.2 dargestellt. Als Eingabewerte dienen Raumtemperatur und Anwesenheit. Für die Temperatur wird stetig geprüft, ob sich diese innerhalb des Soll-Bereichs befindet. Bei Überschreitung des Maximalwerts wird zunächst überprüft, ob das Thermostat geöffnet ist. Falls ja, wird es geschlossen. Sollte das Thermostat bereits geschlossen sein, wird der Ventilator zur Kühlung eingeschaltet. Bei Unterschreiten des Minimums wird hingegen zunächst geprüft, ob der Ventilator eingeschaltet ist. Falls ja, wird dieser ausgeschaltet, falls nein, wird das Thermostat geöffnet. Die Verringerung der Heiz- und Kühlleistung erfolgt in mehreren Abstufungen. Dies ist erforderlich, da die Anpassung der Raumtemperatur mit einer Verzögerung behaftet ist und bei kürzerer Abwesenheit die Wunschtemperatur schnell wieder erreicht werden soll. Bei einer Abwesenheit von mehr als fünf Minuten wird die Lüftung ausgeschaltet und die Heizleistung um 25% reduziert. Bei einer Abwesenheit zwischen zehn und dreißig Minuten wird die Heizleistung um 50% reduziert und danach auf ein Minimum heruntergefahren.

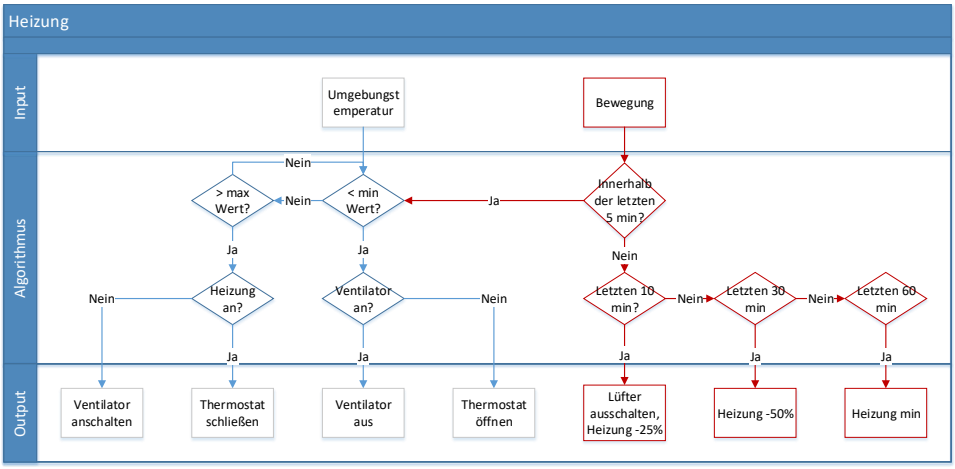


Abbildung 2.2: Heizung Algorithmus

Auswahlgrund—Grund für die Auswahl ist das mehrfache vorkommen intelligenter Heizungssteuerungen in den relevanten HVAC Szenarien 2.4.2 *Smart HVAC Control*

in IoT: Energy Consumption Minimization with User Comfort Constraints [18] und 2.4.1 *GPS-Thermostat [26]/ PreHeat [27]/ SmartThermostat [28]/ TherML [29]*

Benötigte Entitäten—Die für das Heizungsszenario benötigten Entitäten im Smart Space sind:

Sensoren:

- Temperatur - Umgebungstemperatur in Grad Celsius
- Bewegung - Binärwert, ob Bewegung erkannt wurde

Aktoren:

- Thermostat - Prozentwert, zu dem das Thermostat geöffnet wird
- Lüfter - Binärwert zum An- und Ausschalten

2.4.4.3 Szenario 3 - Luftqualität

Aufbau/Funktionalität—Dieses Szenario umfasst die Überwachung der Luftqualität. Als Qualitätskriterien wurden der Anteil von Kohlendioxid sowie Wasserdampf in der Raumluft ausgewählt. Kohlendioxid hat maßgeblichen Einfluss auf die Konzentrationsfähigkeit des Nutzers. Der Anteil von Wasserdampf hat ebenfalls Einfluss auf das Wohlbefinden des Nutzers und kann bei zu hohen Werten negative Auswirkungen auf elektrische Geräte haben.

Der verwendete Algorithmus ist schematisch in Abbildung 2.3 dargestellt. Die Werte für den H2O und CO2 Gehalt werden stetig überprüft. Bei Überschreiten der Grenzwerte wird die Lüftung eingeschaltet und ein Alarm mit der entsprechenden Nachricht ausgegeben.

Auswahlgrund—Das Szenario wurde gewählt, da durch die in 2.4.2 vorgestellten Arbeiten *Indoor Air Quality Monitoring System for Smart Buildings* [20] und *AirSense: An Intelligent Home-based Sensing System for Indoor Air Quality Analytics* [21] ebenfalls Systeme zur Überwachung der Luftqualität umgesetzt wurden.

Benötigte Entitäten—Die benötigten Smart Space Entitäten in diesem Szenario sind:

Sensoren:

- CO2 - CO2 Konzentration in ppm
- H2O - Luftfeuchtigkeit in Prozent

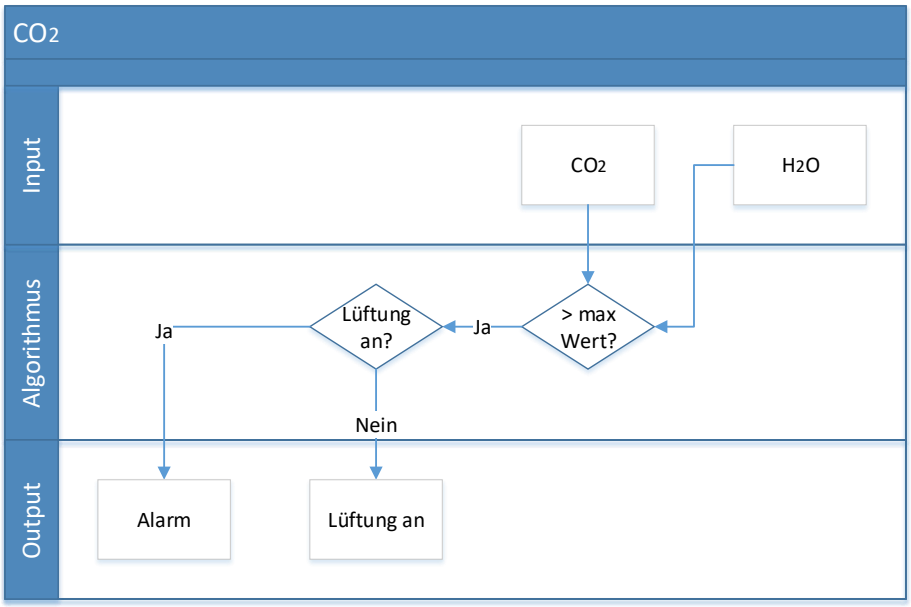


Abbildung 2.3: Luftqualität Algorithmus

Aktoren:

- Lüfter - Binärwert zum An- und Ausschalten
- Alarm - Binärwert, ob Alarm gesetzt, Text mit Auslösegrund

2.4.4.4 Szenario 4 - Identifikation

Aufbau/Funktionalität—Ziel dieses Szenarios ist es, Personen bei Betreten eines Raumes zu identifizieren. Zu diesem Zweck wird über einen im Türrahmen angebrachten Ultraschallsensor die Größe der eintretenden Person gemessen. Dieser Wert wird daraufhin mit den Daten einer zuvor angelegten Liste verglichen, um eine Identifikation zu ermöglichen. Nach erfolgreicher Identifikation wird eine benutzerspezifische Aktion ausgeführt. Zu diesem Zweck wird in der Liste zusätzlich eine Farbe zur jeweiligen Person gespeichert, in welcher nach Eintreten eine RGB-LED leuchten soll. Zur Feststellung der Bewegungsrichtung wird ein Bewegungssensor auf der Innenseite der Tür angebracht.

Der Algorithmus ist in Abbildung 2.5 eingezeichnet. Beim Öffnen der Tür wird ein Reedschalter ausgelöst, woraufhin die Höhenmessung des Ultraschallsensors gestartet wird. Die Messung wird fortgesetzt, bis die Tür geschlossen wird. Sobald ein Nutzer die Tür passiert, verringert sich die gemessene Entfernung. Es wird der minimal gemessene Wert

gespeichert, da er der maximalen Höhe des Nutzers entspricht (Abb.2.4). Nach dem Schließen der Tür wird anhand dieser Entfernung der Nutzer aus der Liste ausgewählt. Gleichzeitig wird bei Auslösen des Reedschalters der aktuelle Wert des Bewegungssensors abgerufen. Bei Bewegung, wird ein Verlassen gezählt, bei dem die Beleuchtung des Raums deaktiviert wird, ansonsten ein Betreten, bei dem das Licht aktiviert wird.

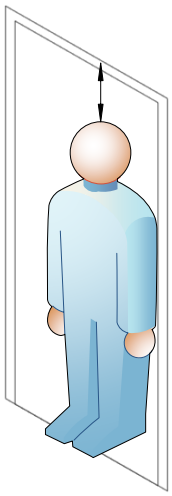


Abbildung 2.4: Ultraschall Messung

Auswahlgrund—Das Szenario wurde ausgewählt, da Nutzeridentifikation ein häufiger Bestandteil relevanter Szenarien ist. Oft wird diese über RFID-TAGs realisiert (2.4.1 Gator tech smart house [30] , Innovative Retail Laboratory [31]). Das interessante an einem Szenario mit Ultraschall Identifikation wie in 2.4.1 *Using Height Sensors for Biometric Identification in Multi-resident Homes* [32] ist, dass kein extra Device (z.B. RFID-Tag) benötigt wird.

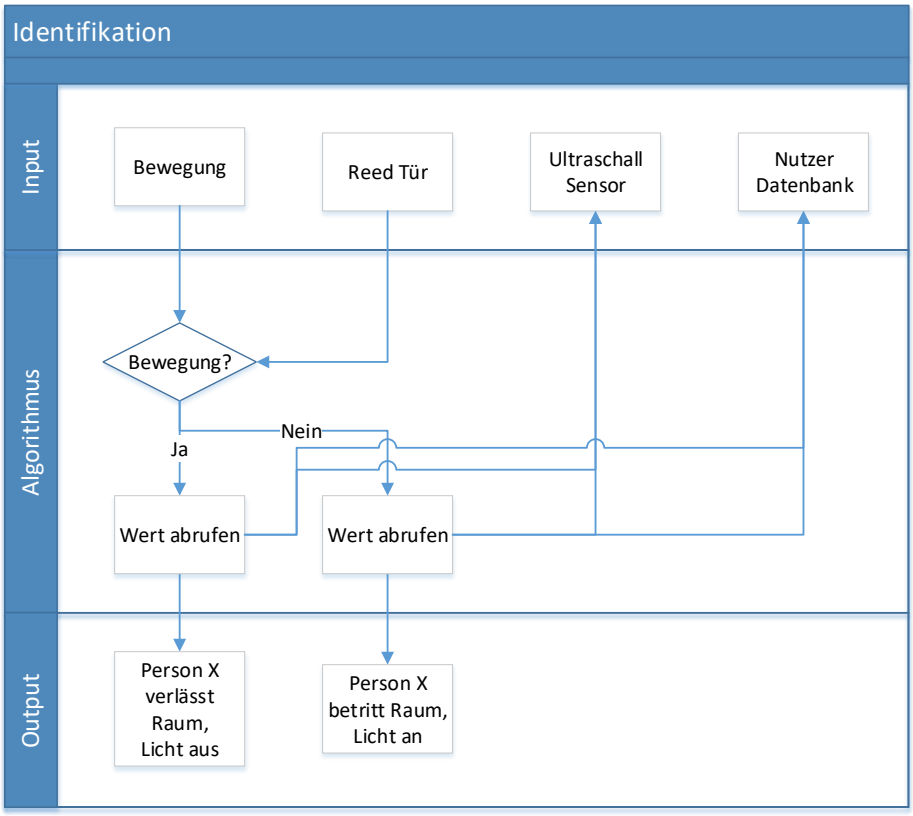


Abbildung 2.5: Identifikation Algorithmus

Benötigte Entitäten—Für das Identifikationsszenario werden folgende Smart Space Entitäten benötigt:

Sensoren:

- Ultraschall Sensor - Entfernung zum nächsten Objekt in Zentimetern
- Reedschalter - Binärwert, ob Tür geöffnet oder geschlossen ist

Aktoren:

- RGB-LED - Farbwert für aktuellen Nutzer

2.4.4.5 Zusammenfassung

In Tabelle 2.6 wurden die gewählten Szenarien mit den jeweils benötigten Sensoren und Aktoren zusammengefasst.

Tabelle 2.6: Zusammenfassung gewählter Szenarien

Szenario	Funktion	Sensoren	Aktoren
Beleuchtung	Beleuchtungssteuerung mit Helligkeits- und Anwesenheitsmessung	Bewegung, Helligkeit	Beleuchtung, Verdunkelung
Heizung	Heizungssteuerung mit Temperatur- und Anwesenheitserkennung	Bewegung, Temperatur	Lüfter, Thermostat
Luftqualität	Überwachung der CO2 und H2O Konzentration	CO2, H2O	Alarm, Lüfter
Identifikation	Identifikation eintretender Personen mittels Ultraschall	Ultraschall, Reedschalter	RGB-LED

2.4.5 Benötigte Sensoren

Temperatur

Die Messung der Raumtemperatur durch wärmeempfindliche Bauteile ist ein wichtiges Kriterium für den Raumzustand. Die Raumtemperatur hat große Auswirkung auf die Arbeitsfähigkeit des Nutzers und sollte immer innerhalb eines engen Bereichs bleiben. Innerhalb der Smart Space Steuerung ist dieser Wert wichtig für die Regelung der Aktoren Thermostat und Belüftung im Szenario Heizung.

Hygrometer

Die Messwerte eines Hygrometers geben Aufschluss über den Wasserdampfgehalt der Luft, beziehungsweise der Luftfeuchtigkeit. Dieser hat ebenfalls große Auswirkungen auf das Wohlbefinden des Nutzers. Zusätzlich können auch Geräte innerhalb des Raumes durch zu hohe Luftfeuchtigkeit in Mitleidenschaft gezogen werden. Aktoren, für deren Steuerung dieser Messwert von Bedeutung ist, sind Heizung, Lüftung und Kühlung, da die Luftfeuchtigkeit ebenfalls stark mit der Raumtemperatur variiert.

CO2 Sensor

Der CO2 Gehalt der Luft ist ein entscheidender Parameter für die Produktivität des Nutzers. Steigende CO2 Konzentrationen führen beim Menschen zu Müdigkeit und Konzentrationsschwierigkeiten. Da der CO2 Gehalt der Luft in der Regel langsam steigt, treten die Auswirkungen ebenfalls nur schleichend ein und werden vom Nutzer zumeist erst spät bemerkt. Der CO2 Gehalt ist eine wichtige Grundlage für die Belüftungssteuerung.

Helligkeitssensor

Die Messung der Helligkeit ist ein wichtiges Kriterium, um im Zusammenspiel zwischen natürlicher und künstlicher Beleuchtung eine ausgewogene Ausleuchtung des Raumes gewährleisten zu können. Zu helle Räume können der Konzentrationsfähigkeit des Nutzers schaden und verursachen vermeidbare Energiekosten. Zu dunkle Räume erschweren die Arbeit und fördern Müdigkeit. Die Helligkeit ist ein wichtiger Eingabewert für die Beleuchtungssteuerung sowie die Verdunkelung mittels Jalousien. Viele Wearables besitzen ebenfalls einen Sensor zur Bestimmung der Helligkeit, sodass die Messung der Helligkeit nicht zwingend stationär erfolgen muss.

Reedschalter

Mittels Reedschaltern, die an Türen oder Fenstern angebracht sind, kann festgestellt werden, ob diese geöffnet oder geschlossen sind. Dazu erfasst der Reedschalter das Magnetfeld eines am zu überwachenden Objekts angebrachten Magneten. Diese Information ist wichtig für Heizung Belüftung und Kühlung. Zusätzlich können auch Alarme bei illegalen Systemzuständen gegeben werden. Beispiele wären ein Verlassen des Raums bei geöffnetem Fenster sowie ein Öffnen des Fensters außerhalb der Nutzungszeiten.

Bewegungssensor / PIR

Ein Bewegungssensor, auch Bewegungsmelder oder Pyroelectric Infrared Sensor (PIR) erfasst Bewegungen auf Basis von Temperaturänderungen im Messbereich. Diese Sensoren werden in der Regel zur Anwesenheitsbestimmung eines Nutzers verwendet. In den Szenarien Beleuchtung und Heizung wird er genutzt um die Abwesenheit eines Nutzers festzustellen und daraufhin Aktoren wie Lampen oder die Belüftung zu deaktivieren.

Ultraschallsensor

Mithilfe eines Ultraschallsensors können Entfernungen bestimmt werden. Dazu werden vom Sensor Schallwellen ausgesendet, die sich mit konstanter Geschwindigkeit ausbreiten. Es wird die Zeit gemessen, bis das Echo wieder beim Sensor ankommt. Aus der Laufzeit der Schallwellen kann die Entfernung zum reflektierenden Objekt berechnet werden.

2.4.6 Benötigte Aktoren

Die Aktoren umfassen sämtliche Instanzen innerhalb des Smart Spaces, die Eingaben erwarten und Aktionen ausführen können.

Im Folgenden wird eine Auflistung möglicher Aktoren innerhalb des Smart Spaces gegeben.

Beleuchtung

Die Steuerung der Beleuchtung umfasst die Regelung des künstlichen Lichts innerhalb des Smart Spaces. Parameter für die Beleuchtung sind die Helligkeit sowie bei bestimmten Lampen ebenfalls die Farbe. Die Farbe der Beleuchtung hat nachweisliche Auswirkungen auf den Zustand des Nutzers. Bläuliches (kaltes) Licht beispielsweise wird genutzt, um Müdigkeitserscheinungen zu verringern.

Verdunkelung

Die Verdunkelung umfasst die Steuerung der Jalousien. Diese können für eine Änderung der Helligkeit sowie der Temperatur durch Erhöhung des Sonnenschutzes genutzt werden.

HVAC

HVAC kurz für Heating, Ventilation und AirCondition. Dies umfasst die Steuerung der Raumtemperatur, in der Regel über die Heizung und einer eventuell vorhandenen Klimaanlage. Darüber hinaus die Steuerung der Belüftung, der Zirkulation der Raumluft und der Frischluftzufuhr.

Alarm

Der Alarm kann durch Erkennung illegaler Zustände im Smart Space ausgelöst werden. Mögliche Umsetzungen eines Alarms können ein akustisches Signal über Lautsprecher oder Summer, ein visuelles Signal, wie ein Alarmlicht, oder eine Textnachricht an den Verantwortlichen sein. Zusätzlich sind auch Aktoren innerhalb von Wearables, wie beispielsweise ein Vibrationsmodul, nutzbar. Illegale Systemzustände können ungewollte Aktionen wie z.B. Fenster geöffnet und Heizung an oder verlassen des Raums bei geöffnetem Fenster sein. Ein weiterer illegaler Zustand wären nicht autorisierte Personen im Smart Space.

2.4.7 Zusammenfassung

In diesem Abschnitt wurden zunächst wissenschaftliche Arbeiten im Bereich der Smart Spaces analysiert. Anhand dieser wurden relevante Themenbereiche identifiziert, aus denen die Szenarien für spätere Implementation gebildet wurden.

An diesem Punkt wurden die zu betrachtenden Szenarien ausgewählt und somit die Grundlage zur Beantwortung der Leitfrage gelegt. Es steht nun fest, mithilfe welchen Szenarien der Wearablenutzen evaluiert wird.

- Relevante Szenarien identifizieren ✓

2.5 Wearables in Smart Space Szenarien

In diesem Abschnitt wird der Wearableeinsatz in Smart Space Szenarien analysiert. Dazu wird untersucht, wie sich die Sensoren aus 2.3.2 in die gewählten Szenarien aus 2.4.4 integrieren lassen. Die Szenarien werden zunächst auf Schwächen hin untersucht. Daraufhin wird unter Zuhilfenahme der in 2.3.3 erstellten Sensorenliste geprüft, ob die Situation durch den Einsatz tragbarer Sensoren verbessert werden kann. Für jeden Sensor wird dessen mögliche Nutzbarkeit betrachtet. Bei dieser Analyse entstehen die für die Evaluation der Leitfrage benötigten erweiterten Smart Space Szenarien. Schließlich werden die zu implementierenden erweiterten Szenarien ausgewählt.

Es werden die folgenden Meilensteine erreicht:

- Erweiterte Wearable Szenarien erstellen
- Hypothesen über Wearablenutzen aufstellen
- Festlegen von Performance Indikatoren
- Auswahl zu implementierender Szenarien

Bei Betrachtung der Szenarien aus 2.4.3 fällt auf, dass nahezu alle Szenarien ausschließlich stationäre Sensoren einsetzen. So werden in der Kategorie AAL (Ambient Assisted Living) and Health Care oftmals Bewegungssensoren und Kontaktschalter zur Überwachung der Patienten eingesetzt. In den Kategorien Convenience, die sich mit der Erhöhung des Nutzerkomforts innerhalb eines Smart Spaces befasst, und Office werden zusätzlich Mikrofone, Kameras und RFID Systeme eingesetzt. In den Kategorien Energy and Lighting sowie HVAC werden Umgebungsparameter wie Helligkeit und Temperatur erfasst. [3] Diese Messwerte sollen nun um die Daten mobiler Sensoren erweitert werden.

2.5.1 Erweiterung relevanter Szenarien

2.5.1.1 Szenario 1 - Beleuchtung

Das Szenario Beleuchtung aus 2.4.4.1 umfasst eine automatische Beleuchtungssteuerung auf Basis der Raumhelligkeit und Anwesenheit eines Nutzers. Beide Parameter werden durch stationäre Sensoren gemessen.

Schwächen

Platzierung des Helligkeitssensors: Durch die Messung der Helligkeit an einem fixen Punkt im Raum wird für diesen Punkt die optimale Helligkeit erreicht. Die Wahl dieses Punktes ist demnach für die Genauigkeit der Steuerung von großer Bedeutung. Zusätzlich erschwert wird die Wahl dieses Punktes, falls der Nutzer verschiedene Positionen innerhalb des Raumes einnehmen kann (Abb. 2.6: A oder B). Die Verwendung mehrerer Sensoren bringt in diesem Punkt keine Besserung, da Messwerte in Konflikt zueinander stehen können und ohne Lokalisierung des Nutzers keine Aussage möglich ist, welchem Sensor höhere Priorität eingeräumt werden muss. Auch das Aufstellen mehrerer Lampen in jedem Arbeitsbereich mit jeweils dediziertem Sensor ist nicht optimal, da ohne Lokalisierung ebenfalls nicht festgestellt werden kann, welche Lampen genutzt und welche ausgeschaltet werden können, um Energie zu sparen.

Um den optimalen Punkt für die Messung der Helligkeit festzulegen, wird zunächst das Ziel der Beleuchtung betrachtet. Sie soll es dem Nutzer ermöglichen, in einem

optimal ausgeleuchteten Arbeitsbereich zu arbeiten. Dieser Arbeitsbereich befindet sich in einem Büroszenario in der Regel auf dem Tisch vor dem Nutzer. Es sollen also Tastatur oder Dokumente so ausgeleuchtet werden, dass sie für den Nutzer gut lesbar sind. Aus diesem Grund ist es sinnvoll, die Helligkeit in genau diesem Bereich zu messen. Die Positionierung eines stationären Sensors in diesem Bereich erweist sich als schwierig. Eine Tastatur, die für viele Aufgaben eingesetzt wird, könnte mit einem Sensor versehen werden. Schwieriger wird die Positionierung, wenn verschiedene Dokumente gelesen werden. Eine Positionierung im Tisch ist nicht möglich, da dieser verdeckt würde, und die Befestigung an den Dokumenten ist aufwändig, da ein Nutzer jedes Mal den Sensor befestigen müsste, was aus Usability Sicht unpraktikabel ist.

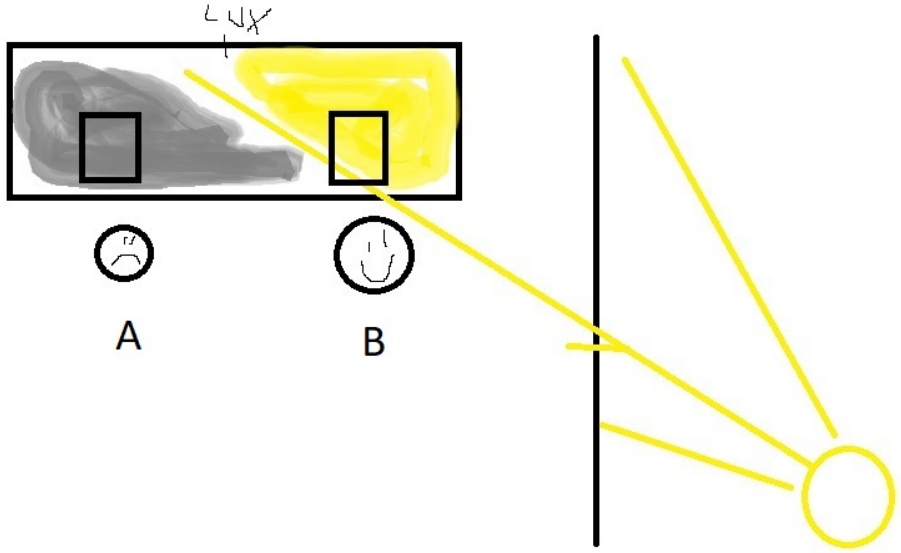


Abbildung 2.6: Feste Position des Helligkeitssensors

Lösung durch Wearablesensoren

Um die Schwäche des Szenarios lösen zu können, muss zunächst untersucht werden, welche Sensoren aus Abschnitt 2.3.3 sich für eine Lösung eignen. Aus der Sensorenliste geht hervor, dass Helligkeitssensoren oftmals Bestandteil der Wearablesensorik sind. Diese können für die Messung im Szenario Beleuchtung verwendet werden.

Die Nutzung des am Handgelenk befindlichen Helligkeitssensors bietet eine Lösung für das Problem der Sensorplatzierung des Beleuchtungsszenarios. Die Helligkeit kann so innerhalb des Arbeitsbereichs unabhängig von der ausgeführten Aktion gemessen werden. Da sich der Sensor nahe des zu beleuchtenden Objekts befindet, ist davon auszugehen, dass eine Optimierung der Beleuchtung erzielt werden kann. Da sich der Sensor mit dem Nutzer bewegt, ist die Lokalisierung des Nutzers nicht mehr nötig. In Abbildung 2.7 „wandert “ der Sensor mit dem Nutzer zwischen A und B.

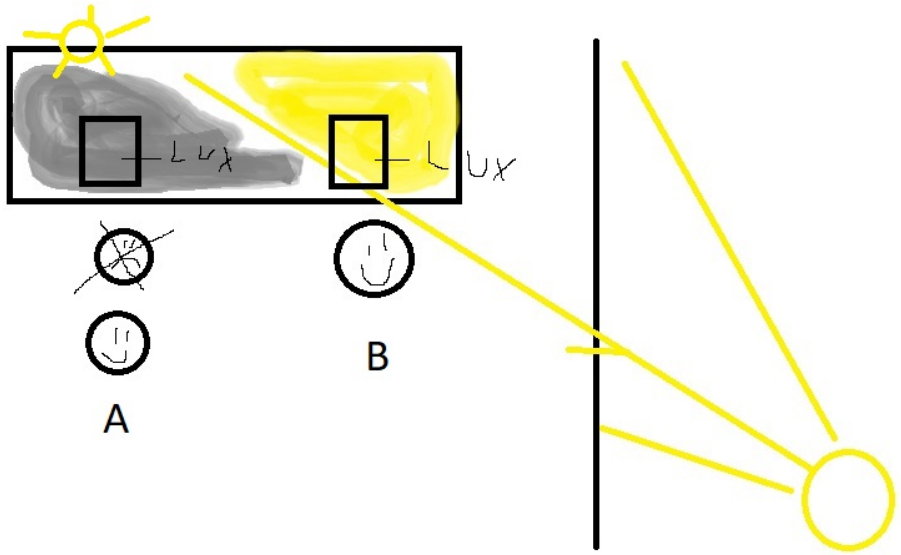


Abbildung 2.7: Bewegliche Position des Helligkeitssensors

Erwarteter Nutzen

Usability: Da es sich zuvor bereits um ein automatisiertes Szenario ohne Nutzerinteraktion gehandelt hat, werden durch das Wearable keine Arbeitsschritte während des Betriebs eingespart. Bei Implementierung des Szenarios im Smart Space muss jedoch kein separater Lux Sensor eingebaut und platziert werden.

Erhöhung der Präzision: Bei der Messung der Helligkeit durch das Wearable ist eine Erhöhung der Präzision der Beleuchtungssteuerung zu erwarten. Es können korrekte Ergebnisse auch bei variierender Position des Nutzers im Raum erzeugt werden. Zudem befindet sich der Helligkeitssensor auf Höhe des Handgelenks in einer akkuraten Messposition, nah an Arbeitsmaterialien, wie zu lesenden Dokumenten oder der Computer Tastatur. Befindet sich der Nutzer in Abbildung 2.7 an Position A, kann die verringerte Helligkeit nun erkannt und z.B. eine Lampe angeschaltet werden.

Key Performance Indikatoren

Die Arbeiten [16] und [17] zeigen, dass durch genauere Eingabewerte bei der Beleuchtungssteuerung Energie eingespart werden kann. Daher ist der Performance Indikator für die Evaluation:

- Helligkeit im Arbeitsbereich - Bestimmung der Abweichung von Wearable und stationärem Sensor von IST-Wert im Arbeitsbereich

Anforderungen

- R1: Implementierung eines Smart Space Szenarios mit Wearable - Nutzung des tragbaren Lux Sensors ✓
- R2: Verbindung zwischen stationären und tragbaren Entitäten - Verbindung zwischen Wearable und stationärer Beleuchtung ✓

2.5.1.2 Szenario 2 - Heizung

Im Szenario Heizung aus 2.4.4.2 wird die Heizung in Abhängigkeit der stationär erfassten Parameter Raumtemperatur und Nutzeranwesenheit gesteuert.

Schwächen

Bekleidungslevel: Bei der Erfassung der Raumtemperatur wird keine Rücksicht auf das Bekleidungslevel der Nutzer genommen. So kann einem Nutzer trotz optimaler Raumtemperatur zu warm oder zu kalt sein, in Abhängigkeit der Kleidung, die er trägt.

Aktivitätslevel: Das Aktivitätslevel des Nutzers wird ebenfalls nicht berücksichtigt. Sollte ein Nutzer beispielsweise Sport treiben und daher erhöhte Körpertemperatur aufweisen, kann dies nicht in die Temperatursteuerung einbezogen werden. Der Nutzer müsste manuell Änderungen an der Steuerung vornehmen, was einer automatisierten Steuerung entgegensteht.

Lösung durch Wearablesensoren

Durch die Messung der Körpertemperatur durch ein Wearable ist anzunehmen, dass die Schwäche des Bekleidungslevels gelöst werden kann, da die Messung direkt am Körper des Nutzers erfasst wird und dort bereits die Auswirkungen des Bekleidungslevels beinhaltet sind. Zusätzlich kann der Hautwiderstandssensor genutzt werden, um die Schweißproduktion des Nutzers zu überwachen, die ebenfalls Indiz für Überhitzung ist.

Für die Erfassung des Aktivitätslevels eignen sich die Sensoren Herzfrequenz, Oxymeter und Hautwiderstand. Mit diesen Sensoren lässt sich Aktivität anhand von erhöhter Herzfrequenz und Schweißproduktion erkennen. Es kann somit die Heizleistung dem aktuellen Aktivitätslevel angepasst werden.

Erwarteter Nutzen

Usability: Da es sich zuvor bereits um ein automatisiertes Szenario ohne Nutzerinteraktion gehandelt hat, werden durch das Wearable keine Arbeitsschritte eingespart. Bei

Implementierung müssen keine Sensoren im Raum installiert werden, da alle benötigten Sensoren durch das Wearable bereitgestellt werden.

Erhöhung der Präzision: Mithilfe des Wearables kann die Körpertemperatur des Nutzers ermittelt werden. Somit kann die Steuerung der Raumtemperatur genauer erfolgen. Stationäre Sensoren können beispielsweise das Bekleidungslevel nicht erfassen.

Generierung neuen Kontexts: Durch Messung der Vitalwerte mit Qxymeter, Herzfrequenz- und Hautwiderstandssensor wird neuer Kontext erzeugt, der in diesem Szenario zur Bestimmung des Aktivitätslevels genutzt wird.

Key Performance Indikatoren

Performance Indikatoren sind basierend auf den Arbeiten [18] und [19]

- Energieverbrauch - Vergleich beider Szenarien (ohne/mit Wearable)
- Körpertemperatur - Messung mit verschiedenem Bekleidungs- Aktivitätslevel

Zusätzlich sollte der neu erzeugte Kontext auf Korrektheit geprüft werden

- Aktivitätslevel - Prüfen ob körperliche Aktivität korrekt erkannt wird. Messung in Ruhe und Aktivitätsphasen.

Anforderungen

- R1: Implementierung eines Smart Space Szenarios mit Wearable - Nutzung von Temperatur, Herzfrequenz, Oxymeter und Hautwiderstandssensor ✓
- R2: Verbindung zwischen stationären und tragbaren Entitäten - Verbindung zwischen Wearable und Heizung ✓

2.5.1.3 Szenario 3 - Luftqualität

Im dritten Szenario aus Abschnitt 2.4.4.3 wird die Luftqualität anhand der CO2 und H2O Konzentration überwacht.

Schwächen

Keine: Da die Verteilung von CO2 und H2O im Raum stets homogen sein sollte, entsteht kein Nachteil durch die Positionierung des Sensors. Solange dieser an einer Stelle mit Luftzirkulation positioniert ist, sollte der gemessene Wert im gesamten Raum gleich sein.

Lösung durch Wearablesensoren

Die Nutzung eines tragbaren CO2 Sensors würde in diesem Szenario keinen Vorteil bringen, da das Gas im gesamten Raum gleich verteilt sein sollte. Es könnte lediglich der Wert des Oxymeters genutzt werden. Da es sich bei Wearable Oxymetern in der Regel um einfache Sensoren handelt, wird voraussichtlich der CO2 Gehalt des Bluts nicht erfasst, sondern ausschließlich der Sauerstoffwert. Da dieser jedoch von vielen weiteren Faktoren abhängt, beispielsweise Erkrankungen und Funktionsstörungen der Lunge / Niere, Kreislaufstörungen oder Stoffwechselstörungen, ist die Genauigkeit des Wertes für die Bewertung der Raumluft fragwürdig.

Erwarteter Nutzen

Usability:

Da es sich zuvor bereits um ein automatisiertes Szenario ohne Nutzerinteraktion gehandelt hat, werden durch das Wearable keine Arbeitsschritte eingespart. Bei Installation des Szenarios kann bei Nutzung von Wearablesensorik auf das Platzieren von stationären Sensoren im Raum verzichtet werden.

Erhöhung der Präzision: Nicht zu erwarten.

Generierung neuen Kontexts: Bedingt, die Messung des Blutsauerstoffwerts kann Aufschluss über eine verringerte O2 Konzentration in der Luft geben. Jedoch ist dieser Wert von weiteren Faktoren abhängig.

2.5.1.4 Szenario 4 - Identifikation

Im Szenario vier aus Abschnitt 2.4.4.4 soll ein Nutzer beim Betreten oder Verlassen des Raumes anhand seiner Körpergröße identifiziert werden. Diese wird mittels eines Ultraschallsensors im Türrahmen ermittelt.

Schwächen

Ungenaues Room Tracking: Das Szenario aus [32] ermöglicht es, Personen beim passieren einer Tür zu identifizieren. Jedoch kann nicht bestimmt werden, ob die Person den Raum gerade betritt oder verlässt. Somit kann die Belegung des Raumes nicht erfasst werden.

Lösung durch stationäre Sensoren: Um ein Room Tracking mit stationärer Sensorik zu ermöglichen, wurde beschlossen, das Szenario um die Nutzung eines Bewegungssensors zu erweitern. Dieser wird auf der Innenseite der Tür angebracht. Beim Öffnen der Tür

wird geprüft, ob eine Bewegung detektiert wurde. Ist dies der Fall, wird ein Verlassen des Raumes festgestellt. Eine nicht vorhandene Bewegung wird als Betreten interpretiert.

In einem Mehrbenutzerszenario wird die korrekte Erkennung der stationären Lösung jedoch eingeschränkt, da ein Nutzer innerhalb des Raumes den Bewegungssensor auslösen kann, während ein zweiter Nutzer im Begriff ist, den Raum zu betreten. Dieses Betreten würde dann fälschlicherweise als Verlassen gewertet. Zusätzlich benötigt der Bewegungssensor eine Zeitspanne zum Zurücksetzen. Betritt vor Ablauf dieser Zeitspanne ein Nutzer den Raum, wird dies ebenfalls als Verlassen gewertet.

Lösung durch Wearablesensoren: Durch Nutzung von Accelerometer und Gyroskop kann die Bewegung der Hand beim Öffnen der Tür aufgezeichnet werden. Es soll so erfasst werden, ob die Tür mittels Druck (Abb. 2.8) oder Zug (Abb. 2.9) geöffnet wird. Die Abbildungen zeigen einen Nutzer bei den jeweiligen Aktionen und den dabei zu erwartenden Ausschlag auf der Beschleunigungsachse. Bei einer Tür, die nur in eine Richtung schwingt, kann so festgestellt werden, ob der Nutzer den Raum betritt oder verlässt. So sollen einerseits die Genauigkeit des Room Tracking erhöht werden und andererseits die Schwächen der stationären Lösung behoben werden.

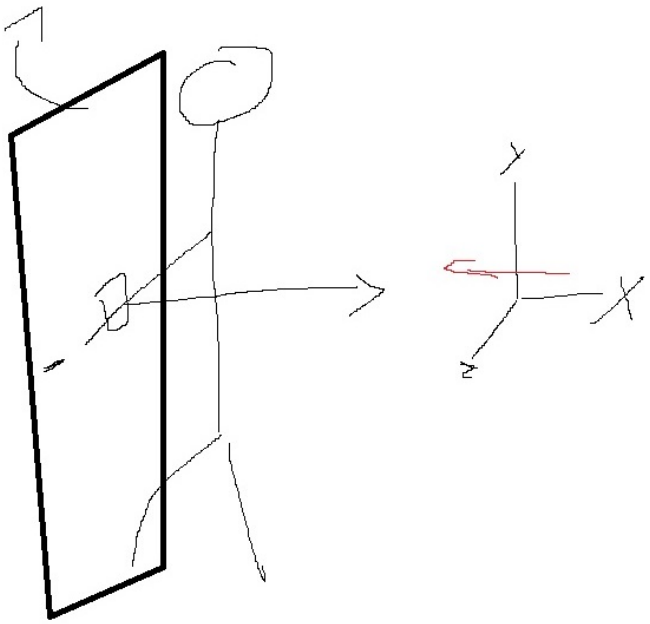


Abbildung 2.8: Tür durch Druck öffnen

Erwarteter Nutzen

Usability: Da es sich zuvor bereits um ein automatisiertes Szenario ohne Nutzerinteraktion gehandelt hat, werden durch das Wearable keine Arbeitsschritte eingespart. Es kann

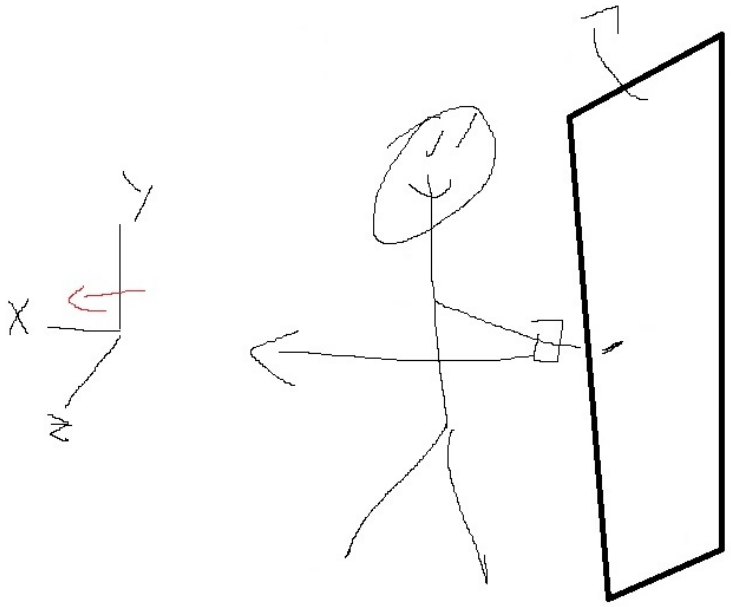


Abbildung 2.9: Tür durch Zug öffnen

in diesem Szenario auch nicht auf die Platzierung der stationären Sensoren verzichtet werden.

Erhöhung der Präzision: Die Genauigkeit des Room Tracking soll durch Auswertung der Werte von Accelerometer und Gyroskop erhöht werden. Dazu werden die Werte analysiert, um festzustellen, ob der Nutzer Druck oder Zug auf die Tür ausübt. Zusammen mit der Annahme, dass sich die Tür nur in eine Richtung öffnet, soll so ein Betreten oder Verlassen des Raumes erkannt werden. Die Ungenauigkeit der stationären Lösung in Mehrbenutzerumgebungen sollte durch das Wearable gelöst werden.

Key Performance Indikatoren

Um die Werte des neuen Kontexts bewerten zu können ist der anzuwendende KPI:

- Genauigkeit - Falsch-positiv/Falsch-negativ Raten durch praktischen Test erfassen, Vergleich stationär und Wearable

Anforderungen

- R1: Implementierung eines Smart Space Szenarios mit Wearable - Nutzung von Accelerometer und Gyroskop ✓
- R2: Verbindung zwischen stationären und tragbaren Entitäten - Verbindung zwischen Wearable, Beleuchtung, und Ultraschallsensor ✓

2.5.2 Auswahl der zu evaluierenden Szenarien

In diesem Abschnitt erfolgt Auswahl der Szenarien aus dem vorhergehenden Abschnitt für die Implementierung und die Evaluation des Wearablenutzens. Basis der Auswahl ist die Anwendbarkeit der KPI im Rahmen einer Masterarbeit sowie die Verfügbarkeit eines Geräts mit benötigter Sensorik.

2.5.2.1 Anforderungen an ein Wearable

Tabelle 2.7 fasst die Anforderungen der erweiterten Szenarien aus 2.5 an die Wearablesensorik zusammen.

Tabelle 2.7: Wearable Anforderungen

Szenario	Sensor	Latenz
Beleuchtung	Helligkeit	Echtzeit
Heizung	Temperatur, Herzfrequenz, Oxymeter, Hautwiderstand	Echtzeit
Luftqualität	Oxymeter	Echtzeit
Identifikation	Accelerometer, Gyroskop	Echtzeit

Betriebssystem

Nach Betrachtung der APIs in 2.3.4.2 fällt die Wahl des Betriebssystems auf Android Wear. Dieses erfüllt über seine API als einziges die geforderten Kriterien aus Tabelle 2.7, die benötigten Sensordaten in Echtzeit direkt vom Gerät auszulesen. Es bietet vollen Zugriff auf sämtliche im Gerät verbauten Sensoren. Zusätzlich können Apps über den zur Verfügung stehenden Appstore verteilt werden.

Durch den ähnlichen Aufbau zu Android können für Android Wear geschriebene Apps zudem bei Bedarf leicht auf Smartphones portiert werden und umgekehrt.

Tabelle 2.8 umfasst einen Ausschnitt der Tabelle 2.1 aus Abschnitt 2.3.1.3. Es sind alle betrachteten Geräte mit dem Betriebssystem Android Wear eingezeichnet.

Tabelle 2.8: Eingabegeräte

Gerät	Accelero	Gyroskop	H-freq	Oxy	Skin	Temp	Lux
Polar M600	✓	✓	✓	X	X	X	✓
Huawei Watch 2	✓	✓	✓	X	X	X	✓
Alcatel OneTouch	✓	✓	✓	X	X	X	X
Sony Smartwatch 3	✓	✓	X	X	X	X	✓

2.5.2.2 Anforderungen an die Szenarien

Damit ein Szenario für die Evaluation ausgewählt wird muss es zwei Anforderungen erfüllen:

- Hardware verfügbar
- KPI auswertbar

Als direktes Resultat aus R4 muss ein COTS Wearable existieren, dass die geforderte Sensorik bereitstellt. Zusätzlich müssen die KPI zur Feststellung des Wearablenutzens auswertbar sein, da basierend auf R3 eine Evaluation durchgeführt werden muss. Die Einhaltung von R1 und R2 wurde bereits bei Erstellung der erweiterten Szenarien sichergestellt.

Beleuchtung

Für die Umsetzung dieses Szenarios wird Echtzeitzugriff auf einen Helligkeitssensor benötigt. Das Szenario kann mithilfe von drei der vier Wearables aus Tabelle 2.8 implementiert werden, die einen Helligkeitssensor besitzen. Aktuell sind dies mindestens zwei Geräte, die auf der aktuellen Android Wear 2.0 Version basieren und den geforderten Sensor bieten.

Hardware verfügbar ✓

Die Auswertung des KPI *Helligkeit im Arbeitsbereich* kann im Rahmen eines Versuchsaufbaus im Labor untersucht werden.

KPI auswertbar ✓

Heizung

Um das erweiterte Heizungsszenario zu implementieren, werden die Sensoren Temperatur, Herzfrequenz, Oxymeter und Hautwiderstand benötigt. Keines der aktuell verfügbaren Wearables mit Android Wear Betriebssystem bietet eine Kombination dieser Sensoren. Lediglich Geräte mit Herzfrequenzmessung sind verfügbar (Tab. 2.8). Dieser Wert allein gibt zwar Hinweise auf das Aktivitätslevel des Nutzers, ohne Körpertemperatur oder Hautwiderstandssensor ist dies jedoch noch kein zwingendes Kriterium für eine Überhitzung des Nutzers.

Hardware verfügbar X

Die Evaluation des Indikators *Energieverbrauch* ist durch einen Laboraufbau möglich. Für die *Validierung der Körpertemperatur Messergebnisse* bei verschiedenen Bekleidungsleveln wird bereits eine größere Nutzergruppe benötigt, um repräsentative Ergebnisse

zu erhalten. Um die *Korrektheit der Aktivitätsmesswerte zu prüfen*, wird ebenfalls die Durchführung einer Nutzerstudie erforderlich, da je nach Fitnesslevel der Nutzer der Grad der Anstrengung variiert. Im Rahmen einer Masterarbeit sind diese Studien nur bedingt möglich.

KPI auswertbar –

Luftqualität

Aktuell ist laut Tabelle 2.8 kein Gerät mit einem Oxymeter verfügbar.

Hardware verfügbar X

Die Auswertung dieses Szenarios gestaltet sich schwierig, da ein prognostizierter Wearablenutzen fraglich ist. Da in dem Szenario keine Schwäche identifiziert wurden und somit kein Wearableeinsatz vorgesehen ist, sind keine KPI festgelegt.

KPI auswertbar X

Identifikation

Die Erweiterung dieses Szenarios benötigt Zugriff auf Accelerometer und Gyroskop in Echtzeit. Alle aktuell in Tabelle 2.8 betrachteten Geräte bieten diese Sensoren.

Hardware verfügbar ✓

Die Messung der falsch-positiv und falsch-negativ Raten der Gestenerkennung sind mithilfe eines Laborversuchs messbar.

KPI auswertbar ✓

2.5.2.3 Auswahl

Tabelle 2.9: Szenarien Auswahl

Szenario	Hardware verfügbar	KPI auswertbar
Beleuchtung	✓	✓
Heizung	X	–
Luftqualität	X	X
Identifikation	✓	✓

Die Wahl für die Szenarien, anhand derer der Wearablenutzen evaluiert werden soll, fällt auf die Szenarien **Beleuchtung** und **Identifikation**, da beide die gesetzten Anforderungen erfüllen (vgl. Tab.2.9). Sie können mithilfe verfügbarer Wearables implementiert

werden und die Messungen der KPI sind in Laborversuchen ohne ausgedehnte Studien möglich.

2.5.3 Zusammenfassung

In diesem Abschnitt wurden die in Abschnitt 2.4.4 ausgewählten Szenarien auf Schwächen hin untersucht. Daraufhin wurden Hypothesen aufgestellt, wie sich diese Schwächen mithilfe der in Abschnitt 2.3.3 identifizierten Wearablesensorik beseitigen lassen. Durch den Einsatz der Wearables sind erweiterte Szenarien mit Wearablenutzung entstanden. Für diese wurde geprüft, ob sie den zuvor gestellten Anforderungen genügen. Für jedes Szenario wurden zudem KPI festgelegt, deren Auswertung ein quantifizierbares Ergebnis zur Beantwortung der Leitfrage liefert.

- Erweiterte Wearable Szenarien erstellen ✓
- Hypothesen über Wearablenutzen aufstellen ✓
- Festlegen von Performance Indikatoren ✓

Zusätzlich wurden zwei Szenarien für die Evaluation des Wearablenutzens ausgewählt. Die Wahl ergab sich aus der Verfügbarkeit der benötigten Sensoren sowie der Auswertbarkeit der KPI zum feststellen des Wearablenutzens.

- Auswahl zu implementierender Szenarien ✓

2.6 DS2OS

Dieses Kapitel dient der Einführung in das *Distributed Smart Space Orchestration System* (DS2OS), das zur Implementierung der Smart Space Szenarien verwendet wird. Das System dient der Steuerung und Verwaltung von Smart Spaces. Es wird verwendet, damit der Fokus bei der Implementierung auf den jeweiligen Szenarien liegen kann. Sämtliche organisatorischen Tätigkeiten werden von der VSL Middleware des DS2OS übernommen. Dies soll eine schnelle Implementierung der Szenarien (Rapid Prototyping) ermöglichen.

Das System soll aus zwei Gründen während der Implementierung evaluiert werden. Einerseits soll erfasst werden, mit welchem Aufwand sich wearablegestützte Szenarien umsetzen lassen. Dazu kann ein direkter Vergleich zu den ursprünglichen Szenarien gezogen werden. Andererseits wird das System am Lehrstuhl für Netzarchitekturen und Netzdienste der TU München stetig weiterentwickelt. Eine Evaluation bei der Implementierung von repräsentativen Smart Space Szenarien ermöglicht es, den Entwicklern ein direktes Feedback über mögliche Verbesserungsmöglichkeiten zu geben.

Es werden im Folgenden die Architektur und die angewendeten Konzepte vorgestellt, um dem Leser eine Übersicht über das in dieser Arbeit genutzte System zu erlauben.

Zunächst soll die Entscheidung zur Verwendung einer Middleware begründet werden.

2.6.1 Middleware

Der Grund, eine Middleware zu benutzen, liegt an der Erleichterung der Implementierung. Statt sämtliche Funktionen zur Verbindung der einzelnen Entitäten im Smart Space implementieren zu müssen, kann auf eine Middleware zurückgegriffen werden. Das DS2OS bietet mit dem VSL bereits Mechanismen zur Speicherung von Daten und eine einfache API zum Lesen und Ändern der Daten. Zusätzlich wird die Verbindung zwischen den jeweiligen Services hergestellt. Der Fokus bei der Implementierung der Smart Space Szenarien kann so völlig auf die Funktionalität ausgerichtet werden.

2.6.2 VSL

Der Virtual State Layer (VSL) ist der Bestandteil des DS2OS , der die Funktion einer Middleware übernimmt. Er verwaltet die Kommunikation mit den einzelnen Entitäten im Smart Space und übernimmt die Datenspeicherung für Services, die oberhalb des VSL laufen. Es wird eine logische Abstraktionsebene geschaffen, auf der Services miteinander kommunizieren. Die Daten eines Services werden als Kontext bezeichnet und in Kontextknoten gespeichert. Der VSL bietet eine API, über die mittels einheitlicher Befehle nach dem Representational State Transfer (REST) Prinzip auf die Daten zugegriffen werden kann. Dies erleichtert das Hinzufügen neuer Services, da unabhängig von der genutzten Hardware dieselben Befehle genutzt werden können.

2.6.2.1 Knowledge Agents

Die Architektur des VSL basiert auf einem Peer-to-Peer (P2P) System. Bei dieser Architektur gibt es keine zentrale Instanz wie beispielsweise den Client-Server Modell, sondern viele gleichrangige Instanzen. Beim VSL sind dies die Knowledge Agents (KA, Abb. 2.10: Entität 1). Bei Ausführung verbindet sich der KA automatisch mit den anderen KA und ermöglicht den Zugriff und Austausch von Kontextdaten. Oberhalb der KA werden die einzelnen Services der Szenarien ausgeführt.

2.6.2.2 Kontextmodell

Jeder Service, der oberhalb des VSL ausgeführt wird, besitzt ein Kontextmodell. Dieses spezifiziert die Daten (Kontext), die von dem jeweiligen Service zur Verfügung

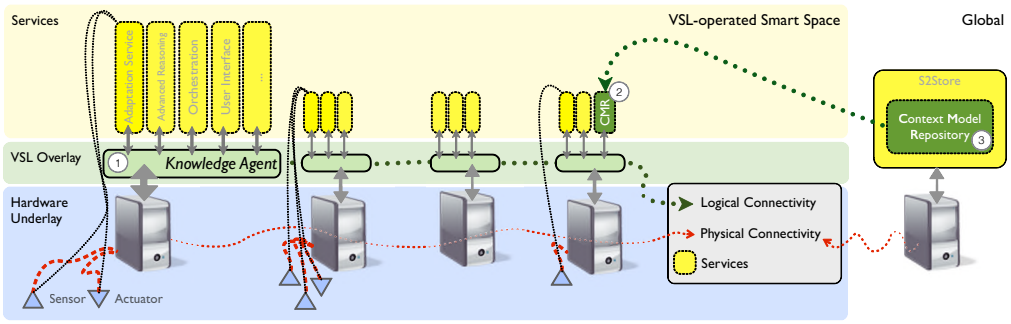


Abbildung 2.10: Aufbau Virtual State Layer [1]

gestellt werden. Die einzelnen Kontextmodelle werden in ein zentrales Context Model Repository (CMR, Abb. 2.10: Entitäten 2 und 3) gespeichert. Lokale Instanzen (Knowledge Agents) downloaden diese oder fügen Modelle hinzu. Die zentrale Verwaltung der Modelle garantiert Konsistenz. Zugefügte Modelle dürfen später nicht geändert werden. Neue Modelle können bestehende jedoch erweitern. Es ist ebenfalls möglich, das mehrere Services das gleiche Kontextmodell verwenden.

Zur Speicherung von Daten werden vier Grunddatentypen zur Verfügung gestellt:

- /basic/text - Speicherung von Strings
- /basic/number - Speicherung von Zahlen
- /basic/list - Speichert dynamisch wechselnde Daten
- /basic/composed - Zusammensetzung verschiedener Datentypen

2.6.2.3 Kontextknoten

Der VSL bietet eine abstrahierte Darstellung der Hardware des Smart Spaces, in der Zustände als Kontext gespeichert werden. Anwendungen werden als Service implementiert, die auf Kontextdaten zugreifen und diese ändern können. Im Folgenden werden die zwei verschiedenen Arten von Kontextknoten vorgestellt.

Reguläre Knoten—In regulären Knoten werden Zustandsdaten von einem Service (Abb. 2.11: Service 2) gespeichert und können von einem anderen Service (Abb. 2.11: Service 1) ausgelesen werden. Ein Beispiel ist ein Service (2), der die Rohdaten eines Temperatursensors ausgelesen hat und diese in einem regulären Kontextknoten speichert. Andere Services (1) können diesen Wert danach mittels *get*-Befehl lesen.

Virtuelle Knoten—Im Gegensatz zu regulären Knoten werden in virtuellen Knoten keine Werte gespeichert sondern Funktionen aufgerufen, die Werte dynamisch erzeugen. Diese

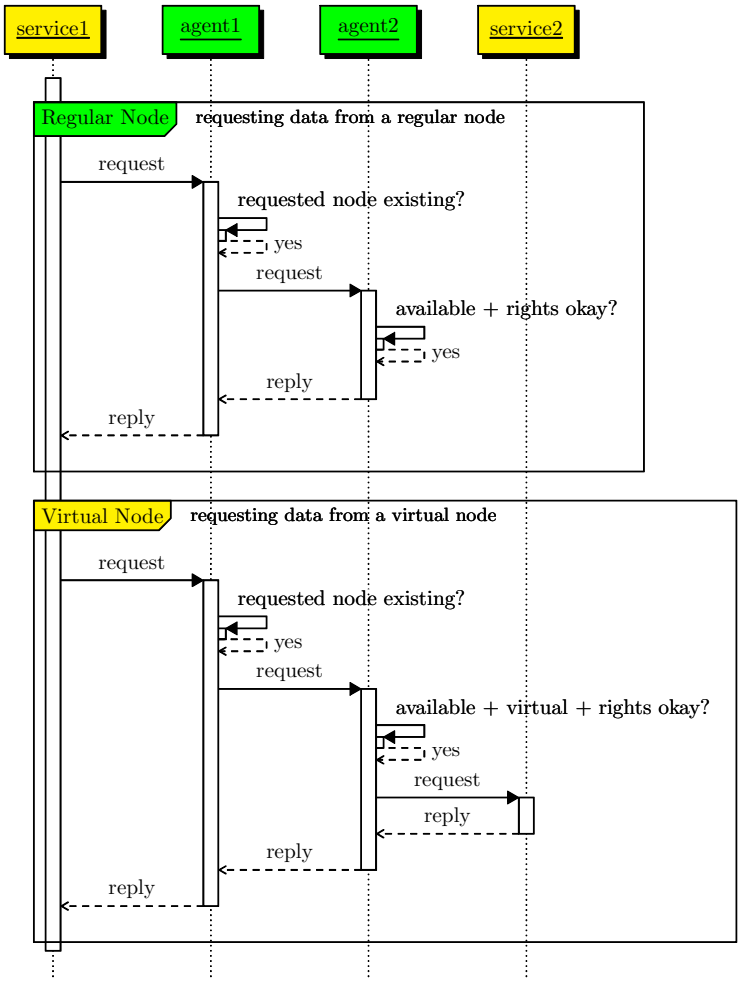


Abbildung 2.11: Kontextknoten [1]

Funktionen werden im Folgenden als Handler bezeichnet. Im vorherigen Beispiel könnte Service 2 ebenfalls ein Knoten für die Temperatur in °C bereitstellen. Wird dieser nun mittels *get*-Befehl gelesen, wird ein Handler (in Abb. 2.11 durch Service 2 implementiert) aufgerufen, der diese Temperatur am Sensor ausliest, umwandelt und als Rückgabewert ausgibt.

2.6.3 Zusammenfassung

In diesem Abschnitt wurde eine Übersicht über den Aufbau und die Konzepte des DS2OS gegeben. Mit diesem Wissen wird das Verständnis der im Kapitel Design zu treffenden Entscheidungen für die Implementierung der Szenarien mittels DS2OS erleichtert. Außerdem wurde begründet, warum das DS2OS zur Implementierung verwendet wird. Es soll durch Bereitstellung wichtiger Grundfunktionen zur Steuerung des Smart Spaces ein Rapid Prototyping der Szenarien ermöglichen.

- Vorstellung des DS2OS ✓

Kapitel 3

Verwandte Arbeiten

In diesem Abschnitt werden verwandte Arbeiten vorgestellt, in denen ebenfalls Wearables zur Erfassung von Daten innerhalb eines Smart Spaces verwendet wurden. Zusätzlich werden Evaluationen weiterer Smart Space Frameworks vorgestellt.

3.1 Szenarien mit Wearable Nutzung

In den folgenden Arbeiten wurden tragbare Sensoren zur Erfassung von Nutzerparametern genutzt. Es wurde bereits eine Vorauswahl getroffen, die nur Arbeiten umfasst, in denen Messungen innerhalb einer Smart Space Umgebung mithilfe von COTS Wearables durchgeführt wurden.

3.1.1 A Smart Space Application to Dynamically Relate Medical and Environmental Information

In [33] stellen Vergari et al. ein System vor, in dem Vitalparameter mithilfe tragbarer Sensoren gemessen und gemeinsam mit Umgebungswerten aus stationären Sensoren zugänglich gemacht werden. Ziel ist es, neue Cross Domain Anwendungen zu ermöglichen.

Zur Aufzeichnung der Vitalparemeter dient ein Zephyr Bioharness BT, mit dem Herzfrequenz, Körpertemperatur, Atemfrequenz und Körperhaltung erfasst werden. Zur Lokalisierung werden zusätzlich RFID-Tags verwendet. Die Aufzeichnung der Umgebungsparameter erfolgt über Intel iMote2 Sensoren und umfasst Raumtemperatur und Luftfeuchtigkeit. Die Werte werden in einer gemeinsamen Datenbank gespeichert und zugänglich gemacht.

Als Szenarien wurde ein Alarm vorgestellt, der ausgelöst wird, sobald einer der erfassten Werte einen zuvor festgelegten Grenzwert überschreitet, sowie ein Gesundheitsmonitor, der die Vital- und Umgebungswerte in Echtzeit visualisiert.

3.1.2 FamilyLog: A Mobile System for Monitoring Family Mealtime Activities

Mit FamilyLog [34] stellen Chongguang et al. ein System zur Erkennung von gemeinsamen Mahlzeiten innerhalb von Familien vor. Zur Bestimmung der Mahlzeiten werden Umgebungsgeräusche von Smartphones und Tablets aufgezeichnet sowie Gesten und Umgebungsgeräusche von Smartwatches. Das Audiosignal wird daraufhin auf das charakteristische Klirren untersucht, das bei den Mahlzeiten durch Geschirr und Besteck erzeugt wird. Zusätzlich werden anhand der Accelerometerwerte charakteristische Armbewegungen, die während des Essens auftreten, erkannt. Zudem kann das System bei Verwendung mehrerer Mikrofone erkennen, ob sich die Familienmitglieder unterhalten oder Fernsehen.

Das System wurde durch den Einsatz in 8 Testfamilien evaluiert. Die Ergebnisse zeigen, dass Mahlzeiten durchschnittlich zu 81% korrekt erkannt werden. Der Vergleich mit der rein durch Gesten oder rein durch Geräusche basierten Erkennung zeigt, dass diese für sich allein genommen in der Regel eine geringere Erkennungsrate aufweisen.

3.1.3 The Potential and Challenges of Inferring Thermal Comfort at Home Using Commodity Sensors

Huang, Rayoung und Newman stellen in [35] einen Ansatz vor, um das Wärmeempfinden von Nutzern innerhalb ihrer Wohnung festzustellen. Es wurden die Parameter Raumtemperatur und Luftfeuchtigkeit mithilfe stationärer Sensoren erfasst. Zusätzlich wurde ein Basis B1 Fitness Tracker eingesetzt, um Körpertemperatur und Hautleitwiderstand zu messen. Diese Werte wurden daraufhin verwendet, um eine Vorhersage über das Wärmeempfinden des Nutzers zu erzeugen.

Über einen Zeitraum von 5 Wochen wurden in 7 Haushalten mit insgesamt 11 Teilnehmern Testdaten aufgezeichnet. Von jedem Teilnehmer wurden pro Tag mindestens 6 Berichte verfasst, in denen äußere Umstände sowie aktuelles Temperaturempfinden festzuhalten waren. Während der Auswertungsphase wurden mithilfe des vorgestellten Ansatzes Vorhersagen anhand der Testdaten erzeugt. Diese wurden daraufhin mit dem tatsächlichen Nutzerempfinden aus den Berichten verglichen.

Als Ergebnis der Auswertung wurde festgehalten, dass durch Nutzung des Wearables der Fehler bei der Vorhersage im Vergleich zur rein stationären Messung halbiert werden konnte.

3.1.4 Zusammenfassung

Die in dieser Arbeit festgelegten Anforderungen an die erweiterten Szenarien sind:

- R1: Nutzung eines Wearables

- R2: Verbindung zwischen stationären und tragbaren Entitäten
- R3: Evaluation des Nutzens
- R4: Wearable muss ein COTS Produkt sein

Tabelle 3.1 zeigt die Anwendung der zuvor erstellten Anforderungen auf die jeweiligen Szenarien. Das Szenario aus [33] erfüllt R1, da Wearable Sensoren zur Erfassung der Vitalwerte genutzt werden. R2 wird jedoch nur teilweise erfüllt, da zwar auch stationäre Sensoren verwendet werden, aber keine Sensorfusion stattfindet und ebenfalls keine Aktoren im Raum genutzt werden. Es handelt sich um ein ausschließliches Überwachungsszenario. R3 wird nicht erfüllt, da zwar festgestellt wird, dass Werte erfasst werden können, jedoch keine Aussage zu deren Korrektheit getroffen wird. R4 wird durch die Nutzung eines Zephyr Bioharness BT erfüllt.

Durch [34] wird R1 erfüllt, da Smartwatch Sensoren zur Erfassung von Geräuschen und Bewegungen verwendet werden. R2 wird hingegen nicht erfüllt, da ausschließlich Messwerte der tragbaren Sensoren verwendet werden und keine Aktoren im Smart Space eingebunden werden. R3 wird erfüllt, da eine Auswertung der Erkennungsrate der Mahlzeiten erfolgt. R4 wird erfüllt, da eine Sony Smartwatch 3 zur Erfassung der Messwerte verwendet wird.

Das in [35] untersuchte Szenario erfüllt R1 durch Nutzung der mobilen Temperatur- und Hautleitwiderstandsensoren. Es erfolgt eine Sensorfusion durch die Verbindung mit den stationären Temperatur- und Luftfeuchtigkeitssensoren. Da jedoch keine konkrete Anwendung in einem Smart Space Szenario stattfindet und keine Aktorik mit den Sensoren verbunden ist, wird R2 nur zum Teil erfüllt. Die Korrektheit der Messwerte wird durch eine Nutzerstudie überprüft, wodurch R3 erfüllt wird. Durch die Nutzung eines Basis B1 Fitness Trackers wird R4 erfüllt.

Zusammenfassend erfüllen alle genannten Szenarien die Anforderungen R1 und R4, die jedoch ohnehin Grundbedingung für die Betrachtung der Szenarien sind. Keines der Szenarien erfüllt jedoch R2 in vollem Umfang. Selbst wenn eine Sensorfusion stattfindet, wird das Ergebnis nicht als Eingangswert für eine Steuerungslogik verwendet. Der Nutzen im konkreten Szenario wird also nicht aufgezeigt. Die Evaluation der erfassten Daten (R3) wird bei immerhin zwei der Arbeiten durchgeführt.

Das Fehlen eines Szenarios, dass sämtliche Anforderungen erfüllt zeigt, dass bisher noch keine Evaluation der Nutzung von COTS Wearables in IoT Szenarien durchgeführt wurde. Dieses wird nun im Rahmen dieser Arbeit durchgeführt.

Tabelle 3.1: Verwandte Arbeiten

Arbeit	Erfasste Werte	Ziel	R1/R4	R2	R3
A Smart Space Application to Dynamically Relate Medical and Environmental Information	Accelerometer, Atemfrequenz, Herzfrequenz, Körpertemperatur	Überwachung der Vitalwerte und Körperhaltung	✓	–	X
FamilyLog: A Mobile System for Monitoring Family Mealtime Activities	Accelerometer, Mikrofon	Erkennung von Gesten	✓	X	✓
The Potential and Challenges of Inferring Thermal Comfort at Home Using Commodity Sensors	Hautwiderstand, Körpertemperatur	Erfassung des Wärmeempfindens	✓	–	✓

3.2 Evaluationen weiterer Smart Space Frameworks

Im Folgenden werden Evaluationen weiterer Frameworks vorgestellt, um die jeweils angewendete Methodik zu erfassen und die Qualitätskriterien der Bewertung zu extrahieren.

3.2.1 Evaluation of an Agent Framework for the Development of Ambient Computing

In [36] stellen Rodríguez und Favela eine Evaluation der SALSA Middleware zur Implementierung von Ubiqutous Computing Szenarien vor.

Die Evaluation gliedert sich in die Ausführung zweier Experimente.

Im erstem Experiment sollte das Kriterium *Ease of Use* ausgewertet werden. Zu diesem Zweck wurde zunächst mithilfe eines Fragebogens der Kenntnisstand der Testgruppe, bestehend aus 19 Teilnehmen, im Entwickeln von Software Systemen festgestellt. Im nächsten Schritt wurden den Teilnehmern die Konzepte, ein Programmbeispiel sowie das folgende Laborexperiment vorgestellt.

Das Laborexperiment bestand aus der Implementierung dreier Teilaufgaben mittels des Frameworks. Während der Durchführung wurden Fragen und Kommentare der Teilnehmer aufgezeichnet sowie der entstandene Code gesichert. Im Anschluss an die praktische Phase wurde den Teilnehmern ein Fragebogen zur Ease of Use Bewertung gestellt, der Aussagen mit einer Zustimmungsskala von 0 bis 7 enthielt.

Während der Evaluation des Experiments wurde die Funktionalität des Code analysiert,

und ob grundlegende Designkonzepte des Frameworks angewendet wurden. Zudem wurden die aufgetretenen Fragen analysiert.

Das zweite Experiment hatte das Ziel, den Ease of Use beim System Design zu ermitteln. In diesem Fall wurde einer Gruppe aus 18 Teilnehmern ein Szenario sowie ein dazugehöriges UML-Sequenzdiagramm zusammen mit 4 Aufgaben gegeben:

- Analyse und Identifikation von Systemkomponenten
- Erweiterung des Szenarios, Modifikation des Sequenzdiagramms
- Erstellung eines Komponentendiagramms der Erweiterung
- Detaillierte Beschreibung aller Komponenten eines Agents(perception, reasoning, action)

In der darauf folgenden Auswertung wurde überprüft, wie gut die Konzepte verstanden und umgesetzt wurden.

Zentrale Probleme, die unabhängig vom System aufgetreten sind, waren Unklarheiten im Umgang mit dem *.XML*-Format sowie der Erstellung eines Sequenzdiagramms mittels UML

3.2.2 One.world: Experiences with a Pervasive Computing Architecture

Mit [37] stellt Grimm eine Evaluation der One.world Pervasive Computing Architektur vor.

Das System wird anhand von 4 Kriterien evaluiert:

- Vollständigkeit: Können auch komplexe Szenarien gebaut werden?
- Komplexität: Wie schwer ist es, Code zu verfassen?
- Performanz: Performanz auch bei realer Auslastung okay, schnelle Antwortzeiten?
- Nutzbarkeit: Können reale Anwendung on Top programmiert werden? Wichtigstes Kriterium.

Zur Bewertung der Kriterien werden eine Reihe von Diensten mithilfe von One.world implementiert. Die Auswertung erfolgt anhand der Erfahrungen und Daten, die während der Implementierung gesammelt wurden. Es werden die benötigte Zeit, eventuelle Grenzen des Systems, sowie das Verhalten der Dienste im Bezug auf Performanz und Nutzbarkeit berücksichtigt. Zusätzlich zu den zuvor festgelegten Kriterien werden die Erfahrungen im Detail vorgestellt und Lehren über weitere Anforderungen an das System gezogen. Zudem erfolgt eine Diskussion über die angewendeten Kriterien, deren Angemessenheit, mögliche weitere Kriterien sowie allgemeine Probleme bei der Evaluation.

3.2.3 Zusammenfassung

In beiden Evaluationen werden unterschiedliche Methoden zur Evaluation genutzt. In [36] werden dafür Testnutzer in zwei Testgruppen verwendet, die nach der Nutzung der Middleware mittels Fragebögen das System zum Kriterium Ease of Use bewerten durften. Zusätzlich wurde der entstandene Code bewertet.

In [37] werden die Evaluationsdaten durch die Implementierung verschiedener Dienste durch die Entwickler erfasst.

Tabelle 3.2: Weitere Evaluationen

Arbeit	Methodik	Qualitätskriterien
Evaluation of an Agent Framework for the Development of Ambient Computing	Testnutzer, Fragebögen, Codeinspektion	Ease of Use
One.world: Experiences with a Pervasive Computing Architecture	Heuristische Evaluation	Vollständigkeit, Komplexität, Performanz, Nutzbarkeit

Kapitel 4

Design

In diesem Abschnitt werden die entscheidenden Schritte zur Vorbereitung der Implementierung gelegt. Diese umfassen die folgenden Meilensteine:

- VSL Design der Szenarien
- Erstellung der Testfälle
- Vorbereitung Softwareevaluation

  Festlegen der DS2OS Evaluationskriterien

  Vorbereitung Cognitive Walktrough

  Vorbereitung Heuristische Evaluation
- Erheben der Daten: Design

Zu Beginn des Kapitels werden zunächst die Vorbereitungen für die DS2OS Evaluation getroffen. Diese bestehen aus der Auswahl der Evaluationskriterien und -methoden und deren spezifischer Vorbereitung. Darauf folgt das VSL spezifische Design der ursprünglichen Smart Space Szenarien. Im letzten Abschnitt erfolgt das Design der erweiterten Szenarien sowie der Entwurf der Testfälle zur Evaluation des Wearablenutzens. Da die Erstellung der Testfälle jedoch ein iterativer Prozess ist, für den Ergebnisse aus den Wearableszenarien benötigt werden, findet dieser zeitgleich mit Implementierung der Szenarien statt.

4.1 Kriterien zur DS2OS Evaluation

Um eine Evaluation des DS2OS durchführen zu können, müssen zunächst die zu betrachtenden Qualitätskriterien festgelegt werden. Darauffolgend müssen Methoden definiert werden, nach denen diese Kriterien bewertet werden.

Da Evaluiert werden soll, wie gut sich DS2OS für die schnelle Implementierung der Smart Space Szenarien eignet, ist die Nutzbarkeit (engl. Usability) ein zentraler Faktor. Die Usability entscheidet darüber, wie schnell der Nutzer den Umgang mit der Software erlernt und in der Lage ist, produktiv mit ihr zu arbeiten. Da sie großen Einfluss auf ein mögliches Rapid Prototyping hat, ist die Usability das Hauptevaluationskriterium.

Die Basis für die Auswahl der Evaluationsmethoden bildet [38], in der Kropp eine Usability-Analyse des DS2OS mithilfe zweier Testgruppen durchgeführt hat. Während der Vorbereitung der Evaluation werden verschiedene Methoden analysiert und ausgewertet. Diese Arbeit soll auf den vorhanden Ergebnissen dieser Analyse aufbauen, die im Folgenden noch einmal kurz vorgestellt werden.

4.1.1 Evaluationsmethoden

Kropp stellt eine Reihe von Evaluationsmethoden vor, aus denen sie, nach Analyse der jeweiligen Vor- und Nachteile, vier für die spätere Evaluation auswählt. Diese vier Methoden sind:

Cognitive Walktrough—Im Rahmen des Cognitive Walthrough bewerten ein oder mehrere Gutachter die Handlungsabläufe, die während der Bewältigung der Testaufgabe aufgetreten sind und vergleichen diese mit den Erwarteten. Diese Technik verwendet die grundlegende Annahme, dass ein Programm mit guter Usability intuitiv zu erwünschten Denk- und Handlungsabläufen führt.

Heuristische Evaluation—Bei der heuristischen Evaluation bewerten ein oder mehrere Gutachter die Benutzerschnittstelle eines Programms. Es wird untersucht, inwiefern bestimmte Usability Kriterien (Heuristiken) eingehalten werden.

Retrospective Probing—Beim Retrospective Probing stellt ein Gutachter am Ende einer Testaufgabe Nutzern eine Reihe von Fragen über die Bewältigung der Aufgabe.

Fragebogen—Mit einem Fragebogen wird einer Gruppe von Testnutzern nach Bewältigung der Testaufgabe eine Reihe von Fragen über die Nutzererfahrung mit dem Programm gestellt.

Da die Evaluation in dieser Arbeit nicht mithilfe einer Gruppe von Testnutzern durchgeführt wird, werden Methoden benötigt, die vom Tester allein durchgeführt und ausgewertet werden können. Aus diesem Grund sind die Methoden *Retrospective Probing* und der *Fragebogen* nicht anwendbar.

Somit bleiben die *heuristische Evaluation* sowie der *Cognitive Walktrough* als mögliche Methoden. Da die Auswahl der Methoden sorgfältig und schlüssig begründet werden und die Kombination dieser beiden Methoden ein sinnvolles Gesamtbild aus allgemeiner Analyse mittels Heuristiken sowie detaillierter Analyse der Teilschritte liefert, werden diese beiden Methoden auch in dieser Arbeit angewendet. Zudem wurde die heuristische Evaluation in der verwandten Arbeit [37] angewendet, in der die Evaluation ebenfalls durch die Entwickler, ohne Zuhilfenahme einer Testgruppe, durchgeführt wurde. Für beide Methoden müssen jedoch zunächst die Leitfragen und das Vorgehen vorgestellt werden.

4.1.1.1 Cognitive Walkthrough

Beim Cognitive Walkthrough werden die einzelnen Teilschritte des Designs und der Implementierung der Szenarien anhand mehrerer Leitfragen bewertet. In diesem Abschnitt erfolgt die Vorstellung der Leitfragen und des Vorgehens.

Leitfragen

Basierend auf [39] wurden für den Cognitive Walkthrough folgende Aussagen herausgestellt. Diese werden im Rahmen der Evaluation in dieser Arbeit für jedes Szenario betrachtet. Die Bewertung gliedert sich in eine Zustimmung zur jeweiligen Aussage auf einer Skala von 1 (keine Zustimmung) bis 4 (volle Zustimmung) sowie eine Diskussion über die Gründe der Auswahl.

- Die Nutzer werden versuchen, den gewünschten Effekt zu erzielen.
- Die Nutzer werden erkennen, dass die korrekte Handlung ausgeführt werden kann.
- Die Nutzer werden erkennen, dass die korrekte Handlung zum gewünschten Effekt führen wird.
- Die Nutzer werden den Fortschritt erkennen, wenn sie die korrekte Handlung ausgeführt haben.

Vorgehen

Zunächst werden während der Vorbereitungsphase die zu erwartenden Nutzergruppen analysiert. Dann werden die idealen Handlungsabläufe für jedes Szenario festgelegt und schließlich die Schnittstelle des DS2OS definiert. Nach der Vorbereitung erfolgt die Erfassung der Daten, indem für jedes Szenario die Leitfragen beantwortet werden.

Annahmen über die Nutzer—Da die Evaluation vom Verfasser der Arbeit selbst durchgeführt wird, werden dessen Kenntnisse als Grundlage gewählt. Diese sind grundlegende Programmiererfahrung mit der benötigten Sprache Java, sowie dem für die Kontextmodelle benötigten XML. Zusätzlich wurde bereits eine Einführung in den Umgang mit DS2OS absolviert.

Ideale Handlungsabläufe—Grundlage für die Definition des idealen Handlungsablaufs ist die Betrachtung des jeweiligen Algorithmus des Szenarios. Nutzer sollten nach Einführung in den Aufbau des DS2OS in der Lage sein, die Building Blocks mit ihren jeweiligen Teilaufgaben zu identifizieren. Für jeden dieser Blöcke sollen die Nutzer neben der Funktionalität zusätzlich den In- und Output in VSL spezifischem Kontext mit Datentyp und Speicherung als regulärer oder virtueller Knoten angeben können. Die zu identifizierenden Einzelaufgaben entsprechen den in 4.2 und 4.3 definierten Building Blocks. Anhand der Building Blocks können dann die Services mit den jeweiligen Registrierungen und Handlerfunktionen implementiert werden.

Definition der Schnittstelle—Dieser Abschnitt umfasst eine Beschreibung der Programmschnittstelle, die der Nutzer bei der Implementierung der Komponente zu sehen bekommt. Da es beim DS2OS keine festgelegte Programmierumgebung gibt, ist es dem Nutzer überlassen, welche IDE er nutzt. Im Folgenden wird Eclipse verwendet. Die Schnittstelle des DS2OS bilden Templates für die Implementierung der Komponenten. Da diese einheitlich sind, werden die Templates nicht für jedes Szenario oder jeden Building Block einzeln vorgestellt.

Für die Erstellung eines Services mit DS2OS werden zwei Templates bereitgestellt. Beide Klassen realisieren die Trennung zwischen logischer Verbindung des Services mit dem Kontext anderer Knoten (Wiring) sowie der Implementierung der Service Funktionalität (Logic). Abbildung 4.1 soll diese Trennung noch einmal verdeutlichen. Die relevanten Abschnitte der Templateklassen befinden sich im Appendix A.

Mit der Klasse *ServiceTemplate.java* (Abb. 4.1 *ServiceWiring.java*) werden die logischen Verbindungen des Services mit dem VSL implementiert. Der eigene Kontext wird registriert und es werden Kontextknoten des eigenen oder anderer Services abonniert, sowie Handlerfunktionen für virtuelle Knoten registriert. Zum Abonnieren regulärer Knoten wird Funktion *registerSubscriptions* verwendet. Die eigentliche Registrierung erfolgt dann durch Aufruf der *subscribe* Funktion mit Angabe des Pfads. Die Aktion, die bei Änderung des abonnierten Knotens aufgerufen werden soll, wird durch die *notificationCallback* Funktion spezifiziert.

Zum Registrieren von Handlerfunktionen für virtuelle Knoten wird die Funktion *registerVirtualNodeHandlers* verwendet. Mit der Funktion *registerVirtualNode* werden die einzelnen virtuellen Knoten mit Angabe des Pfades registriert. Mittels *set*, *get*, *subscribe*

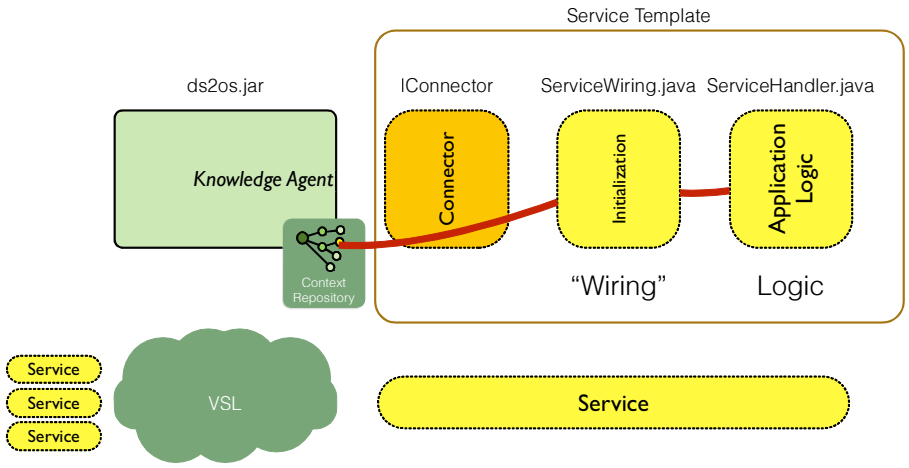


Abbildung 4.1: Logische Trennung der Templates [1]

und *unsubscribe* werden die einzelnen Handler für diese Funktionalitäten des Knotens registriert.

Die Implementierung aller aufgerufenen Handlerfunktionen erfolgt in der Klasse *ServiceTemplateHandler.java* (Abb. 4.1 *ServiceHandler.java*)

4.1.1.2 Heuristische Evaluation

Die heuristische Evaluation umfasst die Bewertung des Designs und der Implementierung mit DS2OS anhand einer Liste von Heuristiken. Diese Liste sowie der Ablauf der Evaluation werden im Folgenden vorgestellt.

Heuristiken

Die zu nutzenden Heuristiken wurden auf Grundlage von [39] gewählt. Die Auswahl basiert auf der Liste genereller Heuristiken nach J. Nielsen sowie der DIN EN ISO 9241-110 Norm. Es wurde eine Auswahl aus diesen Heuristiken vorgenommen, die in dieser Arbeit in vier Kategorien eingeteilt wird. Die Kategorien folgen der Annahme, dass ein Nutzer zuerst verstehen muss, was er mit dem System tun muss. Dies sollte möglichst intuitiv geschehen, denn er kann mit der Bearbeitung erst anfangen, wenn er weiß, wie er das System zum Lösen seiner Aufgabe verwenden muss. Danach sollte er seine Aufgabe möglichst einfach und komfortabel erledigen können. Dies stellt Anforderungen an die Bedienbarkeit, da benötigte Funktionen möglichst schnell gefunden und einfach angewendet werden sollen. Während der Bearbeitung sollte der Nutzer immer eine Rückmeldung über den aktuellen Systemstatus erhalten, um die Auswirkungen auf das

System, die seine Änderungen verursacht haben, überwachen zu können. Zusätzlich sollten Fehler schon beim Design der Software vermieden und bei Auftreten effizient behandelt werden, entweder automatisch durch das System oder mithilfe konstruktiver Fehlermeldungen durch den Nutzer.

Erkennbarkeit—Diese Kategorie befasst sich mit der grundsätzlichen Frage, wie leicht es dem Nutzer erkennbar ist, was er mit dem System tun muss, um sein Ziel zu erreichen.

Bedienbarkeit—Diese Kategorie umfasst Heuristiken zur Untersuchung der Frage, wie dem Nutzer das Erreichen seines Ziels durch das System erleichtert wird.

Systemstatus—In diese Kategorie fallen Heuristiken zur Untersuchung der Frage, ob dem Nutzer der aktuelle Systemstatus erkennbar ist.

Fehler—Diese Kategorie befasst sich mit der Frage, wie Fehler verhindert werden und wie aufgetretene Fehler behandelt werden.

Die genutzte Liste mit allen ausformulierten Unterpunkten befindet sich im Appendix B.

Vorgehen

Die Datenerfassung für die heuristische Evaluation erfolgt während des Designs sowie der Implementierung der Szenarien. Sie erfolgt nicht speziell für einzelne zu implementierende Komponenten, sondern bewertet die gesamte Implementierung und das Design mit DS2OS. Dazu werden ab Kapitel 4.2 alle zu erledigenden Aufgaben anhand der Heuristiken bewertet.

4.1.2 Weitere Qualitätskriterien

Die Evaluation des Rapid Prototyping soll auch explizit bewertet werden. Dazu werden weitere Werte benötigt, die durch den Cognitive Walkthrough und die heuristische Evaluation noch nicht erfasst werden. Aus diesem Grund werden in der Evaluation zusätzlich die **Zeiten** gemessen, die für die Einrichtung des DS2OS sowie der Vorbereitung und Implementierung von Szenarien benötigt werden. Da es das Ziel ist, möglichst schnell von der Einrichtung des DS2OS bis zum laufenden System zu kommen, werden die Zeiten aller dafür benötigten Schritte erfasst. So kann auch der Mehraufwand bei Nutzen eines Wearables explizit erfasst werden.

Darüber hinaus werden die Lines of Code (**LoC**) erfasst. Diese Metrik wurde von Kropp mit der Begründung verworfen, dass die Ergebnisse primär Rückschlüsse auf die Programmierfähigkeiten des Nutzers zulassen und nicht auf die Qualität des genutzten Programms. [38] Dies ist eine schlüssige Begründung für einen Ausschluss, jedoch werden für die Programmierung mit DS2OS Templates gegeben. Da diese dem Nutzer bereits ein Schema und die generelle Struktur des Codes vorgeben, fallen die zu erwartenden individuellen Abweichungen zwischen den Nutzern geringer aus, sodass ein qualitatives Ergebnis bei Betrachtung dieser Metrik erwartet wird.

Um zusätzlich den Vorteil der **Wiederverwendbarkeit** durch den modularen Aufbau der Szenarien zu erfassen, wird betrachtet, in wie vielen Szenarien jeder Service verwendet wird.

Die Kriterien, die zusätzlich zu den in Kapitel 4.1.1.1 und 4.1.1.2 vorgestellten Methoden erfasst werden, sind:

- Zeit für:
  - Erstellung von Building Blocks und Kontext
  - Installation des DS2OS
  - Verkabelung der Entitäten
  - Schreiben der Firmware
  - Implementierung der VSL Services
- Lines of Code
- Anzahl an Wiederverwendungen einer Komponente

4.1.3 Zusammenfassung

In diesem Abschnitt wurden der Cognitive Walktrough sowie die heuristische Evaluation als Evaluationsmethoden für das DS2OS festgelegt. Es wurden die benötigten Leitfragen und Heuristiken festgelegt, sowie das Vorgehen vorgestellt. Im letzten Abschnitt wurden zusätzlich zu erhebende Werte zur Bewertung des Rapid Prototyping definiert.

- Vorbereitung Softwareevaluation ✓
  - Festlegen der DS2OS Evaluationskriterien ✓
  - Vorbereitung Cognitive Walktrough ✓
  - Vorbereitung heuristische Evaluation ✓

Mithilfe der Ergebnisse aus diesem Abschnitt kann im nächsten Schritt mit der Erhebung der Evaluationsdaten begonnen werden.

4.2 Design der ausgewählten Szenarien mit DS2OS

In diesem Abschnitt erfolgt das Design der Building Blocks der in Kapitel 2.4.4 definierten Szenarien für die Umsetzung mit DS2OS. Dazu werden die benötigten Services, Handler und Registrierungen sowie die jeweiligen Kontextmodelle vorgestellt.

Vorher wird zunächst auf die verwendete Hardware eingegangen. Es wird der verwendete Controller für die Anbindung der Entitäten vorgestellt. Darüber hinaus werden die einzelnen Entitäten vorgestellt, um die Rohdaten der Sensoren sowie die erwarteten Eingabewerte der Aktoren für das Softwaredesign zu spezifizieren.

4.2.1 Hardware

4.2.1.1 Arduino

Zur Anbindung der Sensoren wird ein Arduino Mega 2560 mit Ethernet Shield verwendet (Abb. 4.2). Er bietet insgesamt 54 digitale I/O Pins sowie 16 analoge Eingänge.

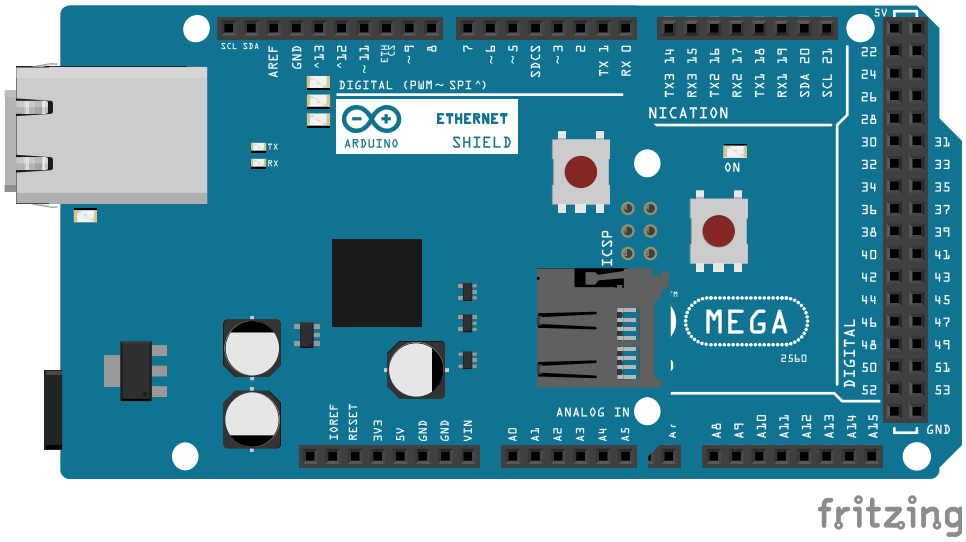


Abbildung 4.2: Arduino Mega 2560 mit Ethernet Shield

4.2.1.2 Sensoren

Helligkeitssensor BH1750—Beim BH1750 handelt es sich um einen digitalen Helligkeitssensor, der über den I2C Bus angesteuert wird. Die Ausgabe der Helligkeit erfolgt bei Verwendung der zugehörigen Bibliothek [40] direkt in der Einheit Lux.

Bewegungssensor HC-SR501—Der HC-SR501 Bewegungssensor bietet die Möglichkeit, die Erkennung von Bewegungen in einem bestimmten Bereich über einen digitalen Ausgang an den Arduino zu übermitteln. Die Belegung des Binärwerts ist:

- 0 - keine Bewegung erkannt
- 1 - Bewegung erkannt

Mittels Jumper auf der Platine kann außerdem aus zwei Modi gewählt werden:

- Modus 1 - Das Signal wird deaktiviert, auch wenn noch eine Bewegung erkannt wird und wird danach neu aktiviert. So wird ein Pegelwechsel erzwungen.
- Modus 2 - Das Signal bleibt pausenlos aktiv, solange eine Bewegung erkannt wird.

Für die Implementierung wird der Modus 2 gewählt, da kein erzwungener Pegelwechsel benötigt wird. Der Ausgabewert soll sich nur ändern, wenn keine Bewegung mehr erkannt wird.

Temperatur- und Luftfeuchtigkeitssensor KY-015—Der KY-015 Sensor ermöglicht die Erfassung der Raumtemperatur sowie der Luftfeuchtigkeit. Er wird digital angesprochen und liefert bei Nutzung der *Adafruit_DHT* Bibliothek [41] die Temperaturwerte in Grad Celsius sowie die Feuchtigkeitswerte in Prozent.

CO2 Sensor SKU:SEN0159—Der SKU:SEN0159 CO2 Sensor besitzt einen digitalen sowie einen analogen Ausgang. Für den digitalen Ausgang kann mithilfe eines Reglers der Schwellenwert zur Auslösung eingestellt werden. Um den aktuellen CO2 Wert auszulesen, muss der analoge Ausgang verwendet werden. Mithilfe der korrekten Umrechnungsfunktion kann so der CO2 Wert in der Einheit *parts per million* (ppm) ausgegeben werden.

Ultraschallsensor HC-SR04—Der HC-SR04 Sensor erfasst die Entfernung zu Objekten mithilfe einer Ultraschallmessung. Er wird digital an den Arduino angebunden und ermöglicht bei Nutzung der *NewPing* Bibliothek [42] die Angabe der Entfernung zum erfassten Objekt in Zentimetern.

Reedschalter KY-025—Der digitale Reedsensor KY-025 erfasst Magnetfelder und ermöglicht die Ausgabe eines digitalen Binärwerts. Die Belegung ist:

- 0 - Kein Magnetfeld erkannt
- 1 - Magnetfeld erkannt

4.2.1.3 Aktoren

Infrarot Transmitter KY-005—Der KY-022 Infrarot Transmitter ermöglicht die Steuerung von infrarotbasierten Fernsteuerungen. Er wird über einen digitalen Anschluss mit dem Arduino verbunden und kann mithilfe der *IRremote* Biblliothek [43] angesprochen werden.

Verdunkelung—Die Verdunkelung des Raumes erfolgt durch elektrisch steuerbare Jalousien. In [44] wurde bereits eine Blendensteuerung implementiert, die auch in dieser Arbeit verwendet wird. Sie ermöglicht das Schließen der Verdunkelung durch Setzen eines Prozentwerts.

Lüftung—Zur Steuerung wird ebenfalls ein bereits vorhandenes Gateway aus [44] genutzt. Es wird einen Binärwert zum Ein- und Ausschalten des Lüfters erwartet.

Thermostat—Ein Thermostat steht aktuell nicht zur Verfügung. DS2OS bietet jedoch die Möglichkeit, einen Thermostat auf Softwareebene zu simulieren. Da im Szenario Heizung keine Messungen durchgeführt werden, ist das Fehlen eines realen Thermostats nicht kritisch. Das simulierte Thermostat bietet einen Eingabewert, der die Öffnung des Thermostats in Prozent angibt.

Alarm—Der Alarm wird einerseits digital mit Fehlermeldung im VSL gespeichert, zusätzlich wird ein Alarmlicht aktiviert. Der Zugriff erfolgt über ein bereits in [44] implementiertes Gateway. Es wird ein Binärwert zum Ein- und Ausschalten erwartet.

4.2.2 Building Blocks

In diesem Abschnitt werden die für jedes Szenario benötigten Building Blocks identifiziert. Ein Building Block beschreibt im Folgenden eine einzelne Softwarekomponente, die eine im Szenario benötigte Teilfunktion bereitstellt. Dieser modulare Aufbau erhöht die Flexibilität, da einzelne Blöcke leichter angepasst, ausgetauscht und auch in mehreren Szenarien genutzt werden können. Im Rahmen der Implementierung im VSL entspricht ein Building Block einem einzelnen Service.

Es werden drei Ebenen verwendet, auf denen Services implementiert werden können.

Firmware beschreibt dabei die unterste Ebene. Sie umfasst die Firmware für die Arduino sowie weitere hardwarespezifische Software, die den REST-Zugriff hardwareseitig implementiert. Die jeweiligen VSL-Services, die eine Verbindung zur Hardware realisieren, werden ebenfalls zur Firmware gezählt. Sie lesen oder setzen die Werte der Hardware mittels REST-Anfragen und laden die Ergebnisse in VSL-Kontextknoten. Software, die der Firmware angehört, ist demnach abhängig von der verwendeten Hardware.

Services, die der Ebene **Adaption** zugeordnet werden, besitzen keine Abhängigkeit zur verwendeten Hardware mehr. Sie lesen und schreiben die Werte der Firmware Services und stellen sie in sensorspezifischen Kontextknoten den Services der Logik Ebene zur Verfügung. Der Vorteil dieser Abstraktion ist die Möglichkeit, Hardware austauschen zu können, ohne dass Änderungen an Prozessen der Logikebene durchgeführt werden müssen. Wird andere Hardware verwendet, können neue Firmware Services erstellt werden, die dann durch die vorhandenen Adaptionsservices zugegriffen werden. Diese laden dann die neuen Informationen in die bereits vorhandenen Kontextknoten, die von den Logikprozessen adressiert werden. Adaptionprozesse können in diesem Modell außerdem mit mehreren Firmwareservices verbunden werden und so redundant auf Informationen zugreifen. Sollte eine Quelle ausfallen, kann der Inhalt einer anderen Quelle im Kontext gespeichert werden.

Services auf der Ebene **Logik** implementieren die Steuerungslogiken der Szenarien. Dazu werden die zuvor festgelegten Algorithmen in diesen Services implementiert. Ein- und Ausgabewerte werden von Services der Adaptionebene bereitgestellt.

Jedes Szenario benötigt einen Service auf der Firmwareebene, der die jeweiligen Sensorwerte ins VSL lädt. Diese werden im Folgenden nicht noch einmal separat vorgestellt. Es werden somit Services ab der Adaptionsebene betrachtet. Zusätzlich zu den Building Blocks wird ebenfalls der jeweils bereitgestellte Kontext angegeben.

4.2.2.1 Beleuchtung

Das Szenario Beleuchtung lässt sich in neben der Firmware in fünf Blöcke einteilen. Diese Einteilung mit den jeweiligen Abhängigkeiten der Blöcke ist in Abbildung 4.3 dargestellt.

Der Bereich **Smart Device Adaption** stellt den Zugriff auf die Sensoren und Aktoren zur Verfügung, die zuvor durch den **Firmware** Service ins VSL geladen wurden. In diesem Szenario sind dies:

- Helligkeitssensor
- Bewegungssensor

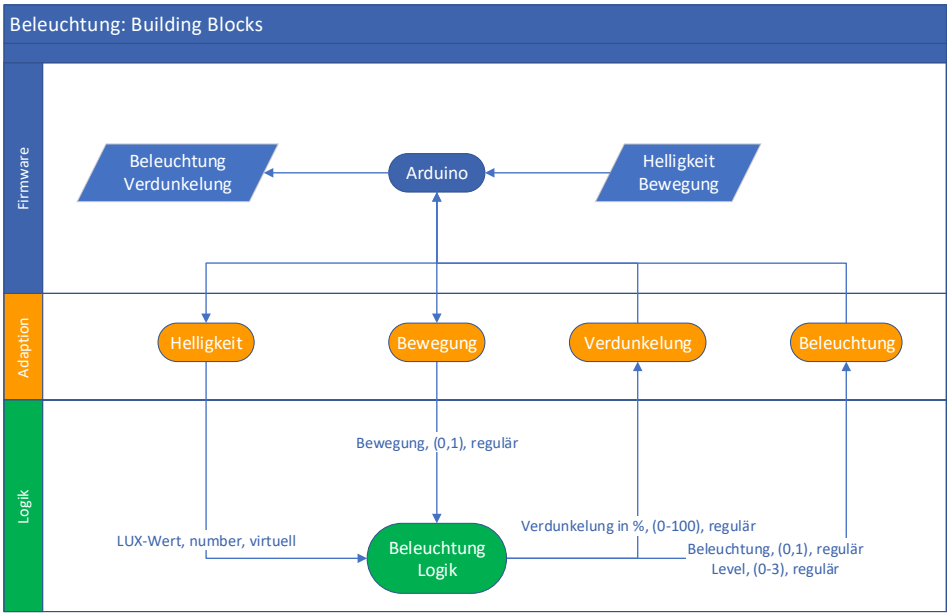


Abbildung 4.3: Beleuchtung Building Blocks

- Verdunkelung (implementiert)
- Beleuchtung

Die erfassten Daten werden in Kontextknoten für andere Services zugänglich gemacht. Der Zugriff auf die **Verdunkelung** wurde bereits in [44] implementiert.

Im Block **Beleuchtung Logik** wird schließlich der Algorithmus aus 2.4.4.1 umgesetzt.

Um den Algorithmus umsetzen zu können wird der in Tabelle 4.1 dargestellte Kontext benötigt.

Tabelle 4.1: Beleuchtung Kontext

Block	Kontext	Knotentyp
Helligkeit	Helligkeit in LUX; number	viruell
Bewegung	Bewegung (Nein/Ja); boolean(0,1)	regulär
Verdunkelung	Geschlossen in %; number(0-100)	regulär
Beleuchtung	Aus/An, Dimmlevel; boolean(0,1) number(0-3)	regulär
Logik	—	–

4.2.2.2 Szenario 2 - Heizung

Die Umsetzung des Szenarios Heizung lässt sich in fünf Building Blocks zusammenfassen, die in Abbildung 4.4 dargestellt sind.

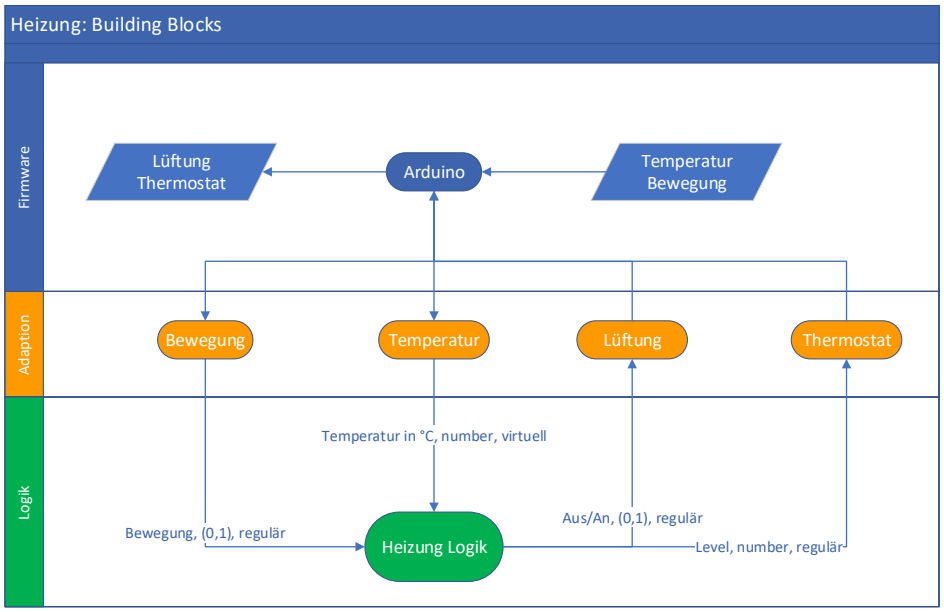


Abbildung 4.4: Heizung Building Blocks

Die **Smart Device Adaption** Services ermöglichen wie zuvor den abstrahierten Zugriff auf die Sensoren und Aktoren. In diesem Szenario auf:

- Bewegungssensor (bereits vorhanden)
- Temperatursensor
- Lüftungssteuerung (implementiert)
- Thermostat

Der Service zur **Lüftungssteuerung** kann aus [44] wiederverwendet werden. Der Zugriff auf den **Bewegungssensor** wurde bereits in 4.2.2.1 modelliert. Der Block **Heizung Logik** beinhaltet die Umsetzung des Algorithmus aus 2.4.4.2, für dessen Umsetzung der in Tabelle 4.2 dargestellte Kontext benötigt wird.

Tabelle 4.2: Heizung Kontext

Block	Kontext	Knotentyp
Bewegung	Bewegung (Nein/Ja); boolean(0,1)	regulär
Temperatur	Temperatur in °C; number	viruell
Lüftung	Aus/An; boolean(0,1)	regulär
Thermostat	Level; number	regulär
Logik	—	–

4.2.2.3 Szenario 3 - Luftqualität

Die Implementierung dieses Szenarios soll eine Überwachung der Luftqualität mithilfe von CO2 und Luftfeuchtigkeitssensoren umfassen. Die dafür benötigten Komponenten sind in Abbildung 4.5 eingezeichnet.

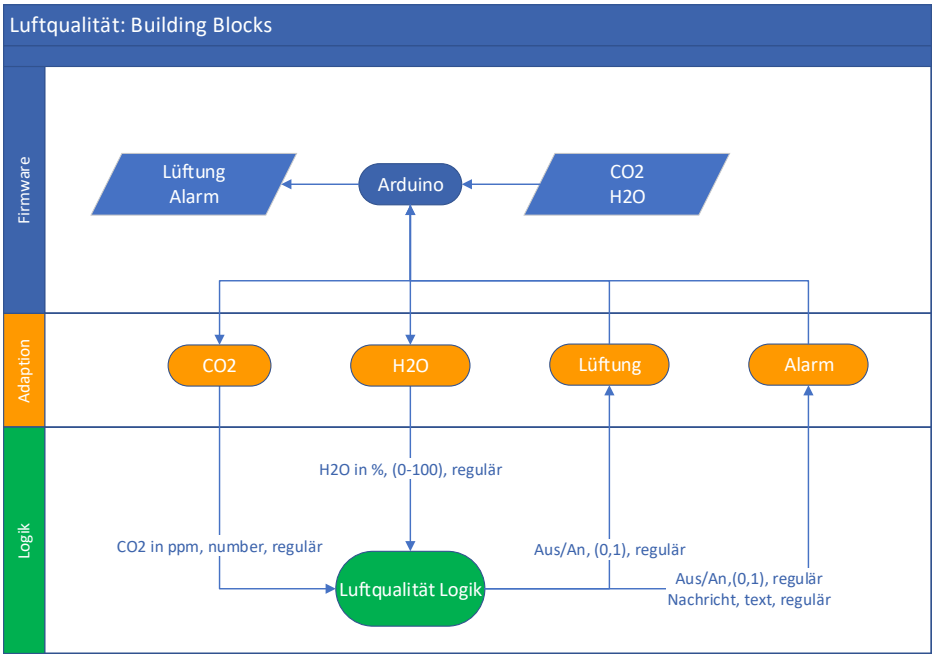


Abbildung 4.5: Luftqualität Building Blocks

In diesem Szenario müssen durch die **Smart Device Adaption** folgende Sensoren und Aktoren eingebunden werden.

- H2O
- CO2
- Lüftungssteuerung (implementiert)
- Alarm (implementiert)

Die Adaptionsservices **Lüftung** und **Alarm** sind bereits in [44] implementiert worden.

Im Block **Luftqualität Logik** wird der Algorithmus aus 2.4.4.3 umgesetzt. Die einzelnen Blöcke müssen dafür den in Tabelle 4.3 gegebenen Kontext bereitstellen.

Tabelle 4.3: Luftqualität Kontext

Block	Kontext	Knotentyp
CO2	CO2 in ppm; number	regulär
H2O	H2O in %; number(0-100)	regulär
Lüftung	Aus/An; boolean(0,1)	regulär
Alarm	Aus/An; boolean(0,1)	regulär
Logik	Alarm, Nachricht; boolean(0,1), text	regulär

4.2.2.4 Szenario 4 - Identifikation

Dieses Szenario soll die Identifikation einer eintretenden Person mittels Feststellung der Körpergröße über einen Ultraschallsensor ermöglichen. Abbildung 4.6 beinhaltet die dafür benötigten Blöcke.

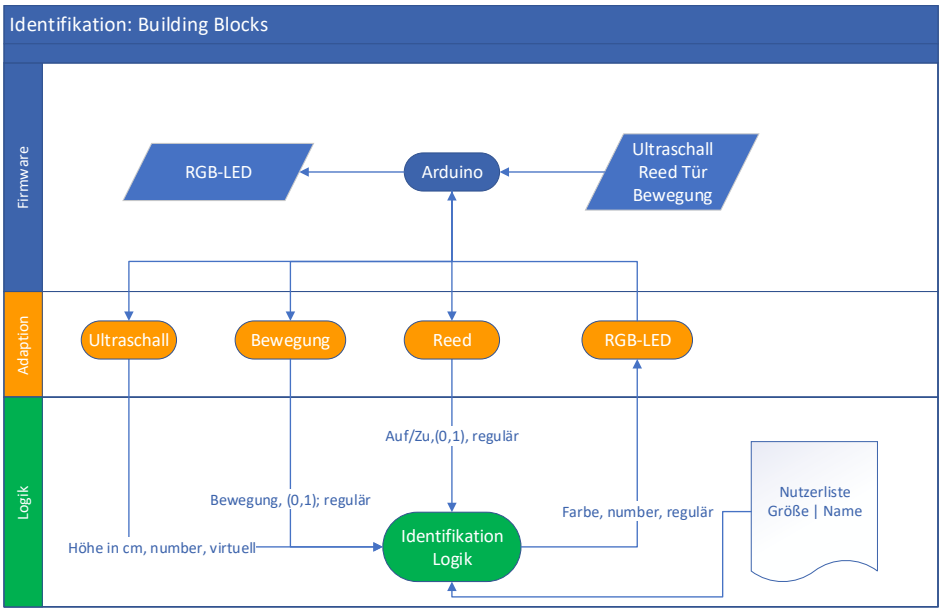


Abbildung 4.6: Identifikation Building Blocks

Für das Szenario Identifikation sind von **Smart Device Adaption** Services einzubinden:

- Ultraschallsensor
- Bewegungssensor (bereits vorhanden)
- Tür Reedsensor
- RGB-LED über Infrarot

Der **Bewegungssensor** Adaptionsservice kann aus 4.2.2.1 wiederverwendet werden.

Der Adaptionsservice für die **RGB-LED** muss Farben und Helligkeit in die entsprechenden Infrarotsignale umwandeln und an die LED-Steuerung senden.

Zusätzlich wird eine **Liste** benötigt, in der Nutzer, ihre Größe und die Lichtfarbe eingespeichert werden, um anhand der gemessenen Werte den Nutzer und die zugehörige Farbe zu ermitteln. Zusätzlich wird ein **Log** angelegt, in dem Nutzer und Zeiten eingetragen werden.

In **Identifikation Logik** befindet sich die Umsetzung des Algorithmus aus 2.4.4.4. Der Kontext der einzelnen Blöcke ist in Tabelle 4.4 gegeben.

Tabelle 4.4: Identifikation Kontext

Block	Kontext	Knotentyp
Ultraschall	Entfernung in cm; number	virtuell
Bewegung	Bewegung (Nein/Ja); boolean(0,1)	regulär
Reed	Geöffnet (Ja/Nein); boolean(0,1)	regulär
RGB-Led	Aus/An, Farbe; boolean(0,1), number	regulär
Logik	Letzte Identifikation; text	regulär

4.3 Erweiterte Szenarien

In diesem Abschnitt erfolgt das Design der beiden Szenarien mit Wearableintegration. Zunächst werden die benötigten Sensoren im Detail vorgestellt. Darauf folgt die Erstellung eines neuen Building Blocks zur Anbindung des Wearables sowie die Vorstellung des benötigten Kontexts. Schließlich folgt die Definition der Testfälle, um den Wearablenutzen zu quantifizieren.

4.3.1 Wearablesensoren

Die Messwerte der folgenden Sensoren werden für die erweiterten Szenarien benötigt.

4.3.1.1 Accelerometer

Die Messwerte des Accelerometers können innerhalb von Android unter den Sensorbezeichnungen *SENSOR_TYPE_ACCELEROMETER* zugegriffen werden. Die Beschleunigungswerte werden in $\frac{m}{s^2}$ ausgegeben. Die Ausgabe erfolgt als Tripel mit den jeweiligen Werten der X,Y,Z-Achsen. Die Werte beinhalten die Erdbeschleunigung von $9{,}81\frac{m}{s^2}$. Die Orientierung der Achsen bei Androidgeräten ist in Abbildung 4.7 dargestellt.

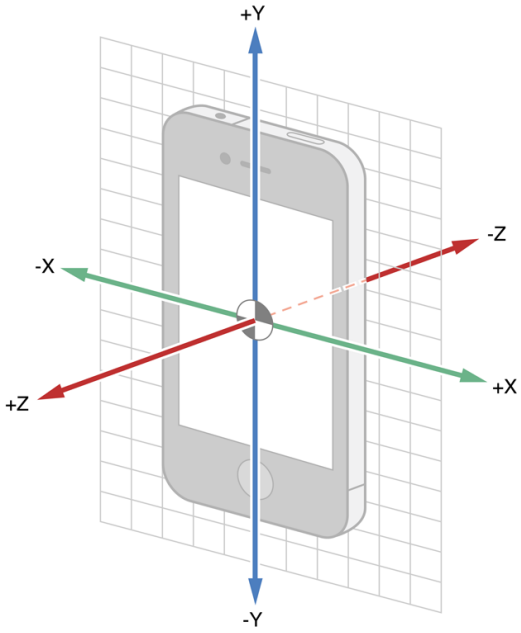


Abbildung 4.7: Achsen Accelerometer [10]

4.3.1.2 Gyroskop

Unter dem Label *SENSOR_TYPE_GYROSCOPE* wird auf die Werte des Gyroskops zugegriffen. Die Neigungsänderung wird in der Einheit $\frac{rad}{s}$ ausgegeben. Die Ausgabe erfolgt ebenfalls als Tripel. Die Drehachsen sind in Abbildung 4.8 eingetragen.

4.3.1.3 Lineare Beschleunigung

Unter dem Label *SENSOR_TYPE_LINEAR_ACCELERATION* kann auf die lineare Beschleunigung des Geräts zugegriffen werden. Der Einfluss der Erdbeschleunigung wird unter Zuhilfenahme des Gyroskops entfernt. Sollte kein Gyroskop vorhanden sein, wird der Kompass verwendet. Bei Ruhelage des Geräts wird auf allen Achsen der Wert 0 gemessen.

Die Werte werden wiederum als Tripel für die X,Y,Z-Achse in der Einheit $\frac{m}{s^2}$ ausgegeben. Die Achsen entsprechen denen des Accelerometers (Abb. 4.7)

Ist weder Gyroskop noch Kompass vorhanden, steht der Sensor nicht zur Verfügung (Ausgabewert 0).

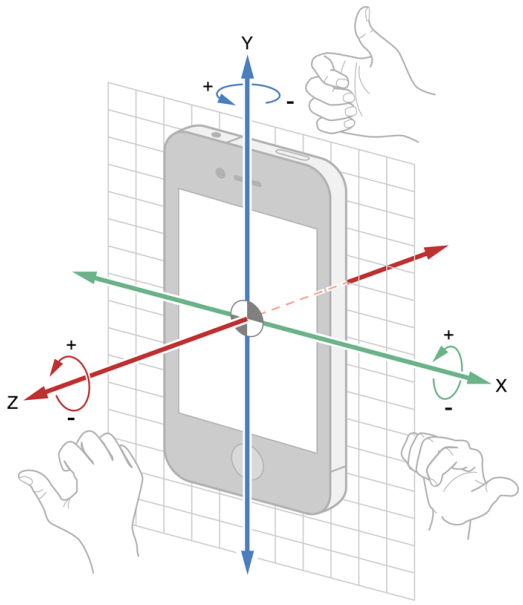


Abbildung 4.8: Achsen Gyroskop [10]

4.3.1.4 Helligkeit

Die Helligkeit kann innerhalb von Android unter dem Label *SENSOR_TYPE_LIGHT* zugegriffen werden. Der Wert wird in der Einheit Lux ausgegeben.

4.3.2 Zugriff

Um Zugriff auf die Sensordaten zu erhalten, muss eine Applikation für Android geschrieben werden, da nur ein nativ ausgeführtes Programm Zugriff auf die Hardware des Geräts anfordern kann. Die App soll es ermöglichen, über eine REST-API auf die Sensorwerte zuzugreifen.

Für die Kommunikation mit der Android App wird wiederum ein Adaptionsservice benötigt. Dieser muss die Werte anfordern und die Rückgabe in VSL Kontextknoten speichern können, um diese zugreifbar für die anderen Services zu machen.

4.3.3 VSL Connector

Die Anbindung des Wearables soll über einen zentralen Service laufen, um kumulative Anfragen zu erlauben und so die Anzahl der Anfragen und der daraus resultierenden Netzwerkpakete zu minimieren. Mithilfe des **VSL Connector** Services soll eine TCP-Verbindung zum Wearable aufgebaut werden, um Sensorwerte nach dem REST-Prinzip anzufordern und die Rückgaben im VSL zu speichern.

Beleuchtung

Für dieses Szenario muss der VSL Connector die Lux Messwerte des mobilen Sensors ins VSL laden. In Abbildung 4.9 sind die Building Blocks des erweiterten Szenarios eingezeichnet. Die neuen Wearable-Blöcke sind rot umrandet.

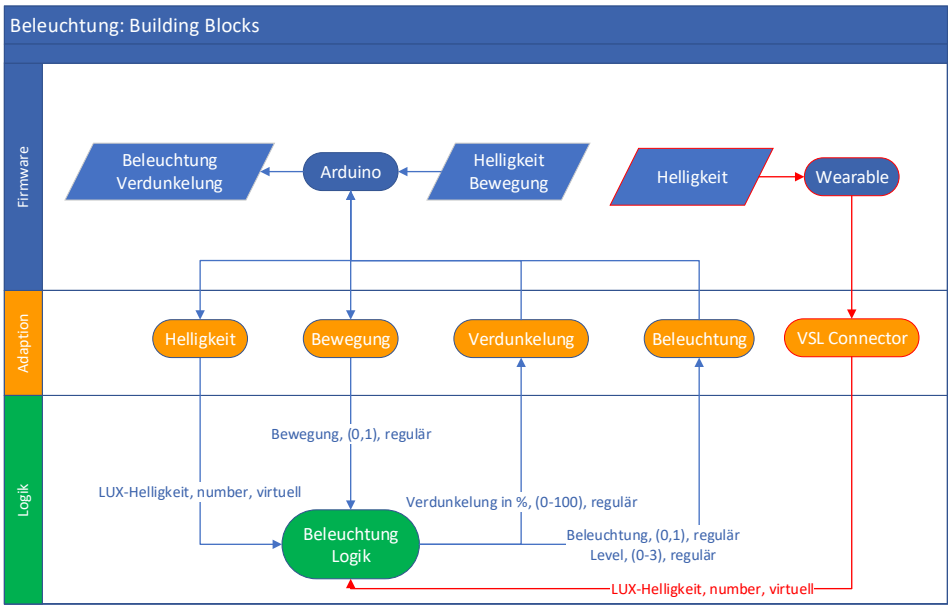


Abbildung 4.9: Erweiterung Beleuchtung Building Blocks

Identifikation

Für das Szenario Identifikation muss der VSL Connector die Werte von Accelerometer, Gyroskop und linearer Beschleunigung auf den einzelnen Bewegungsachsen bereitstellen. Die grafische Repräsentation des neuen Building Blocks ist in Abbildung 4.10 gegeben. Die neuen Wearable-Blöcke sind wiederum rot umrandet.

4.3.3.1 Kontextmodell

Das Kontextmodell des VSL Connector muss auf Basis der betrachteten Szenarien die in Tabelle 4.5 dargestellten Elemente umfassen.

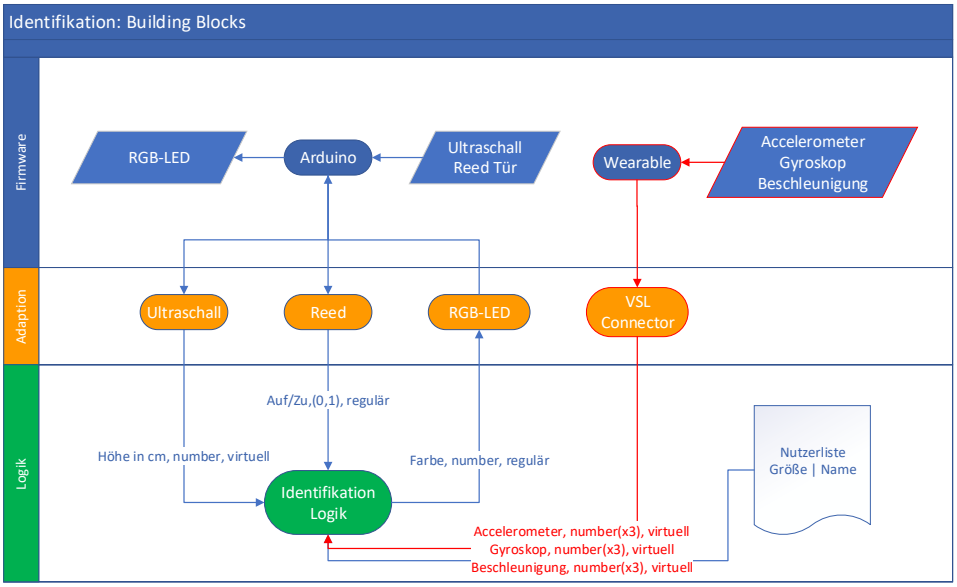


Abbildung 4.10: Erweiterung Identifikation Building Blocks

4.3.4 Testfall I

Mit dem Testfall I soll der Wearablenutzen im Szenario Beleuchtung evaluiert werden. Zu diesem Zweck werden Messungen mit einem Testnutzer durchgeführt, der mit einem tragbaren Luxsensor ausgestattet ist. So sollen Messwerte unter realen Bedingungen erfasst werden.

Aufbau

Der Testaufbau von Testfall I ist in Abbildung 4.11 dargestellt. Zur Bewertung des KPI *Helligkeit im Arbeitsbereich* des erweiterten Beleuchtungsszenarios werden zunächst Sensoren am optimal auszuleuchtenden Ort benötigt (LUX1, LUX2). Im konkreten Fall ist das die Tastatur des Nutzers. Diese Sensoren dienen dazu, eine Ground Truth zu erzeugen, also den IST-Wert im Arbeitsbereich zu messen. Es werden zwei Sensoren an beiden Enden der Tastatur verwendet, da so die Helligkeit im gesamten Arbeitsbereich, vor Allem bei schrägem Lichteinfall (vgl. Abb. 4.11), genauer erfasst werden kann. Um den Nutzen des Wearables zu evaluieren, werden die Messwerte des integrierten Luxsensors erfasst (WEA) und mit der Ground Truth verglichen. Die Differenz dieser Werte lässt die Evaluation des Wearables zu, indem sie den Messfehler durch die Abweichung zur Ground Truth quantifiziert. Um einen Vergleich zum ursprünglichen Szenario herzustellen, werden die Werte des stationären Luxsensors (LUX3) ebenfalls erfasst. Dieser ist zunächst in der Mitte der gesamten Arbeitsfläche positioniert. Der

Tabelle 4.5: VSL Connector Kontext

Kontext	Knotentyp
Helligkeit in Lux, number,	virtuell
Accelerometer X-Achse, number	virtuell
Accelerometer Y-Achse, number	virtuell
Accelerometer Z-Achse, number	virtuell
Gyroskop X-Achse, number	virtuell
Gyroskop Y-Achse, number	virtuell
Gyroskop Z-Achse, number	virtuell
Beschleunigung X-Achse, number	virtuell
Beschleunigung Y-Achse, number	virtuell
Beschleunigung Z-Achse, number	virtuell

Messfehler des stationären Sensors wird ebenfalls durch die Abweichung zur Ground Truth quantifiziert. Um den Nutzen des Wearables evaluieren zu können, werden nach Erfassung der Daten die Messfehler der des Wearables sowie des stationären Sensors miteinander verglichen.

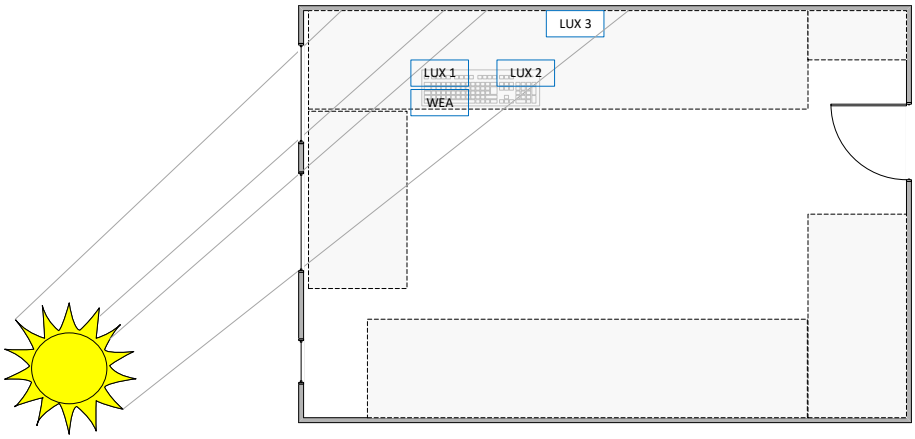


Abbildung 4.11: Testaufbau Beleuchtung

Orientierung des Wearable Luxmeters

Ein Aspekt, der bei der Nutzung des Wearablesensors berücksichtigt werden muss, ist dessen Ausrichtung zur Lichtquelle. Da sich der Sensor am Arm des Testnutzers befindet, ist er nicht konstant horizontal ausgerichtet. Daraus resultieren Abweichungen in den Messwerten, da der gemessene Wert abhängig vom Einfallswinkel des Lichts ist. Aus diesem Grund werden zusätzlich die Messwerte des Accelerometers aufgezeichnet, da diese Aufschluss über die Ausrichtung des Wearables geben. Mithilfe der Lageinformationen besteht die Chance, die Genauigkeit der Luxwerte zu erhöhen, indem ein Faktor für die Neigung einberechnet wird.

4.3.4.1 Erwartete Ergebnisse

Vor allem bei Sonneneinstrahlung sind große Differenzen zwischen stationärem und Ground Truth Sensor zu erwarten, bedingt durch den relativ großen Abstand zueinander. Die Differenz zwischen Ground Truth Sensor und Wearable Sensor sollte geringer ausfallen, da deren Abstand geringer ist. Jedoch ist ebenfalls zu erwarten, dass die Neigung des Wearables zu Schwankungen in den Messwerten und der Differenz zur Ground Truth führt. Je nach Neigung sollten die Werte bei Neigung in Richtung des Fensters nach oben und in Richtung der inneren Wand nach unten abweichen. Wie stark diese Abweichung ausgeprägt sein wird, kann jedoch vor Beginn des Experiments nicht prognostiziert werden. Diese Abweichung wird maßgeblich für die Bewertung des Wearables in diesem Szenario sein, da bei optimaler Ausrichtung des Wearables Werte nahe der Ground Truth zu erwarten sind.

4.3.4.2 Ergebnis I

Die Messungen zur Genauigkeit der Wearabledaten haben im ersten Versuch gezeigt, dass das Wearable bedingt durch die Bewegung des Luxsensors viele Störungen aufweist (Abb. 4.12, WEA). Zusätzlich hat die erste Messung gezeigt, dass die maximalen Messwerte des Wearables auf einen Wert von 10 000 Lux begrenzt sind. Bei direkter Sonneneinstrahlung ist zu erkennen, dass die stationären Ground Truth Sensoren (LUX1, LUX2) Werte von über 24 000 Lux erreichen, das Wearable hingegen den Wert von 10 000 Lux nicht überschreitet (Abb. 4.12, ca. 4000sek). Abgesehen von vorhandenen Störungen folgt das Wearable jedoch dem Verlauf der Ground Truth Sensoren. Im Vergleich zum in der Mitte der Arbeitsfläche montierten stationären Sensor (LUX3) ist die Differenz zum Ground Truth Sensor durch das Wearable größer. Bei genauerer Betrachtung der Installation des stationären Luxsensors im Raum ist dessen Platzierung jedoch unrealistisch. Bei festen Installationen ist die Platzierung auf beweglichen Möbeln unpraktikabel. Zusätzlich könnte ein stationärer Sensor im Tischbereich vom Nutzer oder durch Gegenstände verdeckt werden. Daher wird der stationäre Sensor an die Decke verlegt, da dies einer Verlegung in einem stationären Szenario unter realen Umständen eher entspricht.

Vor erneuter Durchführung des Tests mit den geänderten Parametern wird jedoch zunächst überprüft, ob die Berücksichtigung der Accelerometermesswerte die Genauigkeit der Luxmessungen erhöht. Dazu werden die Daten der Y-Achse betrachtet. Sind die Messwerte dieser Achse bei 0, befindet sich das Wearable in der Horizontalen. Einige Versuche mit verschiedenen Zahlenwerten haben gezeigt, dass aus dem Messwert der Neigung ein Faktor gebildet werden kann, mit dem Messwerte des Luxmeters skaliert werden können. Die Ergebnisse sind in Abbildung 4.13 eingezeichnet. Im Unterschied zu den Rohdaten des Wearable Luxmeters ist eine deutliche Annäherung an die Werte der Ground Truth Sensoren erkennbar. Zusätzlich zu der Skalierung wurde eine Glät-

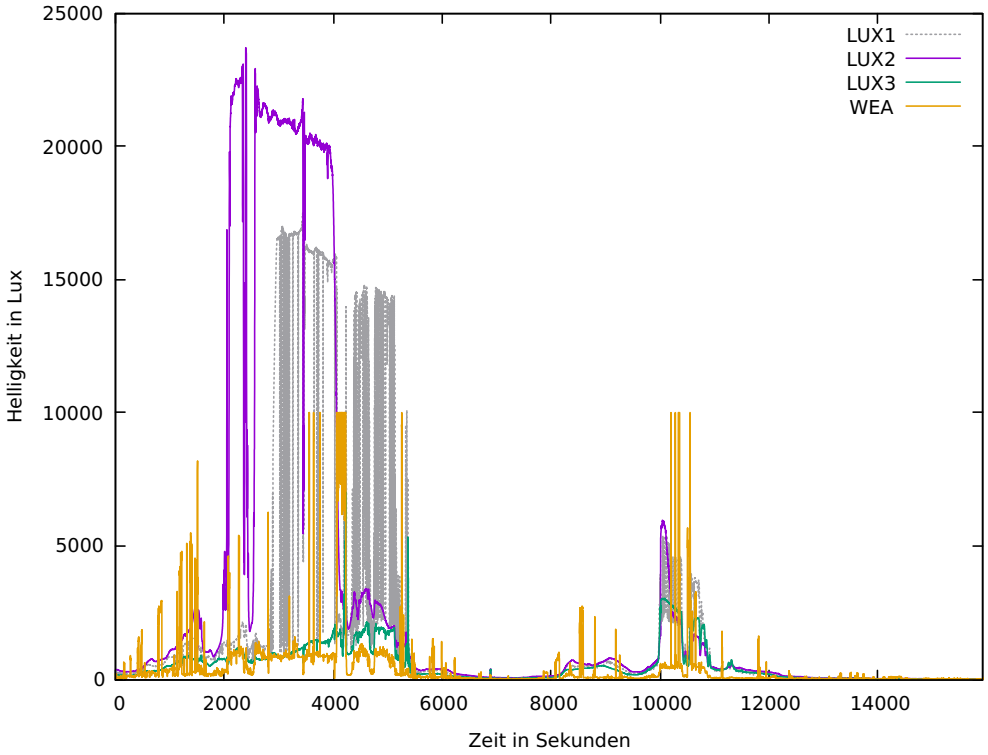


Abbildung 4.12: Ergebnisse Testfall I

tung durchgeführt, um die Störungen durch Bewegungen zu kompensieren. Für diese Glättung wird ein Mittelwert sämtlicher Messwerte der letzten Minute (sekündliche Erfassung → 60 Werte) gebildet.

Da eine deutliche Verbesserung der Genauigkeit erkennbar ist, werden folgende Schritte zur Optimierung durchgeführt:

- Glättung über die letzten 60 Messwerte
- Skalierung mit dem Werte der Accelerometer Y-Achse nach der Formel: $Luxwert * 0.35 * \frac{9.81}{9.81 - |Accelero\_Y|}$

Formel

Bei Erstellung der Formel wurde zunächst die Bruchzahl gebildet. Diese drückt das Verhältnis der Neigung aus. In der Horizontalen nimmt der Wert des Accelerometers auf der Y-Achse den Wert 0 an. In diesem Fall evaluiert der Bruch zum Wert 1. Je stärker das Wearable geneigt wird, desto höhere Werte nimmt der Bruch an. Ein Maximum wird jeweils erreicht, wenn das Wearable um 90 Grad zur Horizontalen geneigt wird. Der Faktor 0.35 wurde via Trial and Error-Versuchen als Wert gefunden, der eine gute Annäherung an die Ground Truth Werte bietet.

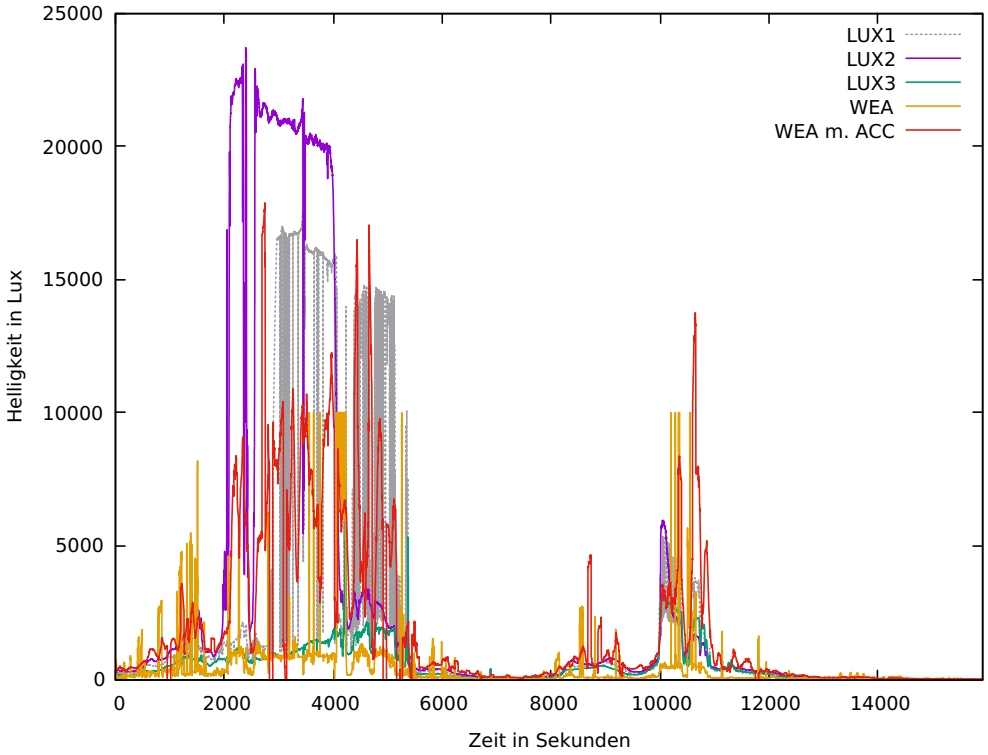


Abbildung 4.13: Ergebnisse Testfall I mit skaliertem Wearable

Das Kappen der Werte über 10 000 Lux beim Wearable Luxmeter kann nicht umgangen werden. Messwerte, die in diesen Bereich fallen, sind daher für die Auswertung der Genauigkeit auszuschließen. Da später der Beleuchtungsalgorithmus mit den Daten gesteuert wird, sollte bei so hohen Werten ohnehin längst die Verdunkelung ausgelöst worden sein. Wichtig ist, wie nah die Luxwerte des Wearables in dunkleren Bereichen bis 1000 Lux am Optimum (Ground Truth) liegen.

Auf Basis der gewonnenen Erkenntnisse wird ein weiterer Test mit leicht veränderten Parametern durchgeführt. Der Luxsensor wird fest im Raum verlegt, an einer Position, die als realistisch für ein stationäres Szenario betrachtet wird. Zusätzlich werden wiederum die Werte des Accelerometers erfasst. Als weiterer Test wird ein direkter Vergleich zwischen stationärem und Wearable Luxsensor durchgeführt. Der Test soll feststellen, ob bei gleicher Ausrichtung zur Lichtquelle die Messwerte abweichen. Es wird sich zeigen, ob beide Sensoren verschiedene Empfindlichkeiten aufweisen, die für Abweichungen in den Ergebnisse verantwortlich sein können. Bei diesem Test werden mithilfe einer dimmbaren Lampe beide Sensoren belichtet. Der Test findet in einem verdunkelten Raum statt, um Störungen durch weitere Lichtquellen auszuschließen.

4.3.4.3 Erwartete Ergebnisse I.b

Zu erwarten ist, dass die Genauigkeit des stationären Sensors abnimmt, da die Entfernung zum Ground Truth Sensor größer wird. Zusätzlich werden die weiteren Testdaten zeigen, ob die Skalierung weiterhin für genauere Ergebnisse sorgt. Der zusätzliche Vergleichstest der Sensoren soll noch einmal sicherstellen, dass beide Sensoren bei identischem Abstand und Ausrichtung zur Lichtquelle dieselben Werte liefern.

4.3.4.4 Ergebnisse I.b

Zunächst wurde der Vergleichstest der Sensoren durchgeführt. Die Messwerte in Abbildung 4.14 zeigen, dass beide Sensoren in etwa dieselbe Empfindlichkeit aufweisen. Um Abweichungen durch individuelle Messfehler der stationären Sensoren auszuschließen, wurde der Test mit einem zweiten stationären Sensor wiederholt. Die zweite Messung kommt jedoch zum selben Schluss. Ab einem Wert von 3500 Lux neigt der Wearablesensor zu Fehlmessungen. Daher nimmt die Genauigkeit ab diesem Wert deutlich ab. Diese Werte sollen bei der Beleuchtungssteuerung aber ohnehin vermieden werden.

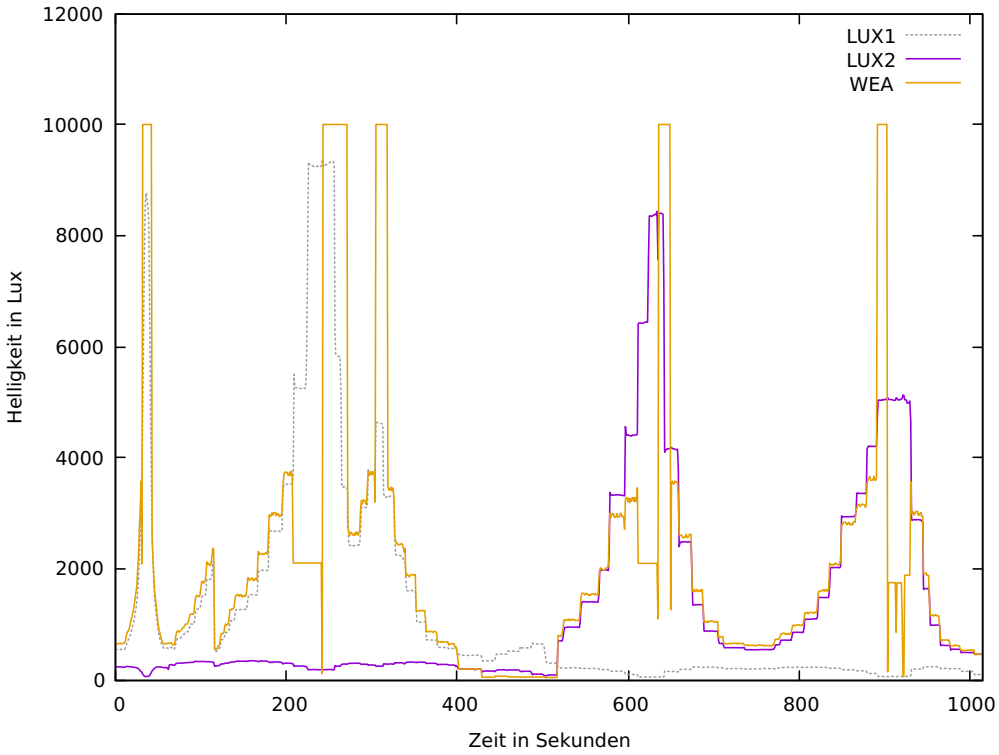


Abbildung 4.14: Ergebnisse Vergleich Luxsensoren

Bei Betrachtung der neuen Testwerte in Abbildung 4.15 zeigen sich die erwarteten Entwicklungen. Die Genauigkeit des stationären Luxsensors fällt sogar in etwa auf das

Niveau des unskalierten Wearables ab. Somit ist das Wearable ohne eine Skalierung fast so genau wie der fest im Raum installierte Luxsensor. Durch die Skalierung, die bereits im ersten Test angewendet wurde, zeigen sich auch bei den neuen Testdaten deutliche Annäherungen der Wearablemessungen an die Werte der Ground Truth Sensoren.

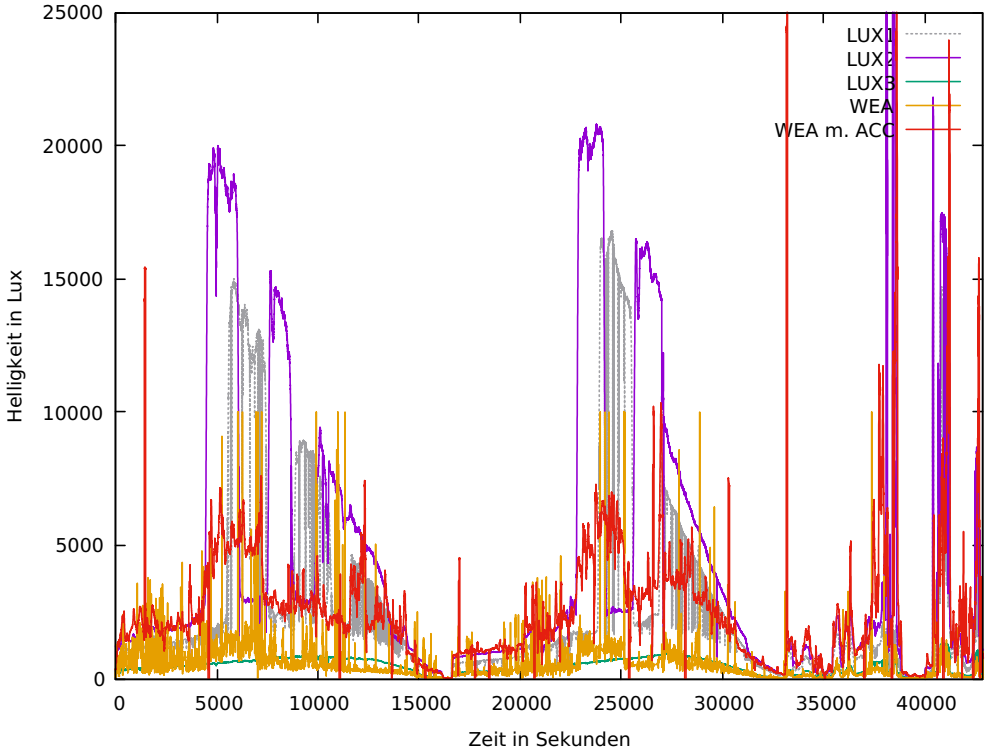


Abbildung 4.15: Ergebnisse Testfall I.b mit skaliertem WEA

4.3.5 Testfall II

Um den in Abschnitt 2.5.1.4 festgelegten KPI *Genauigkeit* auszuwerten, wird ein Testfall benötigt, in dem Werte aus stationärem und Wearable Szenario aufgezeichnet werden. Da jedes Passieren der Tür individuelle Abweichungen aufweist, müssen die Messwerte beider Verfahren gleichzeitig erhoben werden, um Vergleichbarkeit zu gewährleisten.

In diesem Testfall wird ein Testnutzer mit dem Wearable ausgerüstet und passiert wiederholt die zu überwachende Tür. Es werden sowohl die Daten der stationären Sensoren, als auch die des Wearables aufgezeichnet. Zusätzlich wird festgehalten, ob der Nutzer den Raum betreten oder verlassen hat, um eine Ground Truth zu erzeugen. Nach Erfassung der Daten kann untersucht werden, ob die Beschleunigungswerte entlang der einzelnen Achsen des Wearables die Unterscheidung der Gesten *Raum betreten* und *Raum verlassen* zulassen. Dazu werden Korrelationen in den Messwerten gesucht, aus denen die Merkmale zur Erkennung der Gesten extrahiert werden können.

4.3.5.1 Aufbau

Aus Gründen der Vergleichbarkeit wurde in Kapitel 2.5.1.4 beschlossen, das stationäre Identifikationsszenario aus [32] um einen stationären Room Tracker zu erweitern. Zu diesem Zweck wurde ein Bewegungssensor auf der Innenseite der Tür angebracht (Abb. 4.16 PIR). Beim Verlassen des Raumes soll dieser zunächst vom Nutzer vorm Öffnen der Tür und somit vor Auslösen des Reedschalters (Abb. 4.16 REED) ausgelöst werden. Beim Betreten wird angenommen, dass der Wert des PIR-Sensors bei Auslösen des Reedschlaters 0 ist. So kann beim Öffnen der Tür geprüft werden, welchen Wert der PIR-Sensor hat und somit eine Aussage über die voraussichtliche Bewegungsrichtung des Nutzers getroffen werden. Mit dieser Erweiterung kann nun ein Vergleich zwischen stationärer Lösung und Nutzung eines Wearables getroffen werden. So können Datensätze aufgezeichnet werden, die daraufhin durch beide Algorithmen ausgewertet werden. Bei der Aufzeichnung dieser Datensätze muss zusätzlich eine Ground Truth erzeugt werden. Das bedeutet, dass separat festgehalten wird, welche Aktion beim jeweiligen Event wirklich ausgeführt wurde. Danach werden die Ergebnisse beider Algorithmen mit der Ground Truth verglichen, um die Anzahl der Fehlerkennungen zu identifizieren.

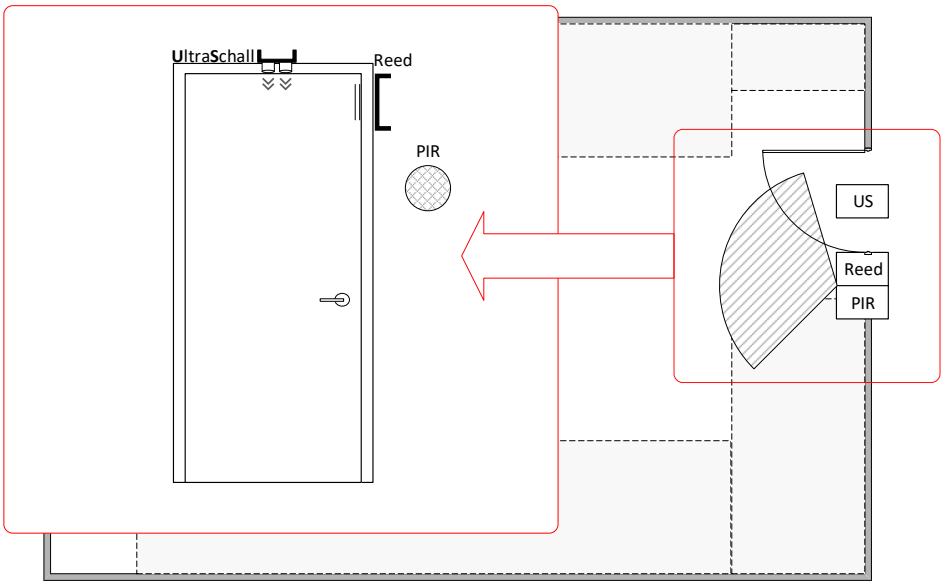


Abbildung 4.16: Testaufbau Identifikation

Die Aufzuzeichnenden Werte in diesem Testfall sind:

- PIR
- Ultraschall

- Reed (implizit, da Messung wenn Reed == 0)
- Lineare Beschleunigung (X-,Y-,Z-Achse)

4.3.5.2 Erwartete Ergebnisse

Zu erwarten ist, dass die stationäre Logik bei einem einzelnen Nutzer relativ zuverlässig funktioniert, da der PIR-Sensor im Raum nicht schon vor Betreten ausgelöst sein sollte. Ist bereits ein Nutzer im Raum und löst den PIR-Sensor aus, wird die stationäre Messung ein falsches Verlassen des Raumes bei Eintritt detektieren. Selbes gilt, wenn der Nutzer nach sehr kurzer Zeit den Raum erneut betritt und der PIR-Sensor noch nicht zurückgesetzt ist. Im Fall des Wearables ist davon auszugehen, dass auf der X-Achse bei Betreten ein Ausschlag in positiver X-Achse durch das „Aufdrücken“ und „Zudrücken“ der Tür mit angewinkeltem Arm oder in negativer Y-Achse durch das Vorwärtsschreiten bei nach unten zeigendem Arm auftritt. Bei Verlassen des Raumes ist zu erwarten, dass ein Ausschlag auf der negativen X-Achse durch das „Aufziehen“ und „Zuziehen“ der Tür auftritt. Zusätzlich ist auf der Y-Achse zunächst ein positiver Ausschlag durch Absenken der Hand nach loslassen des Türgriffs zu erwarten, gefolgt von einem negativen Ausschlag bei erneutem Heben der Hand zum Greifen des äußeren Türgriffs beim Schließen der Tür.

4.3.5.3 Ergebnisse II

Die ersten Ergebnisse der Linearbeschleunigung zeigen zur Unterscheidung der Gesten nur geringe Unterschiede. Abbildung 4.17 zeigt die X-Achse bei Betreten des Raumes. Die erwarteten Ausschläge sind zwar vorhanden, unterscheiden sich jedoch nicht deutlich genug von denen bei Verlassen des Raumes in Abbildung 4.18. Ein ähnliches Bild zeigen die Werte von Y- und Z-Achse, die nicht separat aufgeführt werden. Für eine zweifelsfreie Unterscheidung scheinen die Werte der Linearbeschleunigung demnach nicht ausreichend zu sein. Da diese aus Messwerten von Accelerometer und Gyroskop gebildet werden, wird beschlossen den Versuch zu wiederholen und die Rohdaten dieser beiden Sensoren ebenfalls zu erfassen. Die Idee hinter der Erfassung der zusätzlichen Werte ist, Merkmale zu erfassen, die bei Bildung der linearen Beschleunigung verworfen wurden.

Die Aufzuzeichnenden Werte im Testfall II.b sind:

- PIR
- Ultraschall
- Reed (implizit, da Messung wenn Reed == 0)
- **Accelerometer** (X-,Y-,Z-Achse)

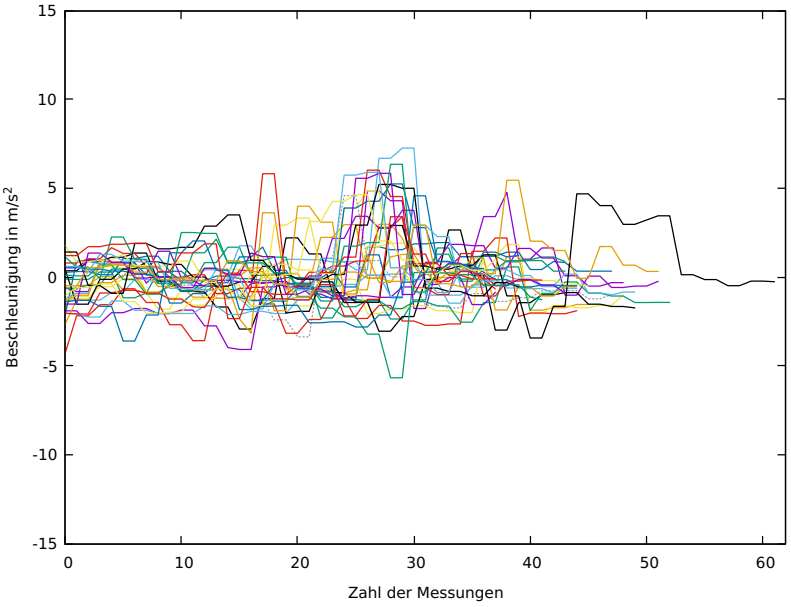


Abbildung 4.17: II Beschleunigung X-Achse, Betreten

- **Gyroskop** (X-,Y-,Z-Achse)
- Lineare Beschleunigung (X-,Y-,Z-Achse)

4.3.5.4 Erwartete Ergebnisse II.b

Die Erwartung in diesem erweiterten Testfall besteht darin, dass die Werte von Accelerometer und Gyroskop eine deutlichere Unterscheidung der Gesten zulassen. Die Annahme besteht darin, dass nun signifikante Merkmale erfasst werden, die bei Berechnung der Linearbeschleunigung aus diesen beiden Werten verworfen werden.

4.3.5.5 Ergebnisse II.b

Nach 50 Samples wurden erste Plots der gemessenen Werte gezeichnet. Sie zeigen vor allem bei den Werten des Accelerometers deutliche Unterschiede zwischen den Gesten. Diese sind bedingt durch die Lage der linken Hand bei Betreten und Verlassen des Raumes. Zusätzlich zur Aufzeichnung der Beschleunigung von Zug und Druck beim Öffnen oder Schließen der Tür wird nun als weiteres Merkmal die Lage der Hand in die Auswertung einbezogen. Die finalen Plots bestehen aus 135 Samples der jeweiligen Aktion Betreten oder Verlassen.

Im Folgenden soll ein einfacher Algorithmus erstellt werden, der anhand von identifizierten Merkmalen eine Unterscheidung der Gesten *Betreten* und *Verlassen* erlauben soll.

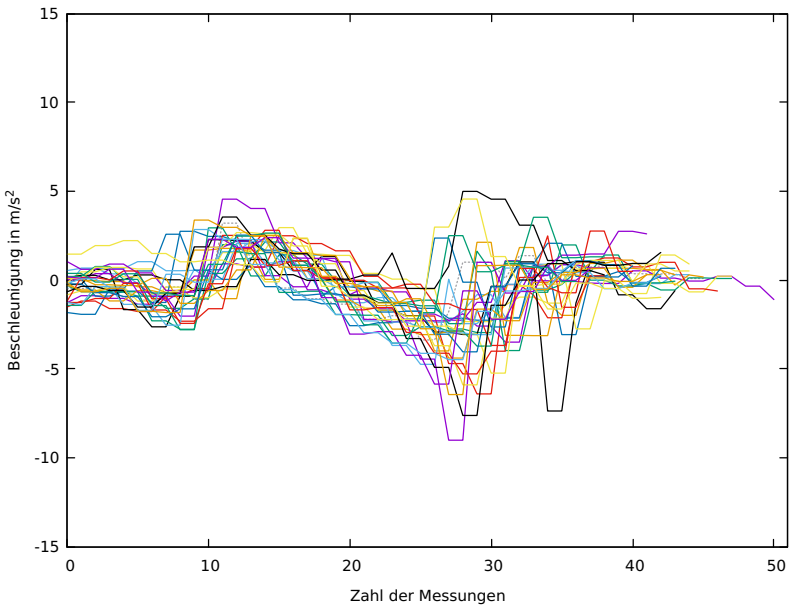


Abbildung 4.18: II Beschleunigung X-Achse, Verlassen

Ist bereits dieser statische Algorithmus in der Lage eine zuverlässige Unterscheidung der Gesten zuzulassen, wird angenommen, dass komplexere Verfahren mindestens dieselbe Erfolgsquote aufweisen werden. Der erstellte Algorithmus soll also als untere Grenze für die Erkennungsraten dienen.

Die ersten Plots mit signifikanten Unterschieden sind die der X-Achse des Accelerometers in Abbildungen 4.19 und 4.20. Zu Beginn der Messungen zeigen sich deutliche Unterschiede zwischen beiden Gesten. Im Bereich der ersten 10 Messpunkte befindet sich ein Großteil der Werte beim Betreten zwischen -10 bis -8.5 (Abb. 4.19). Im Gegensatz dazu befinden sich die Werte beim Verlassen zwischen -8 und -3 (Abb. 4.20).

Als erstes Merkmal für einen einfachen Algorithmus zur Unterscheidung werden demnach festgehalten:

Mittelwert der ersten 5 Messpunkte der Accelerometer X-Achse:

- -3 > Mittelwert > -8.5 → Verlassen
- sonst → Betreten

Weitere Plots, die Merkmale zur Unterscheidung der Gesten enthalten, sind die der Gyroskop Y-Achse. In Abbildungen 4.21 und 4.22 ist zu erkennen, dass beim Betreten die Werte zwischen dem 15. und 20. Messpunkt fast ausschließlich positive Werte annehmen. Beim Verlassen hingegen, sind sämtliche Werte in diesem Bereich negativ.

Als weiteres Merkmal für den Algorithmus wird festgehalten:

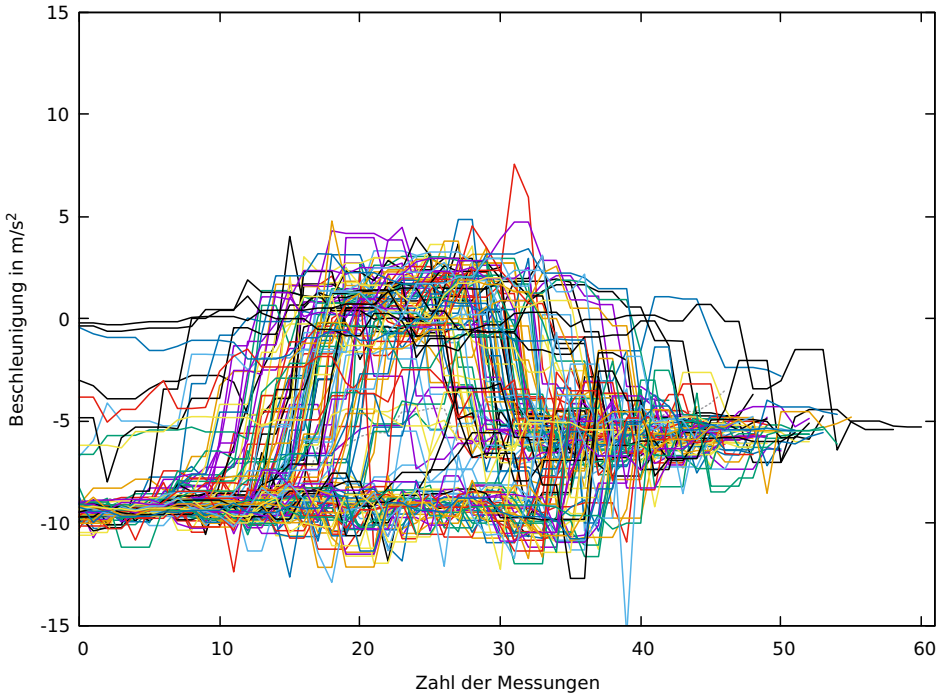


Abbildung 4.19: II.b Accelerometer X-Achse, Betreten

Mittelwert zwischen dem 15. und 20. Messpunkt der Gyroskop Y-Achse:

- Mittelwert < -0.5 → Verlassen
- sonst → Betreten

Die Ergebnisse des Algorithmus, der auf Basis der herausgestellten Merkmale erzeugt wurde, werden in Kapitel 6 vorgestellt.

Die Tatsache, dass Messwerte mit nur einem Nutzer aufgezeichnet wurden und daher für weitere Nutzer gegebenenfalls andere Merkmale gültig sein könnten, wird dadurch kompensiert, dass beim Passieren der Tür eine Identifikation des Nutzers stattfindet. Für jeden Nutzer können somit individuelle Merkmale identifiziert werden, die nach Identifikation des Nutzers auf dessen erfasste Telemetrie angewendet werden.

4.3.5.6 Mehrere Nutzer (II.b.2)

Um zu Prüfen, ob die erfassten Merkmale auf weitere Nutzer übertragbar sind, wurde Testfall II.b mit einem zweiten Nutzer durchgeführt. Ein neues Merkmal, dass für diesen Nutzer anhand der Messwerte (Abb. 4.23 und 4.24) identifiziert werden konnte ist:

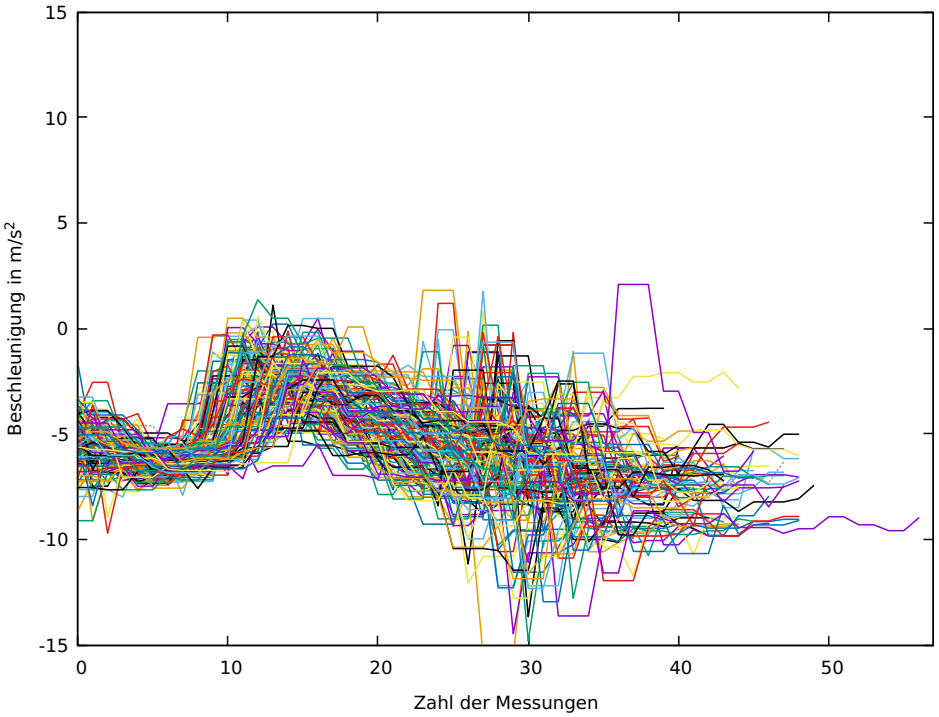


Abbildung 4.20: II.b Accelerometer X-Achse, Verlassen

Mittelwert der Messpunkte 12 bis 20 der Gyroskop Z-Achse:

- Mittelwert > 0.5 → Verlassen
- sonst → Betreten

4.4 Zusammenfassung

Zu Beginn des Kapitels wurden der Cognitive Walkthroug sowie die heuristische Evaluation als Methoden zur DS2OS Evaluation festgelegt. Daraufhin wurde das Design der Szenarien für die Implementierung fertiggestellt. In diesem Schritt wurden einzelne Building Blocks identifiziert. Danach wurden die Testfälle zur Quantifizierung des Wearablenutzens erstellt. Nach ersten Messungen wurden diese Testfälle angepasst, um genauere Testergebnisse erzeugen zu können. Während des Designs wurden zudem die ersten Daten für die DS2OS Evaluation erhoben.

- VSL Design der Szenarien ✓
- Erstellung der Testfälle ✓
- Vorbereitung Softwareevaluation ✓
- Erheben der Daten: Design ✓

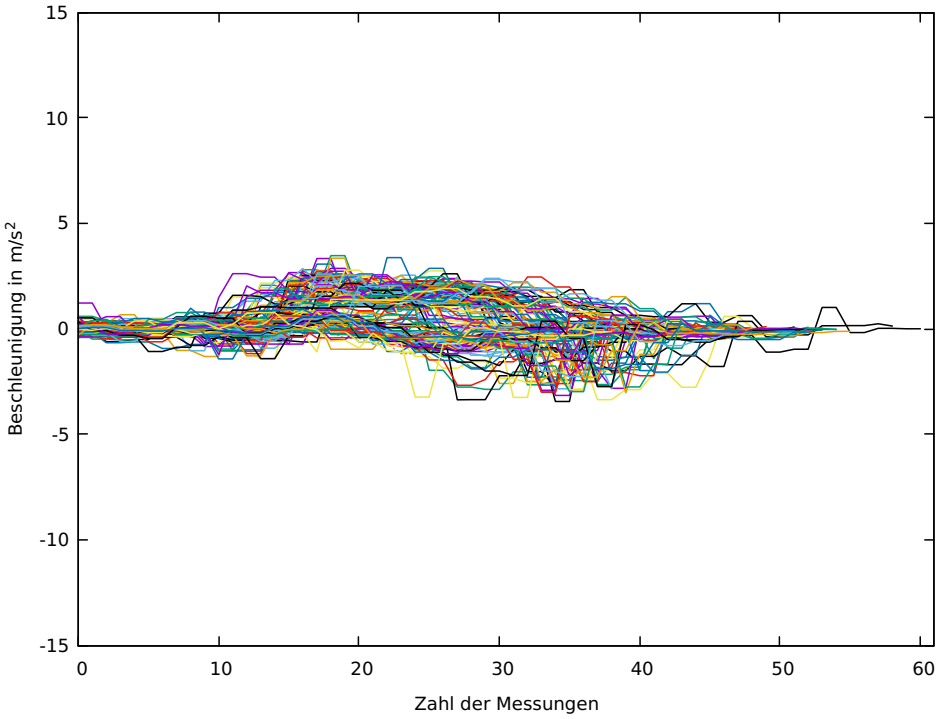


Abbildung 4.21: II.b Gyroskop Y-Achse, Betreten

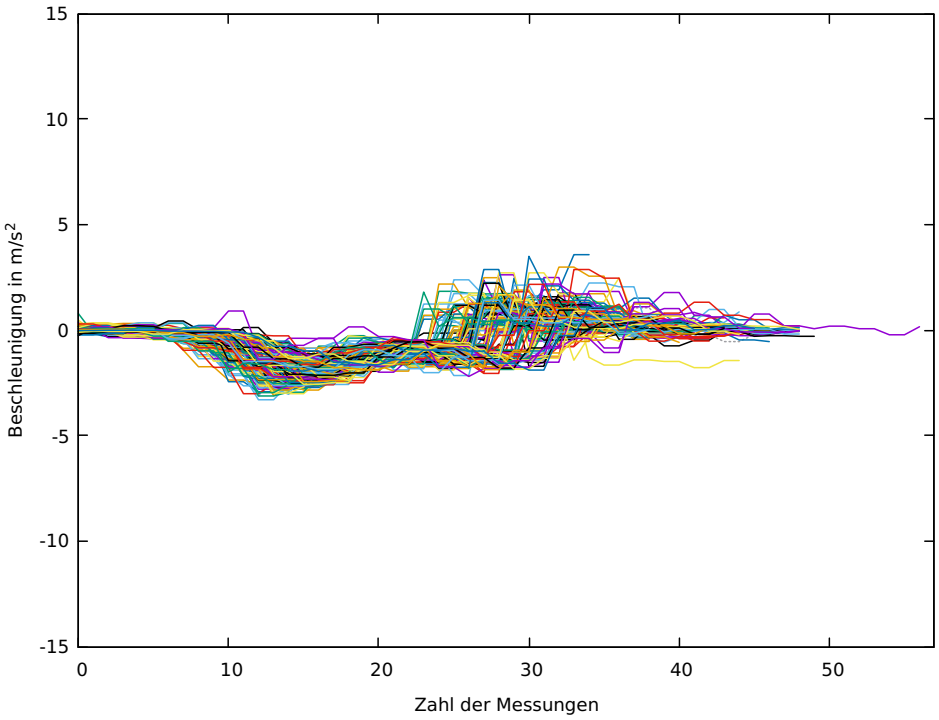


Abbildung 4.22: II.b Gyroskop Y-Achse, Verlassen

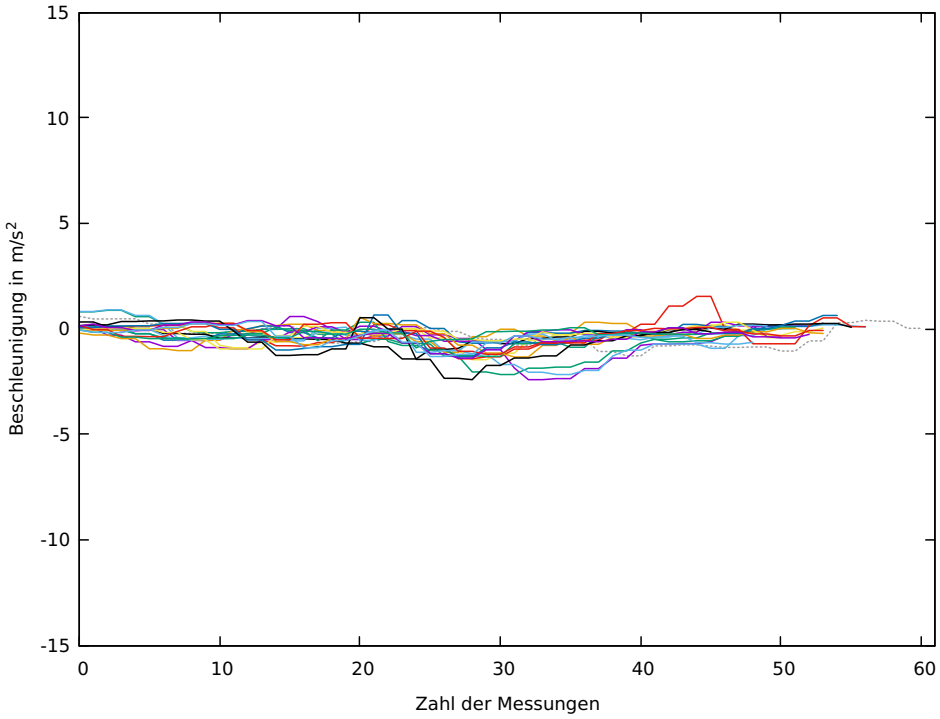


Abbildung 4.23: II.b.2 Gyroskop Z-Achse, Betreten

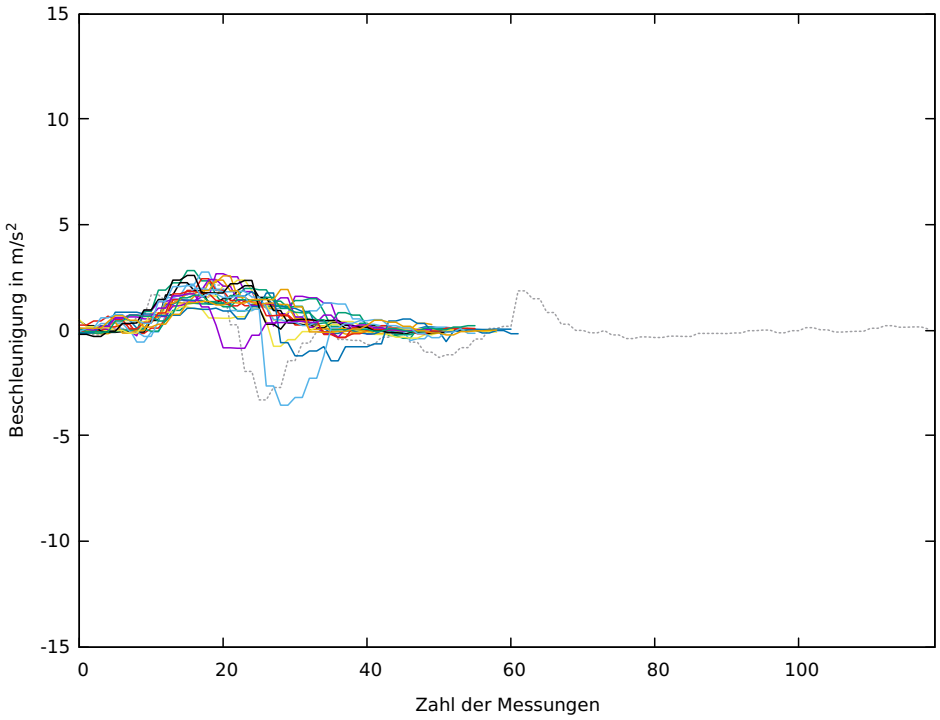


Abbildung 4.24: II.b.2 Gyroskop Z-Achse, Verlassen

Kapitel 5

Implementierung

Dieses Kapitel enthält Details zur Implementierung. Dies sind die Verkabelung der Hardware und die Befehle für den REST-Zugriff auf die Sensordaten. Zusätzlich werden die Topologie nach Implementierung aller Services sowie die Pfade zu den jeweiligen Kontextknoten gegeben.

Am Ende des Kapitels wurden die Meilensteine

- Implementierung der allgemeinen Szenarien ✓
- Implementierung der Wearable Szenarien ✓
- Erheben der Daten: Implementierung ✓

erreicht.

5.1 Hardware

In diesem Schritt erfolgt die Konfiguration der Hardware. Die benötigten Sensoren und Aktoren wurden in Kapitel 4.2.1 vorgestellt. Diese müssen nun an die Arduino angebunden werden.

Beleuchtung—Die Anbindung des BH1750 Luxsensors erfolgt über den I^2C Bus des Arduino. Zu diesem Zweck muss er mit den Signalleitungen für den Takt (SCL / Serial Clock) und die Daten (SDA / Serial Data) verbunden werden. Zusätzlich muss er mit der 3,3V Spannungsversorgung verbunden werden. Der PIR-Sensor (HC-SR501) und der Infrarotsender (KY-005) müssen mit einem freien digitalen Anschluss sowie der 5V Spannungsversorgung verbunden werden (Abb. 5.1). Optional kann der KY-022 Infrarotempfänger zum Einlesen von IR-Codes verwendet werden.

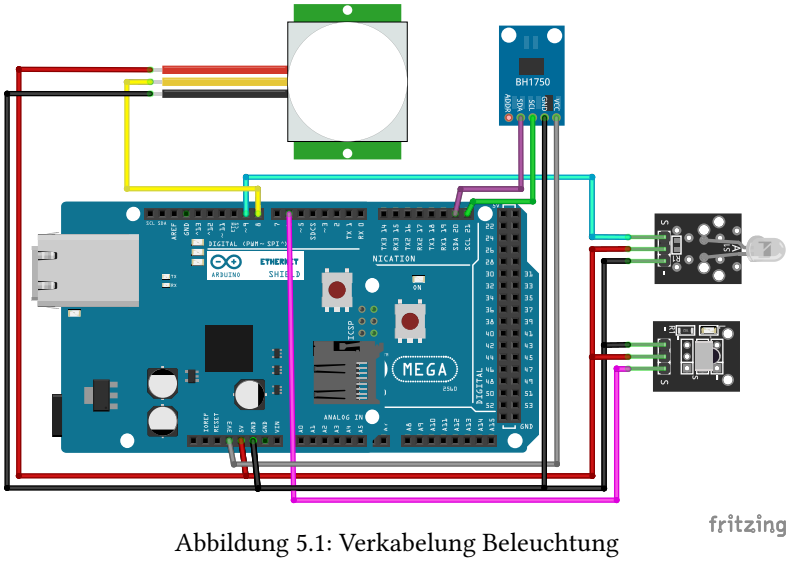


Abbildung 5.1: Verkabelung Beleuchtung

Heizung—Zur Realisierung des Heizungsszenarios müssen der Temperatur- und Luftfeuchtigkeitssensor (KY-015) sowie der HC-SR501 PIR-Sensor jeweils mit einem freien digitalen Port des Arduino verbunden werden (Abb. 5.2).

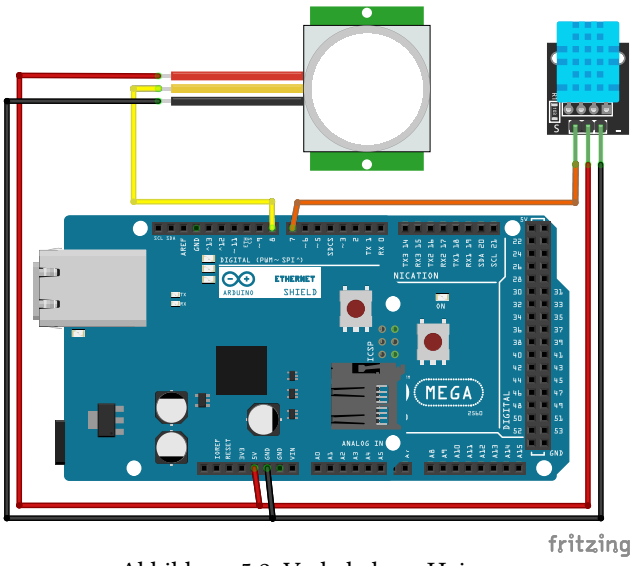


Abbildung 5.2: Verkabelung Heizung

Luftqualität—Der SKU:SEN0159 CO2 Sensor muss mit einem analogen Port des Arduino verbunden werden. Der KY-015 Sensor wird mit einem freien digitalen Port verbunden. Beide müssen wiederum mit der 5V Spannungsversorgung verbunden werden (Abb. 5.3).

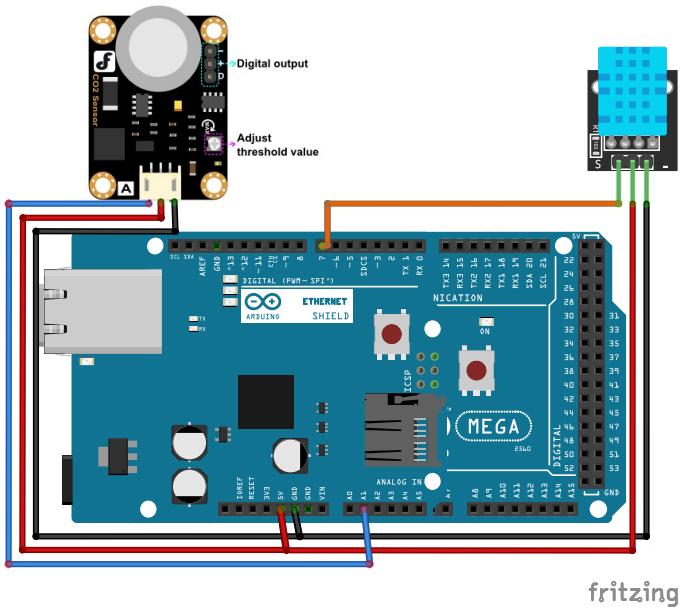


Abbildung 5.3: Verkabelung Luftqualität

Identifikation—Der KY-025 Reedschalter sowie der HC-SR501 PIR-Sensor benötigen einen freien digitalen Anschluss des Arduino. Der Ultraschallsensor HC-SR04 benötigt zwei digitale Ports für das Trigger-Signal (Auslösen der Messung) und das Echo-Signal (Ergebnis der Messung). Alle Sensoren müssen mit 5V Spannung versorgt werden (Abb. 5.4).

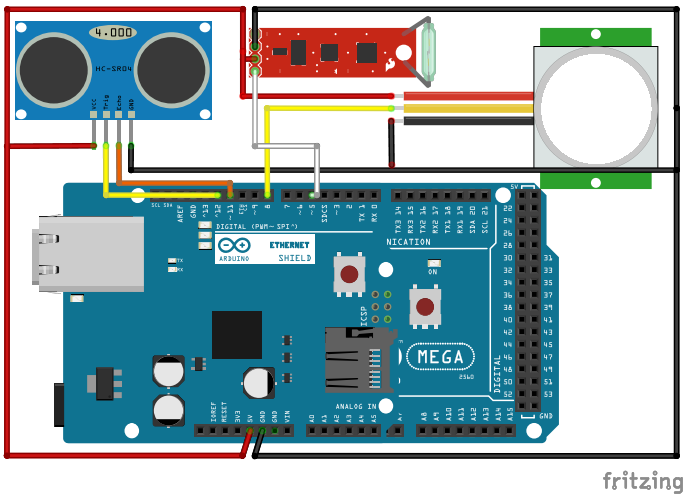


Abbildung 5.4: Verkabelung Identifikation

5.1.1 Gateways

Da alle Szenarien in einem Raum implementiert werden, können die Entitäten verschiedener Szenarien mit einem Arduino verbunden werden. Es wurde entschieden,

insgesamt zwei Arduino zu verwenden. Die Aufteilung der angeschlossenen Entitäten erfolgt nach Sensorik und Aktorik. Dies geschieht aus dem Grund, dass die Werte der Sensoren häufig und periodisch abgefragt werden. Diese Anfragen erfolgen, wie im nächsten Abschnitt begründet wird, kumulativ. Um nicht mit diesen Anfragen zu interferieren, werden die nicht periodisch auftretenden Stellanweisungen an die Aktorik von einem zweiten Arduino behandelt. Bei Implementierung aller Szenarien ist die Verkabelung des ersten Arduino, im Folgenden als **Gateway1** bezeichnet, in Abbildung 5.5 gegeben. Sie umfasst sämtliche benötigte Sensorik.

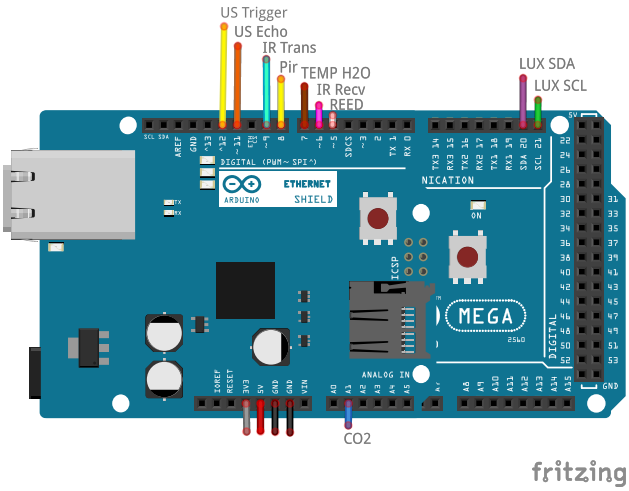


Abbildung 5.5: Verkabelung Gateway 1

Die Verkabelung von **Gateway 2** ist in Abbildung 5.6 dargestellt. Sie umfasst mit dem Infrarotsender den einzigen Aktor, der nicht bereits durch vorhergehende Arbeiten implementiert ist.

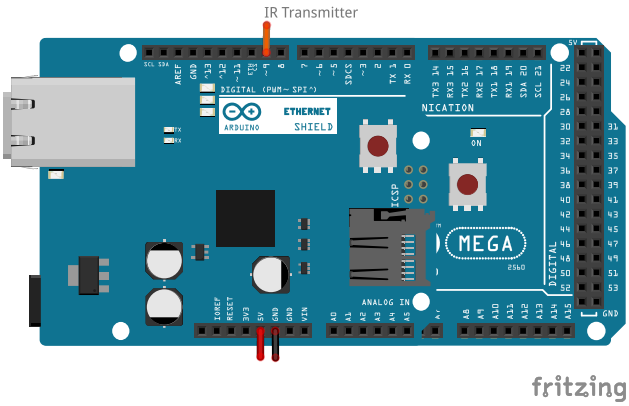


Abbildung 5.6: Verkabelung Gateway 2

5.1.2 Wearable

Das Wearable wird mithilfe eines Android Smarthones simuliert. Es verfügt über alle benötigten Sensoren. Zur Erfassung der Daten wird es am Handgelenk des Testnutzers fixiert (Abbildung 5.7). Die Ausrichtung entspricht der einer Android-Smartwatch, sodass die Achsen von Accelerometer und Gyroskop übertragbare Werte liefern. Es kann dieselbe App wie auf einer Android-Watch verwendet werden. Es muss lediglich eine Anpassung in der Manifest-Datei der App durchgeführt werden, die den Typ des Endgeräts spezifiziert. Beim konkret eingesetzten Gerät handelt es sich um ein LG G2 (D802) mit Android Version 7.1.2.

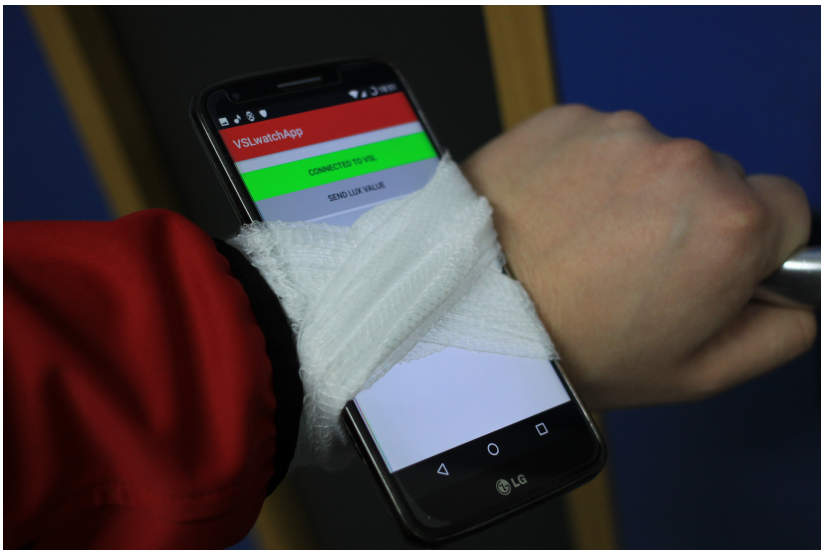


Abbildung 5.7: Befestigung des Android-Geräts

5.2 Firmware

5.2.1 Gateway 1

Die Firmware für das erste Gateway implementiert den Zugriff auf die Sensoren:

- Reed
- PIR
- Ultraschall
- CO2
- Luftfeuchtigkeit
- Temperatur
- Helligkeit

Die URL der jeweiligen Sensoren sind in Tabelle 5.1 gegeben. Zusätzlich ist der Typ (GET/SET) sowie der Rückgabewert eingetragen.

Tabelle 5.1: Gateway 1 Firmware

URL	HTTP Typ	Rückgabewert
/reed	GET	0 ≙ geöffnet, 1 ≙ geschlossen
/pir	GET	0 ≙ keine Bewegung, 1 ≙ Bewegung
/echo	GET	Entfernung in cm
/co2	GET	CO2 in ppm
/h2ot	GET	Temperatur in °C, H2O in %
/lux	GET	Helligkeit in Lux

Im laufenden Betrieb wurde festgestellt, dass es bei häufigem Zugriff auf die einzelnen Sensorwerte (alle 150ms) durch mehrere Services zu Paketverlusten kommt. Daher wurden die Sensoren so gruppiert, dass eine Anfrage für alle benötigten Werte gesendet werden kann, die eine kumulative Antwort zur Folge hat. Die Gruppen sind in Tabelle 5.2 dargestellt. Die Gruppe *Basis* umfasst Sensoren, deren Werte zeitkritisch (mehrfach pro Sekunde) abgerufen werden müssen. Dies sind der Reedsensor zur unmittelbaren Feststellung einer Türöffnung sowie der PIR-Sensor zum Erfassen einer Bewegung. Die Gruppe *Extended* umfasst alle weiteren Sensoren, abgesehen vom Ultraschallsensor. Diese Werte werden in einer größeren Periode abgerufen (jede 50. Anfrage, alle 7,5s). Das Hinzufügen des Ultraschallsensors zur Gruppe *Extended* und der dauerhafte Abruf dieser Gruppe bei geöffneter Tür hatte eine hohe Latenz zur Folge, sodass zu wenige Samples für die Identifikation der eintretenden Person gesammelt werden konnten. Der Ultraschallsensor muss somit separat behandelt werden, da dessen Werte nur bei geöffneter Tür erfasst werden sollen. Für diesen Fall wurde die Gruppe *Ident* erstellt. Bei geöffneter Tür wird ausschließlich diese Gruppe abgerufen. Um weiterhin den Türzustand und Bewegungen erfassen zu können, müssen PIR und Reed eingeschlossen werden.

Tabelle 5.2: Gateway 1 Sensorgruppen

URL	HTTP Typ	Sensoren
/basis	GET	reed;pir
/extended	GET	reed;pir;h2o;temp;co2;lux
/ident	GET	reed;pir;echo

Verwendete Bibliotheken

Einige Sensoren benötigen spezielle Bibliotheken. Die jeweils Genutzten sind:

- HC-SR04 - New Ping [42]
- KY-015 -Adafruit DHT Humidity & Temperature Unified Sensor Library [41]
- BH 1750 - AS_BH1750 [40]

5.2.2 Gateway 2

Die Firmware des zweiten Gateways implementiert den Zugriff auf den Infrarotsender zur Steuerung der LED-Beleuchtung. Die URL für die implementierten Funktionen ist */ir*. Die möglichen Eingabewerte und ihre Bedeutung sind in Tabelle 5.3 gegeben.

Tabelle 5.3: Gateway 2 Firmware

URL	HTTP Typ	Funktion
/ir <ID>	SET	Senden von IR-Codes zur Steuerung der LED-Beleuchtung

Die Bedeutungen des übergebenen *ID* Parameters sind in Tabelle 5.4 gegeben.

Tabelle 5.4: Gateway 2 IR Parameter

ID	Weiss	ID	RGB
0	aus	8	aus
1	an	9	an
2	dunkler	10	dunkler
3	heller	11	heller
4	Helligkeit 25%	12	Farbe Rot
5	Helligkeit 50%	13	Farbe Grün
6	Helligkeit 75%	14	Farbe Blau
7	Helligkeit 100%	15	Farbe Weiss

Verwendete Bibliotheken

- KY-005 - IRremote Arduino Library [43]

5.2.3 Android App

Mithilfe der Android App können Sensorwerte über *http* ausgelesen werden. Die URL mit Typ und Rückgabewert der jeweiligen Sensoren sind in Tabelle 5.5 gegeben.

Tabelle 5.5: Android App

URL	HTTP Typ	Rückgabewert
/lux	GET	Helligkeit in Lux
/accelerometer	GET	Beschleunigung auf X,Y,Z-Achse in $\frac{m}{s^2}$
/gyroscope	GET	Rotationsgeschwindigkeit auf X,Y,Z-Achse in $\frac{rad}{s}$
/acceleration	GET	lineare Beschleunigung auf X,Y,Z-Achse in $\frac{m}{s^2}$
/all	GET	accelerometer,gyroscope,acceleration,lux

Im Abbildung 5.8 ist die Benutzeroberfläche der VSLwatchApp dargestellt. Die Belegungen der drei Tasten sind:

- CONNECTED TO VSL: Verbindung mit VSL herstellen, bei Verbindung grün
- SEND LUX VALUE: Senden des aktuellen Luxwerts
- STOP: Abbau der Verbindung

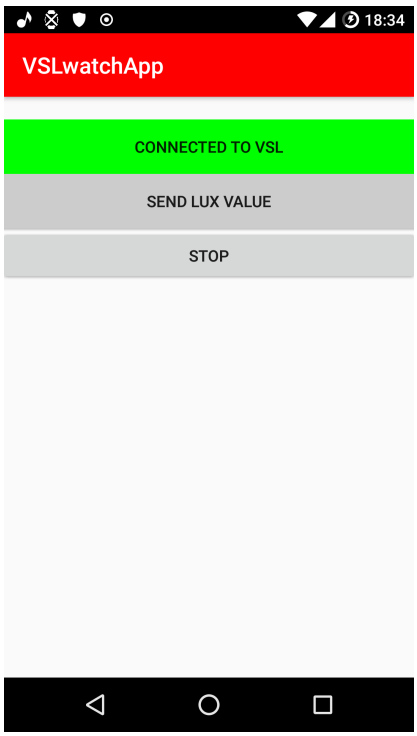


Abbildung 5.8: Screenshot VSLwatchApp

5.2.4 VSL-Services

Die Services *Gateway7* und *Gateway8* kommunizieren mit den Gateways und laden den Kontext ins VSL. Die Werte des Wearables werden über den *AndroidVSL* Service ins VSL geladen.

5.3 Adaption

Die Adaptionsservices der einzelnen Sensoren greifen auf den Kontext der Services *Gateway7* und *Gateway8* zu und stellen diesen unter ihrem jeweiligen Pfad zur Verfügung. Eine Übersicht über die Pfade zum Kontext sämtlicher in dieser Arbeit erstellten Services befindet sich in Appendix C. Eine Topologie, in der die jeweiligen Abhängigkeiten der Services zueinander eingezeichnet ist, befindet sich in Appendix D.

Kapitel 6

Evaluation

In diesem Kapitel werden die Ergebnisse der im Design erstellten Testfälle ausgewertet und die Hypothesen aus der Analyse getestet. Zusätzlich wird das Ergebnis der DS2OS Evaluation vorgestellt. Die in diesem Kapitel gewonnenen Erkenntnisse stellen die Basis für die Beantwortung der Leitfrage dar.

- Auswertung der Testergebnisse
- Testen der Hypothesen

6.1 Beleuchtung

Um die Ergebnisse aus 4.3.4 zu quantifizieren, wird der durchschnittliche Messfehler zwischen Ground Truth und betrachtetem Sensor berechnet. Je geringer dieser ausfällt, desto genauere Werte liefert der jeweilige Sensor.

Um den Fehler zu bestimmen, wurde für jeden Messpunkt die Differenz zwischen den beiden Ground Truth-Sensoren (LUX1, LUX2), die an der Tastatur des Testnutzers platziert sind, sowie Wearable (WEA) und stationärem Sensor (LUX3) berechnet. Diese wurden aufsummiert und am Ende durch die Anzahl der Messpunkte geteilt, um den mittleren Fehler zu erhalten. Bei einem Wert von Null würden die Messwerte der Sensoren der Ground Truth entsprechen. Je höher der Mittelwert jedoch ist, desto größer ist die durchschnittliche Abweichung von der Ground Truth. Ziel ist es, möglichst geringe Werte bei der Abweichung zu erreichen.

Die Abweichung wird für vier Messreihen berechnet. Zunächst werden die Mittelwerte für den stationären und den Wearablesensor berechnet. Um den Grad der Optimierung festzustellen, werden zusätzlich die Werte bei reiner Glättung der Wearablesdaten sowie die Optimierung mittels Skalierung und Glättung betrachtet. Alle Messreihen sind zur Veranschaulichung noch einmal in Abbildung 6.1 dargestellt.

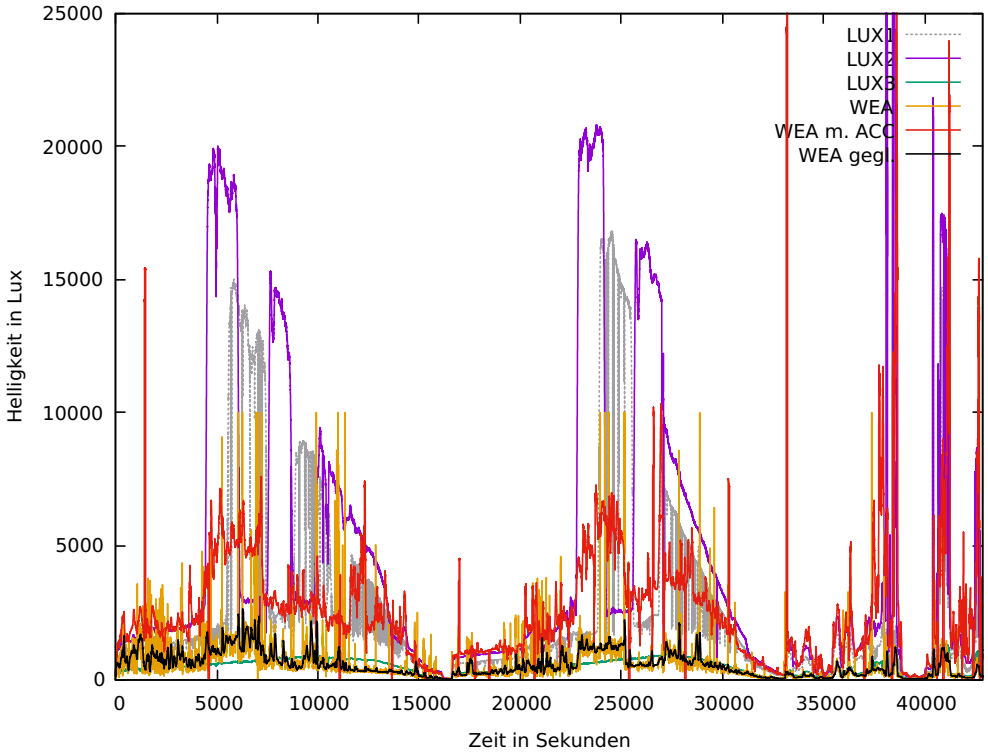


Abbildung 6.1: Ergebnisse Beleuchtung mit Skalierung und Glättung

Bereits bei Betrachtung der Abbildung ist zu vermuten, dass die Genauigkeit des skalierten und geglätteten Wearables am höchsten ist. Die genauen Ergebnisse der mittleren Abweichungen der jeweiligen Messreihen sind in Tabelle 6.1 eingetragen.

Tabelle 6.1: Mittlere Lux Abweichungen

Sensor	Abweichung zu LUX1	Abweichung zu LUX2
Stationär (LUX3)	2466	3980
Wearable (WEA)	2424	3935
Wearable geglättet (gegl.)	2394	3903
Wearable skaliert (m. ACC)	1240	1943

Die Ergebnisse bestätigen das Bild, dass auch schon Abbildung 6.1 zeigt. Der stationäre Sensor weist schließlich die größten Abweichungen auf. Die Abweichung befindet sich in etwa auf dem Niveau des unbearbeiteten Wearables, dessen Differenz im Mittel um etwa 40 Lux näher an der Ground Truth liegt. Dennoch fällt auf, dass das Wearable relativ viele Störungen aufweist. Dies könnte in ständigen Stellanweisungen an den Beleuchtungsalgorithmus resultieren. Um kurzzeitige Abweichungen auszugleichen, wurden die Werte des Wearables nach der in Abschnitt 4.3.4 beschrieben Methode geglättet. Neben der Beseitigung großer Schwankungen in den Werten wurde die Genauigkeit ebenfalls um etwa 30 Lux im Mittel erhöht. Die größte Änderung weist jedoch die An-

wendung der Skalierung aus 4.3.4 auf. In beiden Fällen konnte der mittlere Messfehler nahezu halbiert werden.

6.2 Identifikation

Auf Basis der in Kapitel 4.3.5 extrahierten Merkmale wurde ein Algorithmus erstellt. Dieser wurde auf den vorhandenen Testdaten ausgeführt, um die Erkennungsraten für die Gesten *Betreten* und *Verlassen* zu ermitteln. Um einen Vergleich mit der stationären Lösung mithilfe von Reed und PIR-Sensor zu erhalten, wurden beide Verfahren gleichzeitig auf denselben Testdaten angewendet. Durch jeden Algorithmus wurden Zuordnungen zu den jeweiligen Gesten vorgenommen. Das jeweilige Ergebnis wurde daraufhin mit dem tatsächlichen Wert der Ground Truth verglichen. Für jede falsche Zuordnung wurde ein jeweiliger Zähler inkrementiert. Unterschieden wurde in fehlerhafte Einordnung als Betreten sowie fehlerhafte Einordnung als Verlassen.

Insgesamt wurden vier separate Tests durchgeführt. Um zusätzlich genauere Informationen über die Genauigkeit der einzelnen Weararablemerkmale zu erhalten, wurden diese jeweils noch einmal einzeln angewendet. Die durchgeführten Tests sind:

- Anwendung des stationären Algorithmus (PIR)
- Anwendung des Merkmals Accelerometer X-Achse
- Anwendung des Merkmals Gyroskop Y-Achse
- Kombination Accelerometer und Gyroskop

Die Ergebnisse sind in Tabelle 6.2 eingetragen. Der Testsatz umfasst jeweils 135 Ereignisse des Typs Betreten sowie Verlassen.

Tabelle 6.2: Algorithmen Ergebnisse

Algorithmus	Falsch-Betreten	Falsch-Verlassen
Stationär	12	30
Accelerometer	0	7
Gyroskop	1	0
Accelero & Gyro	0	0

Die Auswertung zeigt, dass der stationäre Fall die höchsten Quoten bei den Fehlerkennungen aufweist. Von den 135 Ereignissen des Typs Betreten wurden fälschlicherweise 30 als Verlassen gedeutet. Das entspricht einer Fehlerrate von etwa 22%. Beim Ereignis Verlassen wurden 12 fälschlich als Betreten gedeutet, was einer Fehlerrate von etwa 9% entspricht. Grund für das falsch-Betreten kann sein, dass beim Verlassen keine Bewegung durch den PIR-Sensor erfasst wurde. Dieser könnte empfindlicher gestellt werden,

was vor allem bei mehreren Nutzern im Raum negative Folgen durch fälschliches Auslösen hätte. Alternativ könnte die Zeitspanne, die der Sensor den Pegel bei Erkennung hält, verlängert werden. Wird diese Zeitspanne jedoch verlängert, kann das Auswirkungen auf die falsch-Verlassen Erkennung haben. Es würde die Zeitspanne erhöht werden, in der ein erneutes Betreten des Raums als fälschliches Verlassen gedeutet würde. Dieser Algorithmus lässt sich demnach nicht ohne Weiteres verbessern.

Bereits die Anwendung des einzelnen Merkmals auf Basis der Accelerometer X-Achse weist eine weit geringere Fehlerrate als der stationäre Fall auf. So wird ein Verlassen zu 100% korrekt erkannt. Lediglich beim Betreten wurden 7 von 135 Fällen fälschlicherweise als Verlassen gedeutet, was einer Fehlerrate von etwa 5% entspricht.

Bei Anwendung des Gyroskop-Merkmals konnte diese Rate noch einmal gesenkt werden, sodass kein fälschliches Verlassen mehr auftritt. Dennoch wurde einer von 135 Fällen Verlassen fälschlicherweise als Betreten interpretiert. Dies entspricht bereits einer Fehlerrate von etwa 0,7%.

Bei Kombination beider Wearablesensoren konnte schließlich eine korrekte Erkennungsrate von 100% für beide Gesten erzielt werden.

Mehrere Nutzer

Die Tests wurden mithilfe eines zweiten Nutzers wiederholt, um die Übertragbarkeit der Merkmale zu testen. Die Testdaten umfassen 41 Ereignisse (Betreten:20, Verlassen:21). Die Ergebnisse sind in Tabelle 6.3 eingetragen.

Tabelle 6.3: Algorithmen Ergebnisse 2. Nutzer

Algorithmus	Falsch-Betreten	Falsch-Verlassen
Stationär	5	3
Accelerometer	8	15
Gyroskop	6	0
Accelero & Gyro	2	0

Erkennbar ist, dass die Kombination von Accelerometer und Gyroskop auch in diesem Fall genauere Werte liefert, als der stationäre Fall (95% zu 80%). Auffällig ist, dass die Werte des Acceleromtermerkmals deutlich größere Abhängigkeit vom Nutzer aufweisen, als die des Gyroskops. Es konnte jedoch bei Betrachtung der Rohdaten ein weiteres Merkmal identifiziert werden, dass für den zweiten Nutzer eine 100-prozentig korrekte Zuordnung zulässt.

6.3 DS2OS

In diesem Abschnitt werden die Ergebnisse des Cognitive Walthrough, der heuristischer Evaluation sowie der weiteren Kriterien der DS2OS Evaluation vorgestellt.

6.3.1 Cognitive Walktrough

Zusammenfassend wird der Cognitive Walkthrough für die Implementierung aller Szenarien betrachtet, da das Verständnis der DS2OS Nutzung szenarienübergreifend ist. Außerdem sollen redundante Aussagen vermieden werden, die bei Betrachtung einzelner Szenarien auftreten. Die Evaluation wurde vom Verfasser der Arbeit durchgeführt. Für jede Aussage wird die Zustimmung auf der Skala von 1 (keine Zustimmung) bis 4 (volle Zustimmung) gegeben, sowie das Verhalten der Nutzer an den jeweiligen Punkten prognostiziert.

Design: Building Blocks

Die Nutzer werden versuchen, den gewünschten Effekt zu erzielen—**3**: Das Verständnis des Szenarios und eine Einführung in des Konzept des VSL ist die Grundvoraussetzung, die es den Nutzern ermöglicht, einzelne Teilaufgaben zu identifizieren und in Blöcke zusammenzufassen. Erleichtert wird die Einteilung in Teilaufgaben durch die Abstraktionshierarchie in 1. Anbindung über die Firmware, 2. Anbindung ans VSL über Adaption Services und 3. Nutzen der Daten durch den Algorithmus. Unklarheiten bei den Sensoren können jedoch deutliche Auswirkungen haben (bei digitalen Sensoren fallen manche Services weg). Bei grobem Verständnis von System und Szenario wird das Resultat schnell ungenau.

Die Nutzer werden erkennen, dass die korrekte Handlung ausgeführt werden kann—**4**: Bei Verständnis des modularen Aufbaus der Szenarien und dem Sinn dahinter kann erfasst werden, dass Funktionalitäten gekapselt werden können. Nutzer wissen, dass der VSL Zugriff auf die Entitäten gibt und der Algorithmus nur mit diesen Daten arbeitet. Wenn der Algorithmus logisch funktionsfähig ist, werden die Nutzer erkennen, dass es einen Weg zur Implementierung geben muss. Da es in allen Szenarien mehrere Teilaufgaben gibt, muss es eine mögliche Zerlegung geben, sodass alle Funktionen in Blöcke kategorisierbar sind.

Die Nutzer werden erkennen, dass die korrekte Handlung zum gewünschten Effekt führen wird—**4**: Die Nutzer erkennen, dass der Algorithmus aus Einzelfunktionen besteht. Wenn er funktionieren soll, müssen diese Funktionen vorhanden sein. Bei schrittweisem

Vorgehen wird dem Nutzer bei jedem Block bewusst, welchen Teil er zum Erreichen des Ziels beiträgt. Bei Erstellung mehrerer Szenarien wird zudem deutlich, dass vorhandene Blöcke wiederverwendet werden können. Beim schrittweisen Vorgehen kann strukturiert Top-down (vom Algorithmus zu den Entitäten) oder Bottom-up (von den Entitäten zum Algorithmus) gearbeitet werden.

Die Nutzer werden den Fortschritt erkennen, wenn sie die korrekte Handlung ausgeführt haben—**4**: Es wird auf das feste Ziel hingearbeitet, den Algorithmus zu implementieren. Daher wird erkennbar, wann alle Anforderungen zur Funktion des Algorithmus erfüllt sind und das Ziel erreicht wurde. Am Ende steht das gesamte Konzept der Implementierung und es gibt keine Unklarheiten beim Prüfen des Algorithmus mehr. Alle Daten sind verfügbar und alle Entitäten angebunden.

Design: Kontexmodell

Die Nutzer werden versuchen, den gewünschten Effekt zu erzielen—**4**: Mit dem Wissen aus der Erstellung der Building Blocks weiß der Nutzer, welcher Kontext für den jeweiligen Block benötigt wird und kann diesen Erstellen.

Die Nutzer werden erkennen, dass die korrekte Handlung ausgeführt werden kann—**2**: DS2OS bietet bereits einfache Datentypen. Aus diesen können beliebig komplexe erweiterte Datentypen erzeugt werden, sodass sämtlicher Kontext gespeichert werden kann. Zusätzlich lassen sich durch die Vergabe von Rechten Einschränkungen beim Zugriff festlegen. Dennoch ist die Dokumentation nicht vollständig, da Funktionen wie Restriktionen und Vererbung nicht ausreichend beschrieben werden. Es fehlt ebenfalls die Dokumentation des Datentyps */basic/list*. Zusätzlich werden für das Erstellen Kenntnisse in XML benötigt, was jedoch auch positiv zu bewerten ist, da der Kontext in einer standardisierten Sprache verfasst wird und kein eigenes Format verwendet wird.

Die Nutzer werden erkennen, dass die korrekte Handlung zum gewünschten Effekt führen wird—**2**: Bei Betrachtung des zugehörigen Building Blocks wird erkennbar, welcher Kontext benötigt wird und zum Ziel führt. Jedoch ist die Modellierung als virtueller oder regulärer Knoten nicht immer ganz eindeutig festzulegen. Für einige Knoten mussten nachträglich der Typ während der Implementierung geändert werden.

Die Nutzer werden den Fortschritt erkennen, wenn sie die korrekte Handlung ausgeführt haben—**3**: Alle benötigten Daten müssen durch eine Repräsentation in Kontextform erfasst worden sein. Ist das erreicht, erkennt der Nutzer den Fortschritt. Jedoch wird

kein Parser bereitgestellt, sodass Kontextmodelle vor Ausführung nicht auf syntaktische Fehler überprüft werden können.

Implementierung: Wiring

Die Nutzer werden versuchen, den gewünschten Effekt zu erzielen—**4**: Nach Erstellung der Building Blocks ist dem Nutzer klar, welche Funktionalitäten ein Service umfasst. Der Nutzer muss nun lediglich die logische Verkabelung durchführen, indem er alle für den Service benötigten Kontextknoten abonniert und auf Handlerfunktionen verweist. Durch die Bereitstellung eines Templates sowie eines Beispielservices wird die Nutzung erleichtert. Auch mit Grundkenntnissen in Java kann die Implementierung erfolgreich durchgeführt werden, da keine komplexen Funktionen benötigt werden. Da für jeden Service dasselbe Template und dieselben Funktionen verwendet werden können, stellt sich schnell eine Routine ein.

Die Nutzer werden erkennen, dass die korrekte Handlung ausgeführt werden kann—**4**: Mithilfe der Building Blocks besitzt der Nutzer einen genauen Plan über die Verkabelung. Indem bereits zusätzlich der Kontext modelliert wurde ist auch klar, welche Verkabelung dieser benötigt, da jeder Knoten mit einem bestimmten Ziel erzeugt wurde.

Die Nutzer werden erkennen, dass die korrekte Handlung zum gewünschten Effekt führen wird—**4**: Durch die Bereitstellung der Beispielservices wird das Schema der Trennung von Verkabelung und Logik praktisch dargestellt. Dort wird klar, wie die Verkabelung funktioniert und der Nutzer kann diese Erkenntnisse leicht auf seine Services übertragen, um zu erkennen, dass die Aktion zum gewünschten Ziel führt. Welche Verkabelungen schließlich zum Ziel führen, ergibt sich aus den Building Blocks.

Die Nutzer werden den Fortschritt erkennen, wenn sie die korrekte Handlung ausgeführt haben—**4**: Wenn anhand der Building Blocks alle Knoten verkabelt sind und jede benötigte Knotenfunktion auf einen Handler verweist ist erkennbar, dass das Ziel erreicht wurde.

Implementierung: Handler

Die Nutzer werden versuchen, den gewünschten Effekt zu erzielen—**4**: Aus der Verkabelung wird direkt auf Handlerfunktionen verwiesen. Bei diesem Verweis wird klar, welche Funktion die Handlerfunktion bereitstellen muss, da diese Verweise bereits mit einer Intention zur Bereitstellung einer bestimmten Funktionalität erstellt wurden.

Die Nutzer werden erkennen, dass die korrekte Handlung ausgeführt werden kann—**4**: Der Nutzer weiß, welcher Input die jeweilige Handlerfunktion bekommt und welcher Output oder Funktionalität erwartet wird. Sie muss lediglich implementiert werden. Die Komplexität der Implementierung ist vom jeweiligen Szenario abhängig.

Die Nutzer werden erkennen, dass die korrekte Handlung zum gewünschten Effekt führen wird—**4**: Da jede Teilfunktion ein Schritt zum Erreichen der Gesamtfunktionalität darstellt und der Nutzer bereits eine Übersicht über alle benötigten Funktionen besitzt, erkennt er auch, wenn ein Schritt zum Erreichen des Ziels abgeschlossen wurde.

Die Nutzer werden den Fortschritt erkennen, wenn sie die korrekte Handlung ausgeführt haben—**4**: Am Ende kann die Funktionalität praktisch durch Ausführung des Services überprüft werden. Wenn alles wie gewünscht funktioniert, ist das Ziel erreicht.

6.3.2 Heuristische Evaluation

Die Auswertung der heuristischen Evaluation erfolgt anhand der in Kapitel 4.1.1.2 definierten Kategorien.

Erkennbarkeit—Durch die Bereitstellung ausführlich kommentierter Templates wird das Learning by Doing unterstützt. Zusätzlich wird der Punkt Konsistenz und Standards eingehalten. Alle Aktionen und Elemente besitzen eindeutige Namen, die konsistent an allen Stellen des Systems verwendet werden. Der Unterschied zwischen virtuellen und regulären Knoten ist nicht sofort klar, vor allem, was zu welchem Zeitpunkt die bessere Wahl ist. Die Nutzung des Ilab-Kurses zur Einarbeitung in das DS2OS ist jedoch gelungen, weil dort schrittweise und praktisch am Code gearbeitet wird. Dort wird auch der Unterschied zwischen den beiden Arten von Kontextknoten klarer aufgezeigt. An einigen Stellen weist die Dokumentation jedoch Schwächen auf. Wie bereits beim Cognitive Walktrough festgestellt wurde, werden einige Funktionen der Kontextmodelle nicht ausreichend beschrieben. Zusätzlich werden die Installation sowie die Parameter der Konfigurationsdatei nicht beschrieben.

Bedienbarkeit—DS2OS bietet als Benutzeroberfläche lediglich eine Konsole an. Diese ist durch den Nutzer nicht weiter personalisierbar. Sie bietet Redo-Unterstützung durch Aufzeichnen einer Befehlshistorie. Es gibt jedoch keine Undo-Unterstützung. Ausgeführte Befehle sind endgültig. Als Eingabehilfe besitzt die Konsole eine Autovervollständigung für Befehle und Pfade. DS2OS bietet keine Sicherung der Daten. Bei Beenden eines KA geht sämtlicher gespeicherter Servicekontext verloren.

Systemstatus—Angaben über den Systemstatus werden nicht gemacht. Es ist vor allem nicht ersichtlich, ob ein Service wirklich läuft oder beendet ist. Der Kontext wird zu Beginn geladen und ist dann vorhanden, ob der Service läuft oder nicht. Lediglich bei Aufruf von Funktionen auf virtuellen Knoten wird eine Fehlermeldung erzeugt. Auf reguläre Knoten kann jedoch weiterhin zugegriffen werden. Bei einem Service, der diesen Wert z.B. mit dem Wert eines Sensors belegen soll, wird einfach keine Änderung mehr geschrieben. Abonnierende Services wissen jedoch nicht, dass der Service beendet ist und gehen von einem konstanten Sensorwert aus.

Fehler—Das Nutzen einer verbreiteten Sprache bei Erstellung der Kontextmodelle bietet den Vorteil, dass mögliches Vorwissen genutzt werden kann und viele Tutorials im Internet genutzt werden können. Somit werden zunächst Fehler gegenüber einer eigenen Syntax minimiert. Jedoch ist kein Parser verfügbar, der zum Erstellzeitpunkt auf Fehler aufmerksam macht. Dies ist ärgerlich, da die Fehlerbehebung aufwändig ist. Fehler fallen erst im laufenden Betrieb z.B. durch fehlende Elemente des Servicekontexts auf. Nach der Korrektur muss der gesamte KA neu gestartet werden, da einmal geladene Kontextmodelle im Nachhinein nicht geändert werden können. Diese grundsätzlich sinnvolle Designentscheidung erweist sich an dieser Stelle als Problem.

Unterstützung beim Erkennen von Fehlern in den Services wird nicht gegeben, da keine Fehlermeldungen direkt ausgegeben werden. Es kann ein Logger genutzt werden, der bei Fehlern einen Logeintrag verfasst. Fehler innerhalb des DS2OS können ebenfalls über Logeinträge identifiziert werden. Bei Eingabe fehlerhafter Befehle in die DS2OS-Konsole werden Fehlermeldungen ausgegeben. Zusätzlich wird jedoch der gesamte Java-Fehlerpfad ausgegeben. Dies ist einerseits unübersichtlich und andererseits für den Nutzer in der Regel eine unnötige Information. Bei willkürlichen Eingaben wird keine Rückmeldung wie beispielsweise „unbekannter Befehl“ gegeben.

Während der gesamten Implementierung ist es nicht einmal vorgekommen, dass DS2OS gecrasht ist. Dies lässt darauf schließen, dass es im Betrieb auch bei fehlerhaften Eingaben stabil läuft. Einzig bei mehrfachem Neustarten eines Agents synchronisierte dieser sich nicht mehr mit einem zweiten Agent, der dauerhaft lief. Erst ein Neustart beider Agents behob das Problem.

6.3.3 Weitere Kriterien

Design

Im Design wurden die Zeiten gemessen, die für die Erstellung der Building Blocks und der Kontextmodelle benötigt wurden. Diese sind in Tabelle 6.4 dargestellt. Zeiten zur Einarbeitung in die jeweiligen Szenarien werden nicht berücksichtigt, da diese unabhängig vom verwendeten System anfallen.

Tabelle 6.4: DS2OS Evaluation zusätzliche Kriterien: Design

Szenario	Building Blocks	Kontextmodell
Beleuchtung	30	20
Heizung	26	20
Luftqualität	26	10
Identifikation	27	10

Diese Ergebnisse zeigen, dass sich die Vorbereitungen für die Implementierung eines Szenarios mit DS2OS in dieser Arbeit durchschnittlich in 43 Minuten durchführen ließen. Von dieser Zeit entfiel etwa ein Drittel auf die Modellierung des Kontext. Werden diese Schritte sorgfältig durchgeführt, entstehen die beim Cognitive Walkthrough genannten Vorteile bei der Implementierung. Da sämtliche Aspekte des Szenarios in den Building Blocks abgebildet werden, muss bei der folgenden Implementierung lediglich blockweise vorgegangen werden. Bei Projekten mit mehreren Programmierern kann anhand der Building Blocks zudem Parallel gearbeitet werden, indem gleichzeitig unterschiedliche Building Blocks implementiert werden. Durch die vorherige Modellierung des Kontexts sind die Schnittstellen jedes Blocks bereits definiert.

Implementierung

Installation—Die Installation des DS2OS umfasst lediglich die Installation von Java auf dem Rechner. Wichtig ist, dass es sich nicht um die Oracle Version handelt, da diese Probleme bei der Autorisierung der Servicezertifikate hervorruft. Daraufhin müssen innerhalb der Konfigurationsdatei Werte wie der Pfad für die Kontextmodelle eingetragen werden. Mithilfe zuvor erstellter Zertifikate kann danach bereits der Knowledge Agent sowie der Konsolenservice gestartet werden. Die benötigte Zeit hängt primär von den zu konfigurierenden Parametern in der Konfigurationsdatei ab, die nicht weiter beschrieben werden, sowie der Zeit für das Erstellen der Zertifikate. Mit Installation der Eclipse IDE und importieren der Templates wurden 80 Minuten benötigt.

In Tabelle 6.5 sind die während der Implementierung erfassten Werte eingetragen. Die angegebenen Zeiten zählen bis zur vollen Funktion, inklusive Fehlersuche und Behebung. Es wird unterschieden nach:

- **HW**: Zeit, die für die physische Verkabelung benötigt wird
- **FW**: Zeit, die für die Implementierung der Firmware benötigt wird
- **Kontext**: Zeit, die für die Implementierung des Kontext benötigt wird
- **Impl**: Zeit, die für die Implementierung der Services benötigt wird

Da die Entscheidung zur Erstellung kumulativer Anfragen an die Sensoren mittels Gateways erst während der Implementierung gefallen ist, entsprechen die Adaptionsservices

Tabelle 6.5: DS2OS Evaluation zusätzliche Kriterien: Implementierung

Service	Zeit (HW/FW/Kontext/Impl)	LoC (W/L)	von Services Genutzt
Adaption			
Ultraschall	187 (30/47/8/102)	98/21	1
Reed	235 (15/20/4/196)	98/21	1
PIR	79 (20/7/7/45)	98/56	3
H2OT	124 (6/77/6/35)	110/24	2
CO2	342 (30/94/5/213)	98/25	1
Lux	149 (35/87/3/24)	100/21	1
Thermostat	13 (0/0/3/10)	92/20	1
IR	376 (30/219/8/119)	170/52	2
Gateways			
Gateway1	139 (-/47/5/87)	75/173	7
Gateway2	49 (-/20/8/21)	170/114	1
AndroidVSL	447 (0/117/15/315)	119/112	2
Logik			
Beleuchtung	85 (0/0/1/84)	130/147	1
Heizung	162 (0/0/1/161)	139/105	1
Luftqualität	153 (0/0/11/142)	128/95	1
Identifikation	129 (0/0/6/123)	118/132	1

den Services, die den jeweiligen Sensor von der Hardware ausgelesen haben. Die Zeiten der Gateways geben den zusätzlichen Aufwand an, die einzelnen Adaptionsservices zusammenzufassen. Daher entfällt dort auch die Zeit zur Verkabelung der Hardware, da dies bereits bei Erstellung der ursprünglichen Adaptionsservices erfolgt ist.

Zu erkennen ist, dass für die Implementierung des AndroidVSL Services die meiste Zeit benötigt wurde. Dies liegt daran, dass für die Implementierung die Templates aus dem Ilab2-Tutorial verwendet wurden. In diesen ist die Anbindung eines Arduino bereits vorbereitet. Es müssen dort lediglich Anpassungen gemacht werden. Da durch den AndroidVSL Servie neue Hardware angebunden werden musste, wurde mehr Zeit benötigt. Es wurde zwar auch dort ein Template verwendet, jedoch musste dies für die Verwendung im VSL stärker modifiziert werden.

Nach der Implementierung verläuft die Kommunikation wie beim Arduino über *http* mittels REST-Anfragen. Innerhalb des VSL können andere Services, genau wie bei allen Adaptionsservices, auf den Kontext des Wearables zugreifen. In Kombination mit der VSL Android-App können nun Daten sämtlicher Androidgeräte in den VSL eingelesen werden.

Die Implementierung des Kontext macht bei sämtlichen Services nur einen geringen Teil der Gesamtzeit aus.

Die Zeit zum Erstellen der Firmware variiert stark und übersteigt teils sogar die Zeit für

die Implementierung des VSL-Services. In dieser Kategorie befindet sich der AndroidVSL-Service ebenfalls im oberen Bereich. Im Vergleich zur Arduino-Firmware musste in diesem Fall jedoch eine Android-App geschrieben werden. Da bereits ein Template mit ähnlicher Funktion zur Verfügung stand, wurde dieses für die Verwendung angepasst.

Die Zeit, die für die Konfiguration der Hardware benötigt wurde, befindet sich in der Regel unterhalb von 30 Minuten. Diese umfasst die Identifikation der benötigten Anschlüsse und die physische Verkabelung. Das Verlegen der Kabel im Raum wurde jedoch nicht mit einberechnet, da diese Zeit je nach Raum und Platzierung des Sensors variiert. In der Zeile des AndroidVSL-Services ist zu erkennen, dass die Verkabelung keine Zeit in Anspruch genommen hat. Dies liegt daran, dass die Sensoren im Wearable bereits konfiguriert sind.

Für die Platzierung der Sensoren im Testlab wurden folgende Zeiten benötigt:

- Ultraschall: 54min
- PIR: 45min
- Helligkeit 100min

Die Zeiten umfassen das Verlöten von benötigten Kabeln, das Verlegen der Kabel sowie das Anbringen des Sensors. Da die Verlegung der Kabel relativ simpel vorgenommen wurde, können bei sorgfältiger Verlegung weit höhere Werte entstehen. Die Nutzung eines Wearables spart in diesem Punkt also mindestens 145 Minuten ein. Wird dieser Wert von der Gesamtimplementierungszeit abgezogen, befindet sich der AndroidVSL Service bereits im zeitlichen Bereich der anderen Andaptionsservices.

Bei der Betrachtung der Lines of Code fallen keine extremen Abweichungen auf. Hervorzuheben ist, dass der AndroidVSL-Service im Vergleich zu den anderen Gateways eine vergleichbare Anzahl an Lines of Code aufweist. Auch hier ist kein erhöhter Aufwand durch das Wearable zu verzeichnen.

Die letzte Spalte gibt an, wie oft auf die einzelnen Services zugegriffen wurde. Werden alle Werte aufsummiert, spart der modulare Aufbau vier Implementierungen bereits vorhandener Module ein. Wird davon ausgegangen, dass die Firmware bereits vorhanden ist und jeder Service inklusive Kontext neu implementiert werden muss, können bis zu 603 Minuten eingespart werden. Die Zeit ergibt sich aus der Summe der Zeiten für Implementierung und Kontexterstellung der eingesparten Services.

6.4 Erkenntnisse

In diesem Abschnitt werden die Erkenntnisse, die sich aus den durchgeführten Tests folgern lassen, vorgestellt.

6.4.1 Testfall I

Allein der Umstand, dass das Wearable bereits ohne Skalierung nahe an den Werten des stationären Sensors liegt, birgt den Vorteil, dass auf die Platzierung eines fest verbauten Sensors verzichtet werden kann. Dies entspricht bereits einem Gewinn an Usability. Nichtsdestotrotz sind zu diesem Zeitpunkt die Schwächen des Szenarios noch nicht behoben. Die zusätzliche Betrachtung der Neigung mithilfe des Accelerometers ermöglicht diesen Faktor in die Berechnung der Helligkeit einzubeziehen. Die resultierenden Werte weisen eine deutliche Erhöhung der Genauigkeit auf, da der mittlere Messfehler nahezu halbiert wurde. Zudem wird durch die Skalierung des Messwerts auch die direkte Sonneneinstrahlung deutlicher erkannt, was die Schwäche des Szenarios löst. Somit kann ein Wearable im Szenario gewinnbringend eingesetzt werden, da sich einerseits die Genauigkeit der Messwerte erhöht und andererseits auf die feste Verlegung eines Luxsensors verzichtet werden kann.

6.4.2 Testfall II

Die Auswertung der Erkennungsraten weist dem stationären Algorithmus eine Gesamtgenauigkeit von 84% zu. Allein die Nutzung des Accelerometers weist bereits eine Genauigkeit von 97% auf, die Nutzung des Gyroskops sogar über 99%. Durch Kombination konnten schließlich 100% der Ereignisse korrekt zugeordnet werden. Das Wearable beweist in diesem Szenario seinen Nutzen durch die erhöhte Genauigkeit und die Ausbesserung der Schwäche des Szenarios. Zusätzlich kann der PIR-Sensor substituiert werden.

Bei Durchführung des Tests mithilfe eines zweiten Testnutzers konnte das Ergebnis bestätigt werden. Das Wearable lieferte auch in diesem Fall genauere Werte, als die stationäre Sensorik (Wearable: 95%, stationär: 80%).

Auch wenn der Stichprobenumfang sowie die Anzahl der Testnutzer noch keine allgemeingültige Aussage zulassen, lassen die Testwerte einerseits darauf schließen, dass individuelle Merkmale identifiziert werden können, die eine sichere Zuordnung der Gesten zulassen. Andererseits lässt das Ergebnis aus der Anwendung des erstellten Algorithmus den Schluss zu, dass allgemeingültige Merkmale existieren, die eine genauere Zuordnung als die stationäre Logik zulassen. Der Algorithmus basiert auf den Merkmalen des ersten Nutzers und wurde unverändert auf den zweiten Nutzer übertragen. Dennoch lieferte er genauere Ergebnisse als die stationäre Lösung.

6.4.3 DS2OS Evaluation

Die Evaluation zeigt, dass sich DS2OS für ein Rapid Prototyping von Smart Space Szenarien eignet. Der Umgang ist weitestgehend intuitiv. Verbesserungen sind vor allem in

der Dokumentation einiger Funktionen wünschenswert. Zudem sollten Kontextmodelle vor dem Laden ins VSL prüfbar sein, um den bei einer Korrektur erforderlichen Neustart des KA und damit aller laufenden Services zu vermeiden.

Die Zeiten für die Implementierungen der Szenarien hängen von der Anzahl und dem Typ der verwendeten Sensoren ab und befinden sich zwischen 8,5 und 20 Stunden.

Ein zusätzlicher Aufwand durch das Wearable ist minimal. Der erhöhte Zeitaufwand beim Programmieren wird durch die Substitution der stationären Sensorik und der damit eingesparten Zeit nahezu ausgeglichen. Innerhalb des VSL entsteht kein zusätzlicher Aufwand durch Umgang mit dem Wearable Kontext.

Kapitel 7

Fazit

In diesem Kapitel erfolgt die Beantwortung der Leitfrage. Dies geschieht auf Basis der Erkenntnisse, die im Kapitel 6 bei der Auswertung der Testfälle zur Evaluation des Wearablenutzens gewonnen wurden.

Abschließend wird ein Ausblick auf mögliche weiterführende Arbeiten gegeben.

7.1 Beantwortung der Leitfrage

Im Laufe dieser Arbeit konnte gezeigt werden, dass Wearables gewinnbringend in Smart Space Szenarien integriert werden können. Letztlich bleibt noch die allgemeine Beantwortung der Leitfrage.

Welche Vorteile bietet die Integration von Wearables in IoT Szenarien?

Die Beantwortung lässt sich mit den Erkenntnissen dieser Arbeit in vier Punkte gliedern.

7.1.1 Substitution von stationärer Sensorik

Die Messungen haben gezeigt, dass bei Verwendung eines Wearables mindestens die Genauigkeit der stationären Sensoren erreicht werden konnte. Bei Überschreitung dieser Grenze können stationäre Sensoren ohne negative Auswirkungen substituiert werden. Dieses gewährt ein Gewinn an Usability. Es kann Sensorik verwendet werden, die dank implementierter App und der nicht erforderlichen Verkabelung sofort verfügbar ist. Dies ist für ein Rapid Prototyping von Vorteil. Zusätzlich kann das Wearable in Umgebungen eingesetzt werden, in denen das Anbringen stationärer Sensoren nicht möglich ist.

7.1.2 Genauere Messwerte

Neben der reinen Substitution von stationären Sensoren wurde zusätzlich eine Erhöhung der Genauigkeit erreicht. Sicherlich ist der jeweilige Gewinn szenarienabhängig, jedoch konnte diese Arbeit zeigen, dass ein Wearable als Alternative zu stationären Lösungen in Betracht gezogen werden sollte. Die beiden gezeigten Szenarien sind nur Beispiele für ein mögliches Vorgehen beim Einsatz eines Wearables. Es konnte jedoch in Beiden ein signifikanter Gewinn an Genauigkeit gegenüber der stationären Sensorik erreicht und so individuelle Schwächen beider Szenarien gelöst werden. Durch die Vielzahl an verfügbaren Sensoren sind weitere Szenarien denkbar, in denen beispielsweise auch die Vitalwertüberwachende Sensorik eingesetzt werden kann.

7.1.3 Geringe Kosten

Dadurch, dass in dieser Arbeit davon ausgegangen wird, dass der Nutzer das Wearable bereits mitbringt, fallen keinerlei Kosten für die Sensorik sowie den Einbau an. Somit können neben den bereits genannten positiven Effekten ebenfalls Kosten eingespart werden. Die Nutzung von Android als Betriebssystem garantiert zudem die Kompatibilität vieler Endgeräte.

7.1.4 Geringer Aufwand

Der Aufwand der Wearableintegration ergibt sich aus der Evaluation des DS2OS. Es ist zu erkennen, dass die Erstellung des Android-Adaptionsservices zwar mehr Zeit als einzelne Adaptionsservices benötigt, jedoch mit diesem einen Service sämtliche Sensoren des Androidgeräts ausgelesen werden. Oberhalb des Adaptionsservices unterscheidet sich der Umgang nicht von dem mit den Daten der stationären Sensoren. Es entfällt das Verlegen stationärer Sensorik. Statt der Firmware für den Arduino musste eine Android-App erstellt werden. Dies ist jedoch ein einmaliger Aufwand. Diese App kann nun immer wieder verwendet werden, da der Zugriff auf die Sensoren innerhalb von Android einheitlich ist. Durch Integration in den Appstore kann diese sogar einfach verbreitet werden. Da sich Android-Smartphones und -Watches nur geringfügig unterscheiden, kann mit Änderung nur einer Zeile in der Manifest-Datei der App zwischen Smartwatch und Smartphone-App umgeschaltet werden.

Zusammenfassend lässt sich die Aussage treffen, dass die Verwendung eines Wearables keinen Mehraufwand darstellt. Auch der Umgang mit den Daten innerhalb des VSL ist identisch mit dem aller anderen Entitäten.

7.1.5 Fazit

Da gezeigt wurde, dass der Einsatz von Wearables in IoT Szenarien im Allgemeinen Vorteile birgt, sollte bei der Erstellung neuer Szenarien stets in Betracht gezogen werden, ob ein Wearable zur Erfassung der Daten genutzt werden kann.

7.2 Ausblick

Weiterführende Arbeiten können im Rahmen des in dieser Arbeit ausgewählten Szenarios *Identifikation* durchgeführt werden. Es kann eine repräsentative Nutzerstudie durchgeführt werden, um allgemein gültige Merkmale zu identifizieren. Zusätzlich können für die Erkennung der Gesten lernbasierte Algorithmen verwendet werden.

Allgemein sollten weitere Szenarien auf die Nutzbarkeit eines Wearables hin untersucht werden. Dort sollten auch Sensoren verwendet werden, die in dieser Arbeit noch nicht verwendet wurden, um deren Nutzen explizit zu erfassen.

Anhang A

Templates

Listing A.1: ServiceTemplate.java, registerSubscriptions

```
/**
* Register all necessary subscriptions. To know which nodes exist have a look at the model
* specified with the Id in myServiceModelId. TODO: Change to connect to your subscription
* handlers
*
* @throws VslException
*        If one of the subscriptions didn't work.
*/
public void registerSubscriptions() throws VslException {
// Register a callback for each node you want to be notified for.

// If you are interested in changes in a whole subtree, you can also
// register to the subtree
// root and differentiate based on the address that is passed to your
// callback. Be aware that such an implementation can result in many
// notifications depending on the nodes of your subtree.
// Thus the first method, to subscribe to each node separately, may be
// more efficient.

// TODO: replace this example with your own handlers if necessary.
// If none are needed, you can remove this (The service may throw an
// exception if the node you register as virtual doesn't exist.)
connector.subscribe(myKnowledgeRoot + "/regularNode", new VslSubscriber() {

@Override
public void notificationCallback(String address) throws VslException {
h.regularNodeHandler(address);
}
}); }
```

Listing A.2: ServiceTemplate.java, registerVirtualNodeHandlers

```
/**
* Register all necessary VirtualNodeHandlers To know which nodes exist have a look at the model
* specified with the Id in myServiceModelId. TODO: Change to connect to your subscription
* handlers
*
* @throws VslException
```

```
*        If one of the VirtualNodehandlers couldn't be registered correctly.
*/
public void registerVirtualNodeHandlers() throws VslException {

// TODO: replace this example with your own handlers if necessary.
// If none are needed, you can remove this (The service may throw an
// exception if the node you register as virtual doesn't exist.)

// IMPORTANT:
// Instead of the VslVirtualNodeHandler class you can use the VirtualNodeAdapter class.
// This allows you to ommit methods that are not required/supported by your service.
// Using these will then return errors to the calling service. A get request with parameters
// will be mapped to the normal get without parameters if ommited.

connector.registerVirtualNode(myKnowledgeRoot + "/someVnode", new VslVirtualNodeHandler() {

@Override
public void set(String address, VslNode value, VslIdentity identity) throws VslException {
h.doSomething();
}

@Override
public VslNode get(String address, VslAddressParameters params, VslIdentity identity)
throws VslException {
return null;
}

@Override
public void subscribe(String address) throws SubscriptionNotSupportedException, VslException {
}

@Override
public void unsubscribe(String address) throws VslException {
}

}); }
```

Listing A.3: ServiceTemplateHandler.java, Handler Funktionen

```
/**
* Implement the functionality of the nodes here:
*/
public void regularNodeHandler(String name) {
// TODO
}

public void doSomething() {
// TODO
}

public VslNode doAndReturnSomething() {
return nodeFactory.createImmutableLeaf("TODO");
}
```

Anhang B

Heuristik Liste

Erkennbarkeit

- Übereinstimmung zwischen System und realer Welt (nutzerorientierte Kommunikation, Verwendung vertrauter Konzepte / Informationen sollen Nutzer natürlich und logisch erscheinen)
- Erkennen vor Erinnern (intuitive Erkennbarkeit von Objekten, Aktionen und Optionen. Anleitung, falls nötig)
- Lernförderlichkeit (Learning by Doing durch Anleitungen und Navigationshilfen unterstützt)
- Konsistenz und Standards (Konvention für Begriffe, Situationen und Aktionen)
- Erwartungskonformität (Dialoge bei ähnlichen Aufgaben vergleichbar und an erwarteter Position)

Bedienbarkeit

- Benutzerkontrolle und Freiheit (einfaches Verlassen unbeabsichtigter Zustände, Undo & Redo Unterstützung)
- Steuerbarkeit (Kontrolle über Dialoge, Eingabehilfen, Beenden ohne Datenverlust)
- Flexibilität und effiziente Nutzung (Möglichkeit häufige Aktionen auf eigene Bedürfnisse zuzuschneiden)
- Individualisierbarkeit (an Nutzerpräferenzen anpassbar um Effektivität, Effizienz und Zufriedenstellung zu steigern)
- Aufgabenangemessenheit (Vorhandensein benötigter Funktionen, unterstützend und entlastend gestaltet)

- Ästhetisches und minimalistisches Design (Vermeidung irrelevanter Informationen in einem Dialog)
- Wahrnehmungssteuerung (Gruppierungen, Farbgestaltung und Informationsreduktion um relevante Informationen hervorzuheben)

Systemstatus

- Sichtbarkeit des Systemstatus (Feedback über aktuellen Zustand)
- Selbstbeschreibungsfähigkeit (Anzeige des Systemstatus einheitlich und unmittelbar, Detailgrad bestimmbar durch Nutzer)

Fehler

- Fehler vermeiden (Design, das Fehler vermeidet)
- Unterstützung beim Erkennen, Verstehen und Bearbeiten von Fehlern (konstruktive, präzise Fehlermeldungen in klarer Sprache mit Lösung)
- Fehlertoleranz (Fehlermeldungen deutlich, Hinweise über Art und Handlungszusammenhang. Information über irreversible Handlungen)
- System- und Datensicherheit (System arbeitet bei Fehleingaben und hoher Auslastung stabil und ohne Datenverlust)

Anhang C

Kontextbaum

```
./androidvsl
  ./androidvsl/acceleration
    ./androidvsl/acceleration/xAxis
    ./androidvsl/acceleration/yAxis
    ./androidvsl/acceleration/zAxis
  ./androidvsl/accelerometer
    ./androidvsl/accelerometer/xAxis
    ./androidvsl/accelerometer/yAxis
    ./androidvsl/accelerometer/zAxis
  ./androidvsl/gyroscope
    ./androidvsl/gyroscope/xAxis
    ./androidvsl/gyroscope/yAxis
    ./androidvsl/gyroscope/zAxis
  ./androidvsl/lux
./co2adaption
  ./co2adaption/co2
./co2logik
  ./co2logik/alert
  ./co2logik/message
./gateway7
  ./gateway7/co2
  ./gateway7/h2o
  ./gateway7/height
  ./gateway7/lux
  ./gateway7/pir
  ./gateway7/reed
  ./gateway7/temp
./gateway8
  ./gateway8/rgb
    ./gateway8/rgb/brighter
    ./gateway8/rgb/color
      ./gateway8/rgb/color/desired
```

```
    ./gateway8/rgb/darker
    ./gateway8/rgb/isOn
      ./gateway8/rgb/isOn/desired
  ./gateway8/white
    ./gateway8/white/brighter
    ./gateway8/white/darker
    ./gateway8/white/isOn
      ./gateway8/white/isOn/desired
    ./gateway8/white/level
      ./gateway8/white/level/desired
./h2otempadaption
  ./h2otempadaption/h2o
  ./h2otempadaption/temp
./identlogik
  ./identlogik/enter
  ./identlogik/id
./iradaption
  ./iradaption/rgb
    ./iradaption/rgb/brighter
    ./iradaption/rgb/color
      ./iradaption/rgb/color/desired
    ./iradaption/rgb/darker
    ./iradaption/rgb/isOn
      ./iradaption/rgb/isOn/desired
  ./iradaption/white
    ./iradaption/white/brighter
    ./iradaption/white/darker
    ./iradaption/white/isOn
      ./iradaption/white/isOn/desired
    ./iradaption/white/level
      ./iradaption/white/level/desired
./luxadaption
  ./luxadaption/lux
./luxlogik
./piradaption
  ./piradaption/pir
  ./piradaption/timer
./reedadaption
  ./reedadaption/reed
./templogik
./thermostatadaption
  ./thermostatadaption/level
    ./thermostatadaption/level/desired
./ultrasonicadaption
  ./ultrasonicadaption/height
```

Anhang D

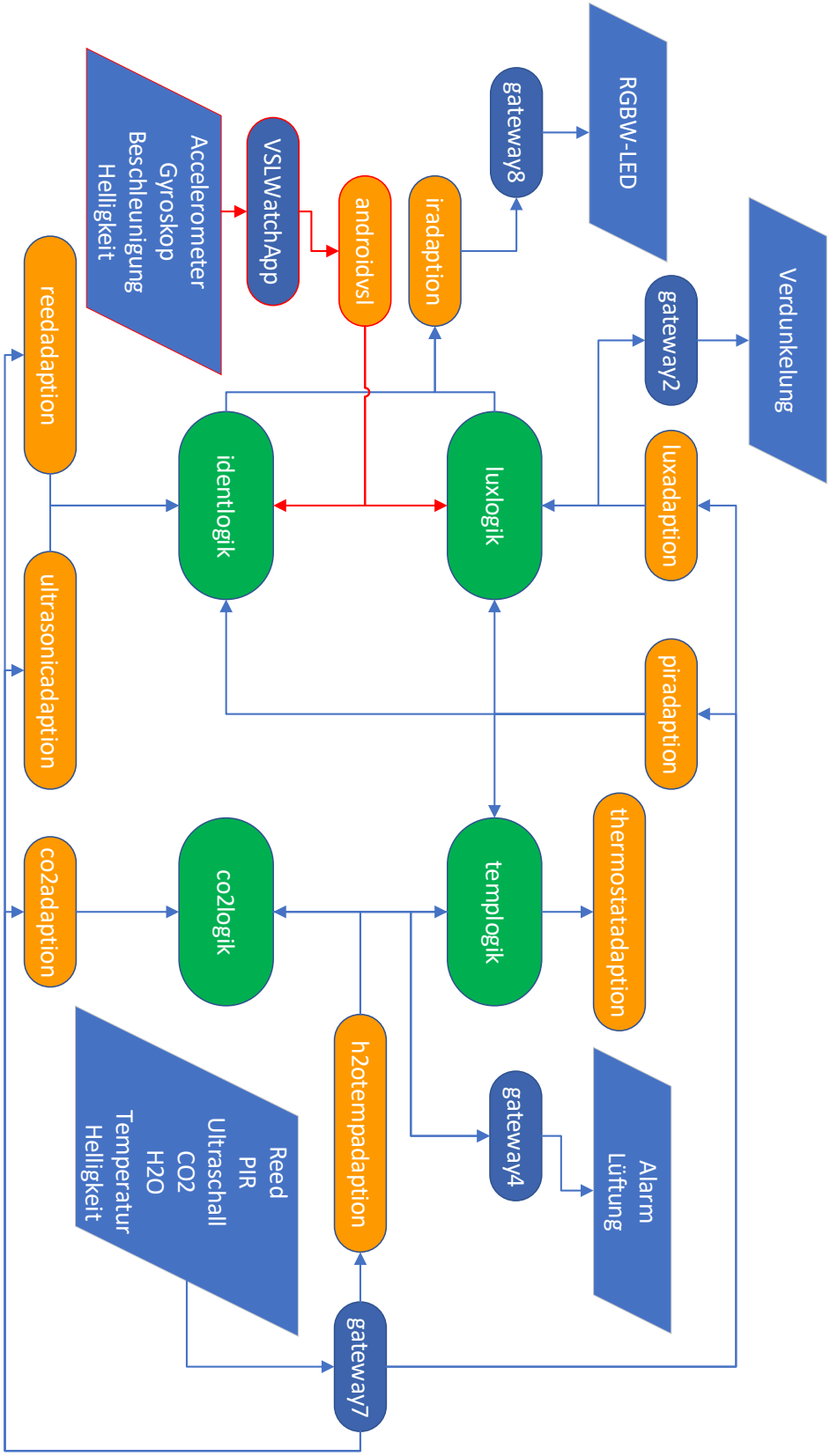
Service Topologie

Das Diagramm enthält alle verwendeten VSL Services. Die Farben entsprechen denen der Building Blocks aus Kapitel 4.2.2:

- Grün Logik
- Gelb Adaption
- Blau Firmware

Wearableservices sind wiederum rot umrandet.

Die Pfeile geben die Richtung des Datenflusses an.


Accelerometer
Gyroskop
Beschleunigung
Helligkeit
VSLWatchApp
androidvsl
iradaption
gateway8
RGBW-LED
reedadaption
identlogik
luxlogik
gateway2
Verdunkelung
luxadaption
ultrasonicadaption
piradaption
co2adaption
co2logik
templogik
thermostatadaption
Reed
PIR
Ultraschall
CO2
H2O
Temperatur
Helligkeit
h2otempadaption
gateway4
Alarm
Lüftung
gateway7

Literaturverzeichnis

[1] M.-O. Pahl, "Distributed Smart Space Orchestration," Ph.D. dissertation, Technische Universität München, München, Jun. 2014.

[2] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, pp. 94–104, 9 1991. [Online]. Available: http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[3] P. Härtinger, "Survey and implementation of smart space scenarios," B.S. Thesis, Technische Universität München, 2014.

[4] R. Akhavian and A. Behzadan, "Wearable sensor-based activity recognition for data-driven simulation of construction workers' activities," in *Winter Simulation Conference (WSC), 2015.* IEEE, 2015, pp. 3333–3344.

[5] L. Wang, T. Gu, X. Tao, and J. Lu, "Sensor-based human activity recognition in a multi-user scenario," in *European Conference on Ambient Intelligence.* Springer, 2009, pp. 78–87.

[6] G. Sprint, D. J. Cook, and D. L. Weeks, "Designing wearable sensor-based analytics for quantitative mobility assessment," in *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on.* IEEE, 2016, pp. 1–8.

[7] J. Dahmen, A. La Fleur, G. Sprint, D. Cook, and D. L. Weeks, "Using wrist-worn sensors to measure and compare physical activity changes for patients undergoing rehabilitation," in *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on.* IEEE, 2017, pp. 667–672.

[8] International Data Corporation (IDC), "Worldwide smartwatch market will see modest growth in 2016 before swelling to 50 million units in 2020, according to idc," stand: 2017-09-27. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS41736916

[9] International Data Corporation (IDC), "Xiaomi and apple tie for the top position as the wearables market swells 17.9according to idc," stand: 2017-09-27. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS42707517

[10] Apple Inc, stand: 2017-09-30. [Online]. Available: https://developer.apple.com/documentation

[11] Google LLC, stand: 2018-01-25. [Online]. Available: https://developer.android.com/guide/topics/sensors/sensors_overview.html

[12] Xiaomi, stand: 2017-09-30. [Online]. Available: http://www.mi.com/en/miband2/

[13] Fitbibt Inc., stand: 2017-09-30. [Online]. Available: https://dev.fitbit.com/de

[14] R. Blasco, l. Marco, R. Casas, D. Cirujano, and R. Picking, "A smart kitchen for ambient assisted living," *Sensors*, vol. 14, no. 1, pp. 1629–1653, 2014. [Online]. Available: http://www.mdpi.com/1424-8220/14/1/1629

[15] R. Das and G. Tuna, "Design and implementation of a smart home for the elderly and disabled," 2015.

[16] J. Liu, W. Zhang, X. Chu, and Y. Liu, "Fuzzy logic controller for energy savings in a smart led lighting system considering lighting comfort and daylight," *Energy and Buildings*, vol. 127, pp. 95 – 104, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378778816304352

[17] M. Li and H. J. Lin, "Design and implementation of smart home control systems based on wireless sensor networks and power line communications," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 7, pp. 4430–4442, July 2015.

[18] J. Serra, D. Pubill, A. Antonopoulos, and C. Verikoukis, "Smart hvac control in iot: Energy consumption minimization with user comfort constraints," vol. 2014, 05 2014.

[19] A. Javed, H. Larijani, A. Ahmadinia, R. Emmanuel, M. Mannion, and D. Gibson, "Design and implementation of a cloud enabled random neural network-based decentralized smart controller with intelligent sensor nodes for hvac," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 393–403, April 2017.

[20] X. Chen, Y. Zheng, Y. Chen, Q. Jin, W. Sun, E. Chang, and W.-Y. Ma, "Indoor air quality monitoring system for smart buildings," in *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '14. New York, NY, USA: ACM, 2014, pp. 471–475. [Online]. Available: http://doi.acm.org/10.1145/2632048.2632103

[21] B. Fang, Q. Xu, T. Park, and M. Zhang, "Airsense: An intelligent home-based sensing system for indoor air quality analytics," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '16. New York, NY, USA: ACM, 2016, pp. 109–119. [Online]. Available: http://doi.acm.org/10.1145/2971648.2971720

[22] H. Pohl, M. Hettig, O. Karras, H. Ötztürk, and M. Rohs, "Capcouch: Home control with a posture-sensing couch," in *Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers*, ser. UbiComp/ISWC'15 Adjunct. New York, NY, USA: ACM, 2015, pp. 229–232. [Online]. Available: http://doi.acm.org/10.1145/2800835.2800932

[23] R. Brotman, W. Burleson, J. Forlizzi, W. Heywood, and J. Lee, *Building change: Constructive design of smart domestic environments for goal achievement.* Association for Computing Machinery, 4 2015, vol. 2015-April, pp. 3083–3092.

[24] S. Mennicken, D. Kim, and E. M. Huang, "Integrating the smart home into the digital calendar," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ser. CHI '16. New York, NY, USA: ACM, 2016, pp. 5958–5969. [Online]. Available: http://doi.acm.org/10.1145/2858036.2858168

[25] J. Lu and K. Whitehouse, "Suncast: Fine-grained prediction of natural sunlight levels for improved daylight harvesting," in *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, ser. IPSN '12. New York, NY, USA: ACM, 2012, pp. 245–256. [Online]. Available: http://doi.acm.org/10.1145/2185677.2185738

[26] M. Gupta, S. S. Intille, and K. Larson, *Adding GPS-Control to Traditional Thermostats: An Exploration of Potential Energy Savings and Design Challenges*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 95–114. [Online]. Available: https://doi.org/10.1007/978-3-642-01516-8_8

[27] J. Scott, A. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar, "Preheat: Controlling home heating using occupancy prediction," in *Proceedings of the 13th International Conference on Ubiquitous Computing*, ser. UbiComp '11. New York, NY, USA: ACM, 2011, pp. 281–290. [Online]. Available: http://doi.acm.org/10.1145/2030112.2030151

[28] J. Lu, T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field, and K. Whitehouse, "The smart thermostat: Using occupancy sensors to save energy in homes," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '10. New York, NY, USA: ACM, 2010, pp. 211–224. [Online]. Available: http://doi.acm.org/10.1145/1869983.1870005

[29] C. Koehler, B. D. Ziebart, J. Mankoff, and A. K. Dey, "Therml: Occupancy prediction for thermostat control," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '13. New York, NY, USA: ACM, 2013, pp. 103–112. [Online]. Available: http://doi.acm.org/10.1145/2493432.2493441

[30] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen, "The gator tech smart house: A programmable pervasive space," *Computer*, vol. 38, no. 3, pp. 50–60, Mar. 2005. [Online]. Available: http://dx.doi.org/10.1109/MC.2005.107

[31] L. Spassova, J. Schöning, G. Kahl, and A. Krüger, "Innovative retail laboratory," in *Roots for the Future of Ambient Intelligence. European Conference on Ambient Intelligence (AmI-09), 3rd, November.* Citeseer, 2009, pp. 18–21.

[32] V. Srinivasan, J. A. Stankovic, and K. Whitehouse, "Using height sensors for biometric identification in multi-resident homes," in *Pervasive*, ser. Lecture Notes in Computer Science, vol. 6030. Springer, 2010, pp. 337–354.

[33] F. Vergari, S. Bartolini, F. Spadini, A. D'Elia, G. Zamagni, L. Roffia, and T. S. Cinotti, "A smart space application to dynamically relate medical and environmental information," in *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*, March 2010, pp. 1542–1547.

[34] C. Bi, G. Xing, T. Hao, J. Huh, W. Peng, and M. Ma, "Familylog: A mobile system for monitoring family mealtime activities," in *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, March 2017, pp. 21–30.

[35] C.-C. J. Huang, R. Yang, and M. W. Newman, "The potential and challenges of inferring thermal comfort at home using commodity sensors," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ser. UbiComp '15. New York, NY, USA: ACM, 2015, pp. 1089–1100. [Online]. Available: http://doi.acm.org/10.1145/2750858.2805831

[36] M. D. Rodríguez and J. Favela, *Evaluation of an Agent Framework for the Development of Ambient Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 374–383. [Online]. Available: https://doi.org/10.1007/978-3-540-88875-8_59

[37] R. Grimm, "One.world: experiences with a pervasive computing architecture," *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 22–30, July 2004.

[38] A. Kropp, "Usability analysis of a smart space middleware," B.S. Thesis, Technische Universität München, 2016.

[39] F. Sarodnick and H. Brau, *Methoden der Usability Evaluation: Wissenschaftliche Grundlagen und praktische Anwendung*, 2nd ed. Bern: Huber, 2011.

[40] A. Schulz, "As_bh1750," stand: 2018-03-30. [Online]. Available: https://github.com/hexenmeister/AS_BH1750

[41] Adafruit, "Adafruit dht humidity & temperature unified sensor library," stand: 2018-03-30. [Online]. Available: https://github.com/adafruit/DHT-sensor-library

[42] T. Eckell, “Newping arduino library for arduino,” stand: 2018-03-30. [Online]. Available: https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home

[43] K. Shirriff, “Irremote arduino library,” stand: 2018-03-30. [Online]. Available: https://github.com/z3t0/Arduino-IRremote

[44] A. Hubel, “A self–adapting Map-Based Interface for Smart Spaces,” M.S. Thesis, Technische Universität München, Garching bei München, 2017.